Next Generation of HEP CPU benchmarks

Domenico Giordano, Evangelia Santorinaiou
CERN, 1211 Geneva CH
E-mail: Domenico.Giordano@cern.ch, evsantorinaiou@gmail.com

Abstract. HEPSPEC-06(HS06) is a decade old suite used to benchmark CPU resources for WLCG. Its adoption spans from hardware vendors, to site managers, funding agencies and software experts. Although it is stable, reproducible and accurate, it is reaching the end of its life. Initial hints of lack of correlations with HEP applications have been collected. Looking for suitable alternatives the HEPiX Benchmarking Working Group has evaluated SPEC CPU 2017 with a number of fast benchmarks. The studies that have been done so far do not show any major advantage in adopting SPEC CPU 2017 with respect to HS06.

A suite based on workloads that HEP experiments run can be an alternative to industrial standard benchmarks. The adoption by LHC experiments of modern software development techniques simplifies the ability to package, distribute and maintain a field-specific benchmark suite. The HEPiX Benchmarking Working Group is actively working to make this possible.

This report summarises the progress of the HEPiX Benchmarking Working Group in building a benchmarking suite based on HEP workloads. Comparisons of results with SPEC CPU 2017 and HS06 will be discussed.

1. Introduction

Since 2009 the WLCG organization [1] adopts the HEP-SPEC06 benchmark suite (HS06) to describe experiment requirements, lab commitments, existing compute resources and specifications to procure new hardware [2]. This suite is based on a subset of the industrial-standard benchmark SPEC CPU 2006 [3].

In recent years discrepancies have been highlighted between HS06 scores and the average normalized time spent by the WLCG applications [4]. The discrepancies being more accentuated for LHCb and Alice applications whereas ATLAS and CMS applications still show good agreement with HS06, within 10% of accuracy. Most of the discrepancies are connected to the introduction of Intel Haswell [5] and Broadwell [6] CPU models and the different usage of the chipset instructions by the WLCG applications and the HS06 benchmark. Other discrepancies can be attributed to the LHC experiments adopting 64-bit compiled applications instead of the 32-bit compiled HS06 suite.

Contrary to the initial expectations, it was proven [7] that adopting SPEC CPU 2017 as new HEP benchmark does not bring much benefit with respect to HS06. In fact, the C++ benchmarks included in the new suite produce scores highly correlated with the HS06 scores. Therefore the adoption of SPEC CPU 2017 would not provide more information than HS06, at least for the range of applications and CPU models currently adopted in WLCG.

For these reasons the HEPiX Benchmark Working Group (BWG) is adopting a benchmark suite for the typical workloads of the LHC experiments. In the following section a quantitative

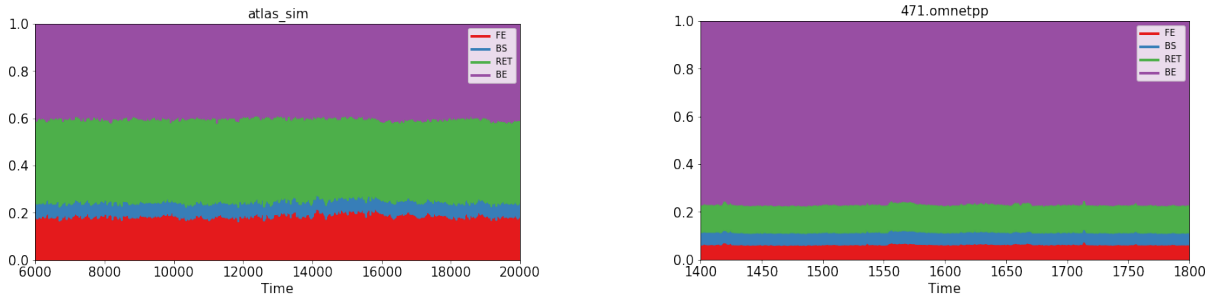
comparison of several benchmarks and HEP applications is reported. In section 3 the approach to build the HEP specific benchmark suite is reported.

