# HEPiX benchmarking solution for WLCG computing resources

*Miguel* F. Medeiros[1,*], *Manfred* Alef[2], *Luca* Atzori[1], *Jean-Michel* Barbet[3], *Ingvild* Brevik Høgstøyl[4], *Olga* Datskova[1], *Riccardo* De Maria[1], *Domenico* Giordano[1], *Maria* Girone[1], *Christopher* Hollowell[5], *Michele* Michelotto[6], *Andrea* Sciabà[1], *Tristan* Sullivan[7], *Randal* Sobie[7], *David* Southwick[1,8], and *Andrea* Valassi[1]
from HEPiX Benchmarking Working Group

[1]CERN, Geneva, Switzerland
[2]KIT, Karlsruhe, Germany
[3]Laboratoire SUBATECH, CNRS-IN2P3, Nantes, France
[4]Norwegian University of Science and Technology, Norway
[5]Brookhaven National Laboratory, USA
[6]INFN, Istituto Nazionale di Fisica Nucleare, Padova, Italy
[7]University of Victoria, Canada
[8]University of Iowa, USA

**Abstract.** The HEPiX Benchmarking Working Group has been developing a benchmark based on actual software workloads of the High Energy Physics community. This approach, based on container technologies, is designed to provide a benchmark that is better correlated with the actual throughput of the experiment production workloads. It also offers the possibility to separately explore and describe the independent architectural features of different computing resource types. This is very important in view of the growing heterogeneity of the HEP computing landscape, where the role of non-traditional computing resources such as HPCs and GPUs is expected to increase significantly.

## 1 Introduction

The HEP-SPEC06 (HS06) [1] benchmark, based on SPEC CPU 2006 [2], has been serving the Worldwide LHC Computing Grid (WLCG) [3] community for over 10 years. It has been the standard CPU benchmark solution and has been assisting on resource capacity planing decisions, such as resource acquisition, estimation and accounting. Computing sites have been using HS06 for their hardware procurement since it can be used as a reference to purchase CPU resources for the lowest financial cost, while taking other considerations such as electrical power efficiency measured in HS06 per Watt. It has served as a common metric to quantify the computing needs of the experiments and also the resources offered by each computing centre for a given year. It gives funding agencies and review boards a detailed accounting of computing resources [4].

The HS06 benchmark is no longer representative of High Energy Physics (HEP) workloads, especially when looking at technological evolution [5–7]. CPU architectures have

*e-mail: miguel.fontes.medeiros@cern.ch

evolved significantly, and hardware landscapes have become even more heterogeneous at HEP and High Performance Computing (HPC) centres. Graphical Processing Units (GPUs) for Machine Learning applications, hardware accelerators, FPGAs and non-x86 architectures such as ARM and Power are a reality today. The HS06 consistency was considered one of the main reasons for its success, but its also contributing to its downfall due to its inability to adapt to evolving infrastructure. There is a need for an evolved benchmark that can cope with the evolving and emerging technologies. A new benchmark would assist procurement teams on its decision-making concerning new equipment, especially with the tendency to move towards more energy efficient Data Centers as it is the case in the European Union [8].

The Benchmarking Working Group (BWG) [9] has been tackling this challenge and working towards an alternative benchmark solution based on HEP workloads. The BWG has created the HEP Benchmark project, which comprises several repositories from benchmarks, orchestration and analysis [10]. Its proof of concept was already demonstrated [5, 6]. Recently, a WLCG Task Force has been formed to evaluate the feasibility of replacing HS06 with the newly implemented HEPscore benchmark tool. It is the role of the Task Force to identify the combination (HEPscore2X) of HEP workloads that will define the final score.

This paper depicts the current state of the BWG activities. In the following section, we describe the requirements of a HEP benchmark as well as the general technical solution for packaging the workloads. In Section 2.1 we introduce the HEP workloads, which contains the applications provided by the HEP experiments. The HEPscore utility and HEPscore2X benchmark are presented in Section 2.2. In Section 3 we describe the benchmark orchestrator named HEP Benchmark Suite, which is used to run, collect, store and process benchmark data. Finally, Section 4 details the challenges of running on other platforms, such as GPUs, HPC facilities and ARM processors.

