UNIVERSIDADE DA CORUÑA

COMPUTER ARCHITECTURE GROUP
UNIVERSITY OF A CORUÑA
GAC.UDC.ES

Big Data Evaluator: User Guide

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

January X, 2020

Contents

1 Overview

The Big Data Evaluator (BDEv)[1] is an benchmarking tool that extracts valuable information about the performance, resource utilization, energy efficiency and microarchitectural behaviour of several Big Data 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 of the configuration 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 by 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 at `http://bdev.des.udc.es`.

2 Citation

If you have used 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 Framework download

By default, Big Data processing frameworks are stored in the `$BDEv_HOME/solutions/dist` directory. BDEv includes the following frameworks within its distribution package:

- Hadoop-YARN version 2.9.2
- Flame-MR version 1.2

The rest of the frameworks and versions are not included in order to keep the BDEv distribution into a reasonable size. To use one of them in the evaluations, the user must download and store it in the following directory:

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

Where `$FRAMEWORK_NAME` and `$FRAMEWORK_VERSION` must be the framework name and version, respectively, as they appear in the `$BDEv_HOME/experiment/solutions.lst` file.

4 Configuration of the experiments

The configuration of a experiment affects the following files:

- hostfile
- bdev-conf.sh
- system-conf.sh
- experiment-conf.sh

[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.

- 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
```

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.

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 using the dstat tool

`ENABLE_ILO`: records system power consumption via the HPE iLO interface (requires its previous instalation)

`ENABLE_RAPL`: measures CPU and memory power consumption using a built-in RAPL tool implemented with PAPI

`ENABLE_OPROFILE`: counts microarchitecture-level events via Oprofile

`ENABLE_BDWATCHDOG`: monitors resource usage using the BDWatchdog framework[2]

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 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`.

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

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.

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.

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.

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.

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.

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. From version 3.3 onwards, BDEv automatically switches to command mode if no solution is selected in this file. This can be useful to run any application in the cluster while taking advantage of all the monitoring features 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 with 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 time 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 dstat for monitoring resource usage (i.e., ENABLE_STAT is true), the metrics from the execution of a benchmark are stored 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 at `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 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`).

When using BDWatchdog for monitoring resource usage (i.e., ENABLE_BDWATCHDOG is true), the metrics are stored in the OpenTSDB database managed by BDWatchdog. The appropriate REST endpoint for OpenTSDB must be properly configured in `bdev-conf.sh`. The time series plots stored in OpenTSDB can be visualized using the BDWatchdog's web interface. Furthermore, the time stamping feature provided by BDWatchdog can be used together with BDEv to manage the generation of "start" and "end" UNIX timestamps for each benchmark. To do so, the MongoDB database managed by BDWatchdog must also 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 can be obtained at https://bdwatchdog.readthedocs.io.

6.4 Energy monitoring

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 for 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) and PP1. Separated values for the memory components of each socket are also provided. The summary directory contains the average power values among the slave nodes and their total energy consumption values for each moment. The `gen_graphs.eps` script allows to generate the graphs after 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 monitoring are contained in `{cluster_size}/{framework}/{benchmark}_{num_execution}/oprofile_records`. As with the other 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, shown in the `graphs/oprofile` directory.