2. Comparisons of benchmarks

HS06 consists of seven applications. Four applications are dominated by floating point operations (namd, dealII, soplex, povray) and the other three are dominated by integer operation (omnetpp, astar, xalancbmk). Similarly SPEC CPU 2017 has eight C++ applications. Some of those applications are a readaptation of SPEC CPU 2006.

The dissimilarities between HEP applications and SPEC CPU benchmarks can be highlighted by monitoring the usage of the CPU during the applications' runs to collect various low-level metrics related to the processor and the memory. In order to perform this study, an analysis toolkit of hardware and software performance counters called Trident [8] was used. Trident divides the instructions executed per CPU cycle into four classes: successful completion also named retiring (RET) class; stall in CPU Front End (FE) class or Back End (BE) class; rejections due to branch misprediction (BS) class.

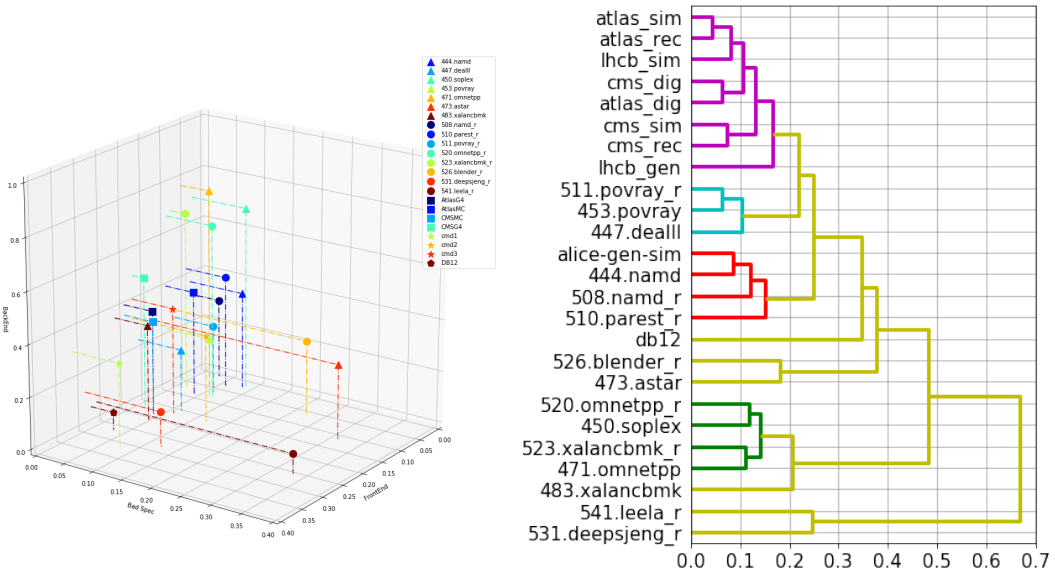
The CPU counters have been collected using Trident during the execution of the applications in the HS06 suite, SPEC CPU 2017 suite and the fast benchmark DB12 [9]. The same analysis has been performed on several experiment workloads, such as detector simulation, signal digitisation and event reconstruction. Applications from all four LHC experiments have been used. Figure 1 shows a comparison between ATLAS simulation and one of the HS06 applications (omnetpp) in terms of the percentage of CPU cycles spent in one of the four introduced categories (FE, BE, BS, RET). The different usage of the CPU spent by the two applications is evident.



(a) (b)

Figure 1: Percentage of CPU cycles spent in FE, BE, BS, RET, as a function of the application's running time, for Atlas simulation (a) and HS06's omnetpp application (b). The initialization and finalization phase of the execution have been excluded.

In total, 25 applications have been studied with Trident. For each of them, the time series of the extracted metrics have been analysed. After removal of the initialization and finalization phases, the metrics' evolution during the sequence of processed events have been summarised by averaging on the runtime window. For the CPU counters this approach gives three average values of the FE, BE, BS, that are representative of each studied application. The similarity between two applications in terms of those metrics has been defined as the L_1 distance of the two points representing the given applications in the 3D metric space (FE, BE, BS). The L_1 Euclidean distance was used in order to magnify the discrepancies between two points in each metric coordinate. The applications have been then clustered using hierarchical clustering [14]. The distance between the nearest applications in the metric space, as well as the distances between agglomerative clusters are summarised in the dendrogram tree of Figure 2b.