## 2 HEP Benchmarks

The BWG proposed that a new benchmark should be created from HEP workloads as an alternative to HS06. For the new benchmark to succeed, the results must be reproducible with acceptable tolerances. It should be easy to run and collect the results. Additionally, having a benchmark solution that adapts well to a cross-platform environment will be adopted by WLCG centres, and also HPC centres and other non-HEP organizations such as vendors and site procurement teams. Finally, a strict version control, tamper-proof repositories, checksums, code signatures and a clear license statement are some of the requirements that need to be enforced. The latter is of extreme importance since it would allow the adoption of this benchmark and avoid possible vendor-locks due to industry policy changes that affect software communities.

Using experiment workloads as a benchmark was an impossible task to solve in the past. Experiment code bases are complex and contain millions of lines of code [11] that rely on specific dependencies that differ from experiment to experiment. To run these workloads, these dependencies need to be shipped as well. Recent IT technological advances contributed to the feasibility of this project and two technologies play a key role: Linux Containers [12] and CernVM File System (CVMFS) [13]. Container technologies [14, 15] allow the isolation of the workload and CVMFS allows the software distribution from the HEP community.

### 2.1 HEP workloads

The HEP Workloads project started as a collection of workloads from the four LHC experiments. These workloads include a number of different steps including event data Generation

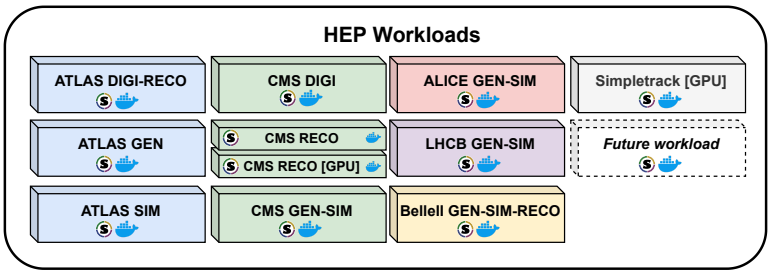


**Figure 1.** Organization of HEP Workloads project. Each experiment workload is a dedicated container.

(GEN), Simulation (SIM), Digitization (DIGI) and Reconstruction (RECO). The description of these workloads is beyond the scope of this paper. In Figure 1, the organization of the HEP workloads project is presented. Each HEP experiment maintains its specific workload package and provides an orchestrator script, which act as an entry point. The orchestrator prepares all runtime dependencies and executes the workload. Each workload returns a json file that contains the measurement of the number of processed events per second for a given period of time (wall-clock time). The *hep-workloads* repository [16] includes all the code to build each standalone container for each experiment. These containers are then built, tested, versioned and hosted under a Gitlab registry. Being a container, each workload can be run independently with any given container technology. Currently, the HEP workloads are based on the software used during the Run 2 of the LHC data taking period. Still, there are on-going works to have the new workloads integrated that will be based on the experiment code that will be used on Run3. The versioning scheme of HEP workloads allows future comparison studies between the different workloads (Run 2 and Run 3). Initially, the HEP Benchmarks project only encompassed LHC specific workloads, however, its modular design allows the integration of non-LHC experiments. Recently, a Belle II [17] workload was included, becoming the first integration of a non-LHC experiment workload.

The HEP Workloads project started with a focus on CPU benchmarking. With the rapid emergence of GPUs, the BWG has been following closely the developments of LHC experiment workloads that can be used to run on GPU architectures. However, GPUs are a recent architecture and not all LHC experiment workloads are ported to run natively on GPUs. Still, there are other workloads that are being investigated as well. For example, Simpletrack developed in the context of the SixTrack project [18] is particle tracking code used to compute trajectories of many charged particles circulating in the accelerator for many turns. The algorithm is fully parallel and well suited to run on GPU hardware. The code is written using OpenCL1.2 [19] with one thread associated to each particle. This allows to maximize performance in Nvidia, AMD GPU Intel and AMD CPU. The benchmark tracks a given number of particles $n$ for a given number of turns $t$ in a LHC models and return the $nt/sec$. This workload has been containerized and tested [20]. Another workload is the GPU version of MadGraph5_aMC@NLO [21]. It is currently being ported to GPU, but it will likely not be production ready in the near future. The workload is a particle collision event generator. Event generation is well-suited for running on GPUs as it is highly-parallelizable tasks that is usually computationally intensive depending on the number of particles involved. The workload has not yet been containerized or included in the benchmark, but there are plans to do so in the near future.