```
report_BDEv_27_10_17-00-00
├─ gen_graphs.sh ........................ Script to generate all resource/RAPL graphs
├─ hostfile ............................. Nodes used in the evaluation
├─ hostfile.gbe ......................... GbE nodes used in the evaluation
├─ hostfile.ipoib ....................... IPoIB nodes used in the evaluation
├─ log .................................. Execution log
├─ summary .............................. Experiments configuration and main results
├─ 5 .................................... Output directory for cluster size 5
│  ├─ Hadoop-2.9.2-GbE .................. Output directory for Hadoop-2.9.2-GbE
│  └─ Hadoop-2.9.2-IPoIB ................ Output directory for Hadoop-2.9.2-IPoIB
│     ├─ etc
│     │  └─ hadoop ...................... Hadoop configuration directory
│     ├─ logs ........................... Hadoop log directory
│     ├─ wordcount_1 .................... Output directory for the 1st execution of Wordcount
│     │  ├─ elapsed_time ................ Elapsed seconds
│     │  ├─ output ...................... Workload output
│     │  ├─ oprofile_records ............ Microarchitecture-level metrics directory
│     │  │  ├─ log ...................... Oprofile monitoring log
│     │  │  ├─ sum.csv .................. Oprofile events summation among the nodes
│     │  │  ├─ node-0 ................... Node 0 (master) Oprofile statistics directory
│     │  │  ├─ node-1 ................... Node 1 Oprofile statistics directory
│     │  │  └─ ...
│     │  ├─ rapl_records ................ Energy/power monitorization directory
│     │  │  ├─ log ...................... RAPL graphs generation log
│     │  │  ├─ gen_graphs.sh ............ Script to generate RAPL graphs
│     │  │  ├─ avg ...................... Average RAPL statistics directory
│     │  │  ├─ node-0 ................... Node 0 (master) RAPL statistics directory
│     │  │  ├─ node-1 ................... Node 1 RAPL statistics directory
│     │  │  └─ ...
│     │  ├─ stat_records ................ Resource utilization directory
│     │  │  ├─ log ...................... Stat graphs generation log
│     │  │  ├─ gen_graphs.sh ............ Script to generate resource utilization graphs
│     │  │  ├─ avg ...................... Average statistics directory
│     │  │  ├─ node-0 ................... Node 0 (master) statistics directory
│     │  │  ├─ node-1 ................... Node 1 statistics directory
│     │  │  └─ ...
│     │  └─ bdwatchdog .................. Temporary files from BDWatchdog daemons
│     │     ├─ Atop_hostname.err ........ Error file for atop daemon
│     │     ├─ Atop_hostname.out ........ Output file for atop daemon
│     │     ├─ Atop_hostname.log ........ Log file for atop daemon
│     │     └─ ...
│     ├─ wordcount_2 .................... Output directory for the 2nd execution of Wordcount
│     └─ ...
├─ 9 .................................... Output directory for cluster size 9
└─ graphs ............................... Performance graphs directory
   ├─ log ............................... Graph generation log
   ├─ wordcount.eps ..................... Time results for the Wordcount benchmark (graph)
   ├─ wordcount.dat ..................... Time results for the Wordcount benchmark (data file)
   ├─ oprofile .......................... Oprofile summary graphs
   │  ├─ wordcount ...................... Graphs for wordcount
   │  └─ ...
   └─ rapl .............................. RAPL summary graphs
      ├─ wordcount ...................... Graphs for wordcount
      └─ ...
```

Figure 1: BDEv output directory structure

A About Flink configuration

Recent versions of Flink do not support Hadoop compatibility by default. In order to enable it, 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 jar must be placed inside the `lib` folder of the distribution.

B About batch-queueing schedulers

As most 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 cases. The correct behaviour of BDEv under this kind of systems has been tested with Open Grid Scheduler/GE (OGS/GE), Son of Grid Engine (SGE) and Slurm.

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

C About Modules Environment

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

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 BDWatchdog, the cluster nodes must be properly configured to allow regular users to access hardware performance counters. Moreover, the microarchitecture-level monitoring using Oprofile also requires this specific configuration. The following script contains the minimum set of commands that are needed in order to enable this feature in each node of the cluster.

```
modprobe msr # Load MSR kernel module
chmod 666 /dev/cpu/*/msr # Enable read permisions to MSR registers
echo 0 > /proc/sys/kernel/perf_event_paranoid # Enable access to hardware counters
echo '$USER ALL=NOPASSWD:/sbin/setcap' >> /etc/sudoers # Allow setcap for the user
    running BDEv
```

Further information about this topic can be obtained at http://icl.cs.utk.edu/projects/papi/wiki/PAPITopics:RAPL_Access.

E About BDWatchdog

When monitoring energy consumption using turbostat, regular users must have access to 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 nethogs.

F System Requirements

The following packages need to be installed:

- Java JRE 1.8 (or higher)
- Expect 1.1 (needed if timeout is enabled)
- Gnuplot 4.4 (needed for generating the graphs)
- MPI[3] (needed for DataMPI)
- Oprofile 1.2.0 (needed for microarchitecture-level event counting)
- Python 3 (needed for BDWatchdog)
  - Required Python modules: python-daemon, requests
- turbostat tool (needed for BDWatchdog when enabling turbostat)

G Contact

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

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

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

[3]DataMPI has been tested with MVAPICH2