(a) (b)

Figure 2: 3D location of every studied benchmark in the metric space (FE, BE, BS) (a) and dendrogram of the applications' distance in this space (b). Different colors have been used for the clusters in which the distance is shorter than 18%.

Most of the HEP applications have an average distance between them smaller than 0.18. This means that the average CPU cycles spent in FE, BS, BE differ for not more than 18% of the processing time. On the contrary, the HS06 and SPEC CPU 2017 applications are distributed in other subspaces, reaching a discrepancy up to the order of 60%, as in the case of the deepsjeng_r application. To be also noticed that the similar applications in HS06 and SPEC CPU 2017 differ in the usage of the CPU within 15% (namd, omnetpp, xalancbmk). The only exception about the similarity of HEP applications is the Alice generation and simulation, that in average differs for more than 20% from all the other HEP applications. This can be explained considering that the Alice simulation is based on Geant3 [10] that has a completely different code base and programming language respect to the simulation framework Geant4 [11] used by the other experiments.

3. HEP benchmark suite

A suite based on workloads that HEP experiments run can be an alternative to industrial-standard benchmarks and can benefit of the adoption by the experiments of modern software development techniques such as container technologies as well as continuous integration and development methodologies. These techniques simplify the ability to package, distribute and maintain a field-specific benchmark suite, implementing build, test, packaging phases in pipelines.

As for any other benchmark suite, there are essential requirements that the HEP benchmark suite must satisfy. It must be reproducible, therefore the configuration of each application needs to be well defined, so that the application can process the same fixed sequence of events, with the same software version and calibration data. Moreover, it must be portable to different platforms. In order for it to be also usable by vendors and sites not having external connectivity, assumptions such as remote access of software and data must be dropped. Finally, the HEP benchmark suite has an open source license.

To satisfy those requirements the HEPiX BWG has decided to adopt the approach of building standalone containers encapsulating all and only the components needed to run the benchmark (fig. 3). These components fall in three categories: software, configuration files and input data. Software framework and algorithms are the same running in the typical WLCG production jobs. The configuration file and input data define the algorithms' sequence and the events' complexity in order to have a benchmark representative of the most common event patterns in the production jobs. That representation is directly connected to the LHC luminosity and energy that affect the amount of tracks to be simulated or reconstructed per bunch crossing.

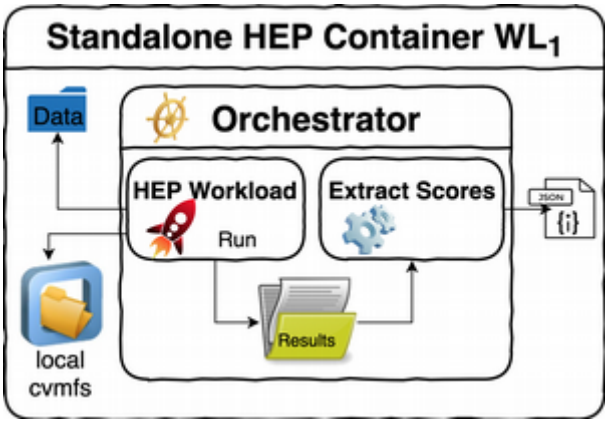


Figure 3: Standalone HEP Container

An orchestrator script manages the environment configuration, the start of the HEP application, the error handling and the results' parsing. The orchestrator exposes few tuning parameters via CLI, and produces a benchmark report compliant with the defined data types. The report is formatted as a JSON document for easy insertion in monitoring services. The running mode of the container includes the possibility of starting a configurable number of parallel and independent copies (processes) of the same application. The default benchmark running-mode targets all cores of a machine under test, and makes sure that resources are not over-committed: therefore the number of parallel processes is determined by the number of available cores and the number of threads each process will spawn.