Each HEP workload container has a single workload that can be executed independently. Using these workloads combined, presents an opportunity to create a benchmark. In the next

section, we present HEPscore and how it uses the HEP workloads to create a benchmark and return a metric.

### 2.2 HEPscore

HEPscore (High Energy Physics score) is the utility designed to orchestrate the execution of multiple benchmark containers from the HEP Workloads project. Using the results from these containers, it calculates a single overall throughput benchmark score for a system. This score is normalized to the performance of the benchmark on a given reference machine, allowing for the simplified comparison of scores between systems.

The HEPscore application is highly configurable, and is passed a yaml configuration file specifying the parameters of the overall benchmark to execute. The yaml file includes the benchmark container repository, the list of individual benchmark containers to execute, and the parameters to pass to these benchmarks. HEPscore creates json/yaml summary output containing the overall score, as well as the results from all the executed benchmark containers, and various system metadata information including OS kernel version, container platform, execution time and other parameters. Both Singularity and Docker are supported for container execution.

HEPscore includes a default demonstration benchmark, HEPscore2X, that runs workloads from ATLAS, CMS, LHCb, and combines these results into a single score via geometric mean. This is the same numerical method used to aggregate the results of the dissimilar sub-benchmark scores in HS06. A host at CERN with a single Xeon E5-2630v3 @ 2.40 GHz (Haswell) CPU was used as the reference machine for HEPscore2X. Similar to HS06, each sub-benchmark in HEPscore2X is executed three times, and the median result taken in order to minimize the possibility of anomalous conditions affecting the score.

HEPscore version 1.0 includes several improvements over the prior beta releases. Perhaps the most substantial was the modularization of the core Python [22] code. This now allows the utility to be easily imported and called from other Python applications, which simplified the integration of HEPscore into version 2.0 of the HEP Benchmark Suite (described in Section 3). Other significant improvements include support for running benchmark containers from CVMFS and Singularity Hub registries (previously only Docker registries were supported), an option for per-container registries, support for GPU-based benchmark containers, and the implementation of a weighted geometric mean score aggregation method. The new HEPscore version 1.0 also provides an option to force Singularity user namespace-based container execution, which is particularly useful in nested Singularity environments. Finally, version 1.0 includes optional cleaning functionality for the benchmark container image cache (both Docker and Singularity) and scratch directories. This is an important feature as the container images and scratch directories created during benchmark execution can easily grow to the tens of gigabytes in size, and many computing systems have limited space available in local scratch areas and user home directories.

HEPscore's modular approach renders it agnostic to HEP Workloads, which allows their easy integration. This is of special importance with the emergence of new workloads that can leverage capabilities to benchmark GPUs. The BWG efforts on benchmarking GPUs are described in Section 4.

## 3 HEP Benchmark Suite

The HEP Benchmark Suite (Suite) is an orchestrator for benchmark workflows [23] and currently supports the following benchmarks: HS06, SPEC2017, HEPscore and DB12 [24]. The

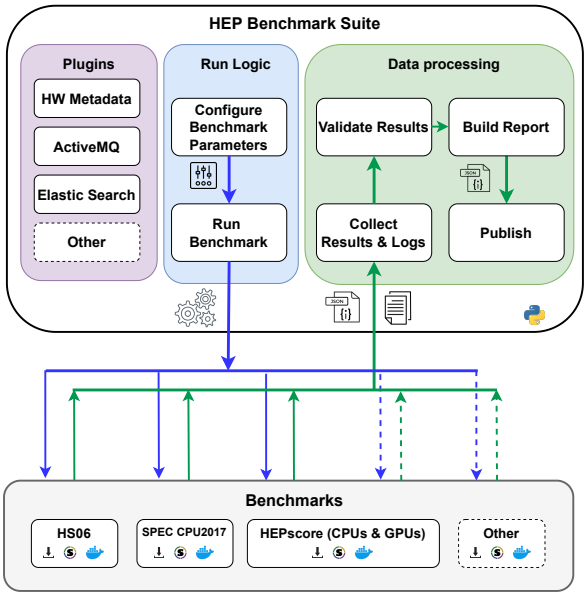


**Figure 2.** HEP Benchmark Suite version 2 workflow. It depicts the Suite main functional blocks together with the decoupled benchmarks.