In the process of building a standalone container the requirement on the container image size drives another technical choice on the software installation. The size of the container is determined by the input data size and the amount of code stored in the container. Given that, not all the algorithms and libraries need to be executed during the benchmark run thus there is no need of installing the full software stack. On the other hand, the input data size depends on the single event size and on the number of events to be processed in order to get a representative benchmark. The number of events is being carefully studied for each application in order to reduce the statistical fluctuation of the measurement.

To install only the software that the benchmark application effectively runs, and rely on what distributed on WLCG sites via CVMFS [12], a procedure has been developed that is based on the CVMFS Trace and Export [13] utilities to export applications software from CVMFS to a local folder inside a container. This procedure requires a first run of the application with access to the real CVMFS mount point, in order to trace the accessed software files. Therefore the Export utility copies the traced files to a local archive, that can be included in a docker image at build time. This building procedure simplifies the amount of information requested to the LHC Collaborations in order to be able to run a given application of their framework. No need to have RPMs distributed and installed, and the advantage of limiting the software to what will

effectively run.

Another requirement is to provide stable procedures to build and distribute the benchmark suite. This has been realized adopting the GitLab Continuous Integration (CI) [15] for the automation of the container build and validation, and the GitLab Container Registry [16] for the container distribution. Two container solutions are offered so far, Docker and Singularity.

4. Conclusion

The dissimilarities between HS06 and HEP applications have been highlighted using CPU counters. At the same time, commonalities between HEP applications are also shown. This evidence motivates the investigation of a new benchmark approach for HEP, that focuses on the adoption of applications specific to the field itself. A model based on standalone containers has been developed for software distribution. At the time of writing, generation, simulation and reconstruction workloads from the four LHC experiments have been already distributed via containers.

References

[1] Welcome to the Worldwide LHC Computing Grid, `http://wlcg.web.cern.ch/`

[2] Michelotto M et al. *J. Phys. Conf. Ser. 2010* **219** 052009

[3] Henning, John L 2006 *SPEC CPU2006 Benchmark Descriptions, SIGARCH Comput. Archit. News, September*

[4] P. Charpentier, J. Phys. Conf Ser. 2017 **898** 082011

[5] Hammarlund P *Haswell: The fourth-generation Intel core processor* IEEE Micro, Mar./Apr. 2014 10.1109/MM.2014.10 **34** no. 2, pp 6–20

[6] M Kumashikar, S Bendi, S Nimmagadda, A Deka, A Agarwal *4nm Broadwell Xeon processor family: Design methodologies andoptimizations* inProc. IEEE Asian Solid-State Circuits Conf., 2017 pp 17–20

[7] Alef M et al. *Next Generation of HEP CPU Benchmarks, CHEP 2018 proceedings (to be published)* `https://indico.cern.ch/event/587955/contributions/2937888`

[8] Muralidharan S, Smith D *Trident: An Automated System Tool for Collecting and Analyzing Performance Counters*

[9] Dirac Benchmark 2012 `gitlab.cern.ch/mcnab/dirac-benchmark/tree/master`

[10] Brun R, Bruyant F, Maire M, McPherson A C and Zanarini P,*Geant3* CERN-DD-EE-84-1

[11] Cosmo G and the Geant4 Collaboration *Geant4 - Towards major release 10* J. Phys. Conf Ser. 2014,**513** 022005

[12] CVMFS `https://cernvm.cern.ch/portal/filesystem`

[13] CVMFS shrinkwrap `https://github.com/nhazekam/doc-cvmfs/blob/shrinkwrap_tutorial/cpt-shrinkwrap.rst`

[14] Scipy: `https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html`

[15] GitLab Continuous Integration Delivery: `https://about.gitlab.com/product/continuous-integration/`

[16] GitLab Container Registry: `https://about.gitlab.com/2016/05/23/gitlab-container-registry/`

[17] HEPiX HEP reference workloads `https://gitlab.cern.ch/hep-benchmarks/hep-workloads/container_registry`