Suite is not a benchmark and it does not redistribute HS06 and SPEC2017 in its package due to license constraints. Suite users will still need to provide their remote or local SPEC installations and licenses. Still, users can use the SPEC container image [25] provided by the BWG, which ensures standardization of compilation flags and reporting.

Recently, the BWG has focused its efforts on addressing some limitations not foreseen in the initial design of the Suite. The initial design of the Suite followed a tightly coupled approach. All benchmarks were built into a single Docker image which contained all benchmark layers required to run. Despite being a porting advantage (single container image with all benchmarks), this approach required in some cases the users to run containers with Docker-in-Docker or Singularity-in-Singularity. It required extended privileges which are disabled by default by computing centres as a security measure. Additionally, it was becoming difficult to maintain and the addition of new features and/or benchmark modifications would result in a full Suite image rebuild. Given these issues, the BWG decided to redesign the Suite.

In Figure 2, the workflow of Suite version 2 is presented. The Suite v2 follows a full decoupled approach with several advantages compared to the previous version (v1.8). It is a package fully written in Python 3. The easy installation, a rich standard library, and its availability in major Linux distributions were some of the arguments that led to the selection of this programming language. It is a lightweight package, relying only on a few dependencies to ensure its ease of portability. This allows the easy packaging (e.g. Python wheels), especially for locations that have restricted network connectivity. As depicted in Figure 2, the Suite v2 adopts a micro-service like approach and all benchmark containers are decoupled from the Suite. The decoupling allows the ease of maintainability. Benchmark contributors can propose modifications without breaking compatibility with other production benchmarks. To ensure the benchmark integration, each benchmark must provide an interface for the Suite and output a json report file containing all the results from that benchmark. The Suite is

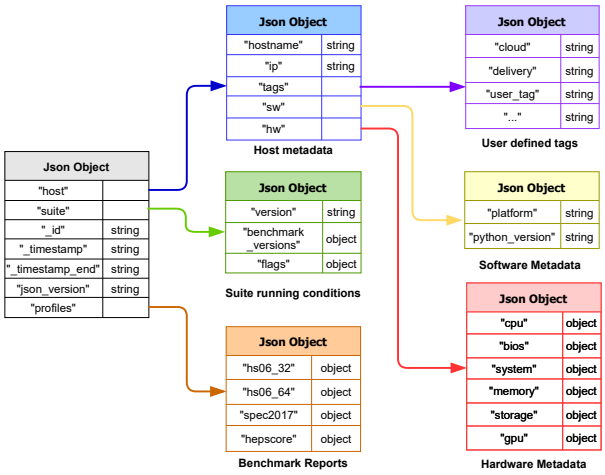


**Figure 3.** Suite version 2 json metadata. This metadata structure is modular to allow future expansion.

capable of using different container engines, which becomes agnostic to the type of benchmark. With this approach, the Suite is capable of running benchmarks even outside the HEP domain.

The Suite v2 has three main components: Plugins, Run Logic and Data processing. The plugins block contains all add-on features, which at the moment includes hardware metadata and communication interfaces. The Run Logic is where the user interacts with the Suite. All benchmarks are configured in a single yaml file named *benchmarks.yml* where options like the type of benchmark, versions to use, the number of cores and user tags are specified. For example, if a user wants to run HEPscore, then the Suite will install the desired version, run the benchmark and collect its report. Finally, the data processing is responsible for managing all the data flow such as benchmark data, logs and construct the final json report. All benchmark results together with its running conditions are reported in a single json document.

The BWG has improved the json metadata in Suite v2. In Figure 3, the improved json structure is presented. The host metadata comprises all the information to uniquely identify a host. It is composed by the user defined tags, software and hardware metadata. The user defined tags is the only json object that can be modified by the user. All remaining json objects are automatically populated by the Suite. This object allows users to add extra information that can be later used to identify the host. For example, it is the section used by procurement teams to assign a node to a given delivery number. The software and hardware json contains all relevant information that could influence a benchmark. With the current heterogeneous environment, it is becoming even more important that not only the final metric is obtained, but also that the running conditions are recorded. The Suite running conditions object is used to document configurations, together with the benchmark versions that were used. The benchmark profiles object contains all the benchmarks reports being each object a specific report. It is the responsibility of each benchmark maintainer to provide the Suite the benchmark information. The Suite has no control over the digested json objects from each benchmark. The modular json structure allows future expansion when needed. The *json_version* field can be used to mark specific json versions that differ on schema. The json file is saved locally at the end of benchmark completion. This format allows portability and different orchestration workflows. In Figure 4 the Suite benchmark data flow is presented,

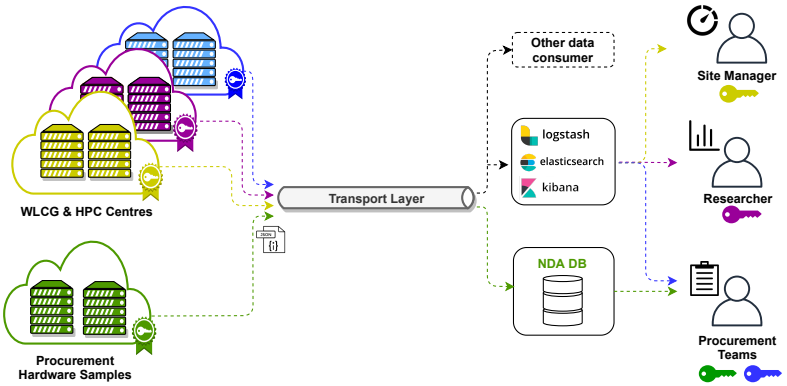


**Figure 4.** Suite v2 benchmark data workflow. It depicts the data flow from each computing site and how the interest parties can interact with this data.

depicting the main workflows of typically interested parties: Site Managers, Procurement Teams and Researchers. The json format allows the publishing to any data consumer. For example, site managers can benchmark their clusters and view later these results with the assistance of visualization frameworks. Additionally, having the benchmark results on these frameworks allows researchers to easily search, index, filter and visualize data an integrate it onto their data analytic solutions. Still, there are cases where the benchmark results need to be restricted and not publicly visible, especially when evaluating hardware samples. Procurement teams can publish benchmark results from evaluating samples on separate restrictive databases. This allows a better access control list enforcement given the user access nature.

## 4 Benchmark Heterogeneous environments

Computational resources are becoming heterogeneous and several programming frameworks [19, 26] are facilitating the development process to a cross-platform environment. In this Section, we describe the BWG efforts and challenges on benchmarking these platforms.

### 4.1 GPUs

GPUs can provide considerable parallel computational power which can be used to speed up applications. Having workloads that can leverage both system resources (CPU+GPU), will facilitate the BWG efforts on having a unique benchmark to evaluate a given system's performance. Still, challenges remain on selecting a metric for the benchmarked heterogeneous system. Profiling tools from GPU vendors can help evaluating GPU workloads [27] in order to characterize the workloads CPU and GPU contribution. In light of this expectation, the BWG has been prototyping a GPU benchmark. While containment technology for CPU related tasks has been around for some time, support for host hardware such as GPUs is a relatively new addition. Harnessing host peripherals requires additional configuration to a standard container call to inform the container the resources available outside. For GPUs, this requires that a driver be present and loaded into the host kernel, and exposed to the container runtime. Within the container, a library compatible with the exposed host hardware driver must be included, as there is no uniform way to expose host hardware libraries. This adds additional overhead to container development and image size. Additionally, this becomes

problematic when a host GPU architecture is unknown, resulting in many images maintained for a variety of accelerator architectures, or many libraries being included in a single image with the hopes of supporting more hardware.

Within this context, the BWG has produced a single "generic" GPU container image which contains a set of libraries for each of the top accelerators (Nvidia, AMD, and Intel). This allows the same image to be used to benchmark all popularly available GPUs by simply specifying the host hardware exposure command during container instantiation. With this approach only one container definition needs to be maintained, which eliminates sources of performance variation across vendors.

### 4.2 ARM architectures

After having dominated the market share in the smartphone sector, processors based on ARM technology [28] are gaining market shares in the server sector thanks to their requisites of high performance and low power consumption, allowing data centre optimization and reduction of total cost of ownership [29]. As an example, AWS is offering since May 2020 general purpose instances utilising the AWS-designed ARM-based chip Graviton2 using 64-bit ARM Neoverse cores. AWS claims a 40% improvement on cost-performance ratio respect to x86-64 architectures on similar size instances [30]. Considering this trend of adoption, and the increasing interest of the HEP community in ARM-based servers, the BWG has invested effort in extending the HEP benchmarks to ARM-based chips. In order to enable performance comparison w.r.t. x86 architectures, it is planned that both the new HEPscore benchmark and the current official benchmark HS06 shall run on ARM. The first achieved goal has been to provide ARM based container to run HS06 and SPEC CPU 2017. The toolkit is available and documented in the associated repository [25]. SPEC CPU 2017 natively supports ARM chips and the selection of appropriate compiler flags is the only requirement to switch from x86 to ARM architectures [31]. More complex has been the inclusion of support for HS06 on ARM chips. Being the SPEC CPU 2006 toolkit, on which HS06 is based, old, it does not support natively ARM processors, and the SPEC organisation does not help being SPEC CPU 2006 retired. We have solved the issue building the toolkit for ARM, after having patched some of the original files to cope with unsupported bug fixes. The procedure to follow in order to apply the patch to a site-owned SPEC CPU 2006 distribution is provided at [32]. At the time of writing a small number of ARM processors (ThunderX2, AWS Graviton2) have been already profiled with HS06 and SPEC CPU 2017.

### 4.3 HPC

The target hardware for benchmarking during the HS06 era has largely been nodes which are owned or administered by the WLCG centres. This has allowed a common set of assumptions for permissions, hardware, job submission, and accounting. In contrast, HPC sites are generally much more restrictive and heterogeneous. For security reasons, they are adverse to any elevated permissions or processes, and may not permit external network connectivity from compute nodes. They may contain a wide variety of hardware and network topologies, which are generally inflexible. They use their own job submission, scheduling, and monitoring tools, which must be adapted to. This change in assumptions requires all activities to operate unprivileged. Jobs must be modified or wrapped into a compliant format for the site's specific scheduler. The wide variety of hardware and related software libraries provided by the site make compilation and comparison of the resulting performance a potentially challenging task. Thankfully, this task may be simplified by the use of recent container technologies commonly available on HPC sites. In the case of restrictive node connectivity, additional steps

may be required for data ingress and egress through permitted endpoints. In contrast with the almost exclusive Intel-x86 compute environment at WLCG sites for the past decades, HPC sites offer a diverse environment of instruction sets and compute accelerators.

The HPC challenge was one of the motivators for the HEP Benchmark Suite redesign on version 2.0 which enabled an array of modular workloads to run across hundreds of nodes on a HPC site. The support for unprivileged execution, as required on HPC, has been added in the form of unprivileged Docker and corresponding Singularity images. Execution has been greatly simplified due to the leveraging of Python's modular packaging system. This enables simple integration with any manner of HPC scheduling services (e.g SLURM [33]). This has enabled the collection of over 2000 benchmark results across 418 unique nodes representing over 122,000 cores. Deployment across 200 simultaneous nodes has been tested without issue. For further detail, see the proceedings on HPC Exploitation in this issue [34].

## 5 Conclusion

Current benchmarking techniques used by the HEP community are becoming outdated in the context of recent technological evolutions. The HEP Benchmark project addresses this issue by proposing the creation of a new benchmark based on HEP workloads. The HEPscore utility and demonstrator benchmark have successfully shown that it is possible to create a reproducible/consistent CPU benchmark using containers from this project. The HEP Benchmark Suite has proved itself a great addition in orchestrating benchmarking workflow. Additionally, it adapts well to HPC centres and other heterogeneous environments, and allows modular future expansion. It is flexible enough to allow the usage outside of the HEP community. Currently, a WLCG Task Force was formed to evaluate the potential of replacing HS06 with a variant of HEPscore. The BWG has the infrastructure in place to accommodate the modular addition of future workloads. It is becoming clear that the selection and adoption of the new benchmark is becoming less a technical problem but rather a policy and accounting decision. Still, there are technical challenges being faced. The packaging of GPU workloads into smaller container images and the creation of multi-architecture container images for all the workloads are some of the examples.

## References


[1] M. Michelotto, et. al., Journal of Physics: Conference Series **219**, 052009 (2010)

[2] J.L. Henning, SIGARCH Comput. Archit. News **34**, 1–17 (2006)

[3] *Worldwide LHC Computing Grid*, https://wlcg.web.cern.ch/, accessed: 2021-02-16

[4] P. Sinervo, *Computing Resources Scrutiny Group Report - Plenary RRB 51st Meeting October 2020*, `https://indico.cern.ch/event/957354/contributions/4023648/`

[5] A. Valassi, et al., EPJ Web of Conferences **245**, 07035 (2020)

[6] D. Giordano, M. Alef, M. Michelotto, EPJ Web of Conferences **214**, 08011 (2019)

[7] P. Charpentier, *Benchmarking worker nodes using LHCb productions and comparing with HEPSpec06* (2017), `10.1088/1742-6596/898/8/082011`

[8] C. Nieuweling, *Code of Conduct for Energy Efficiency in Data Centres*, `https://ec.europa.eu/jrc/en/energy-efficiency/code-conduct/datacentres`, accessed: 2021-02-16

[9] *Benchmarking Working Group*, `https://w3.hepix.org/benchmarking.html`, accessed: 2021-02-16

[10] *HEP-Benchmarks*, https://gitlab.cern.ch/hep-benchmarks, accessed: 2021-02-16

[11] The HEP Software Foundation, et al., Computing and Software for Big Science **3**, 7 (2019)

[12] *Red hat - what's a Linux container?*, https://www.redhat.com/en/topics/containers/whats-a-linux-container, accessed: 2021-02-16

[13] J. Blomer, et al., Journal of Physics: Conference Series **898**, 062031 (2017)

[14] *What is a Container? | App Containerization | Docker*, https://www.docker.com/resources/what-container, accessed: 2021-02-16

[15] *Singularity*, https://sylabs.io/singularity/, accessed: 2021-02-16

[16] *HEP-Workloads*, https://gitlab.cern.ch/hep-benchmarks/hep-workloads, accessed: 2021-02-17

[17] E. Kou, et al., Progress of Theoretical and Experimental Physics **2019**, 123C01 (2019)

[18] *Sixtrack project*, https://cern.ch/sixtrack (2021), accessed: 2021-02-10

[19] *Opencl*, https://www.khronos.org/ (2013), accessed: 2021-02-13

[20] *HEP-Workloads-GPU*, https://gitlab.cern.ch/hep-benchmarks/hep-workloads-gpu, accessed: 2021-02-17

[21] J. Alwall, et al., Journal of High Energy Physics **2014** (2014)

[22] *Python Programming Language*, https://www.python.org/, accessed: 2021-02-16

[23] *HEP Benchmark Suite project*, https://gitlab.cern.ch/hep-benchmarks/hep-benchmark-suite, accessed: 2021-02-16

[24] *DB12: Dirac benchmark 2012*, gitlab.cern.ch/mcnab/dirac-benchmark/tree/master, accessed: 2021-02-25

[25] *hep-spec-package*, https://gitlab.cern.ch/hep-benchmarks/hep-spec/ (2021), accessed: 2021-02-10

[26] *Intel oneAPI: A Unified X-Architecture Programming Model*, https://www.intel.com/content/www/us/en/develop/tools/oneapi.html, accessed: 2021-02-13

[27] *NVIDIA Developer Tools Overview*, https://developer.nvidia.com/tools-overview, accessed: 2021-02-16

[28] *Arm*, https://www.arm.com/ (2021), accessed: 2021-02-10

[29] *ARM-powered Data Centers Optimized for Cost and Efficiency*, https://www.arm.com/blogs/blueprint/optimizing-data-center (2021), accessed: 2021-02-10

[30] *AWS ARM*, https://aws.amazon.com/blogs/aws/new-m6g-ec2-instances-powered-by-arm-based-aws-graviton2 (2021), accessed: 2021-02-10

[31] *spec2017-arm*, https://gitlab.cern.ch/hep-benchmarks/hep-spec/-/tree/master/scripts/spec2017 (2021), accessed: 2021-02-10

[32] *HS06-ARM-patch*, https://gitlab.cern.ch/hep-benchmarks/hep-spec/-/tree/master/patch_SPEC2006 (2021), accessed: 2021-02-10

[33] *SLURM*, https://slurm.schedmd.com (2021), accessed: 2021-02-10

[34] D. Southwick, et al., *Exploitation of HPC resources for data intensive sciences* (2021), submitted to CHEP2021 conference