International Journal of HIGH PERFORMANCE COMPUTING APPLICATIONS

Special Issue Paper

An HPC benchmark survey and taxonomy for characterization

The International Journal of High Performance Computing Applications
2026, Vol. 40(1) 42–51

DOI: 10.1177/10943420251351424
journals.sagepub.com/home/hpc
Sage

Andreas Herten[1], **Olga Pearce**[2,3] and **Filipe S. M. Guimarães**[1]

Abstract

The field of High-Performance Computing (HPC) is defined by providing computing devices with highest performance for a variety of demanding scientific users. The tight co-design relationship between HPC providers and users propels the field forward, paired with technological improvements, achieving continuously higher performance and resource utilization. A key device for system architects, architecture researchers, and scientific users are benchmarks, allowing for well-defined assessment of hardware, software, and algorithms. Many benchmarks exist in the community, from individual niche benchmarks testing specific features, to large-scale benchmark suites for whole procurements. We survey the available HPC benchmarks, summarizing them in table form with key details and concise categorization, also through an interactive website. For categorization, we present a *benchmark taxonomy* for well-defined characterization of benchmarks.

Keywords

HPC, Performance, benchmarking, taxonomy

1. Introduction

High-Performance Computing (HPC) is, by definition, aiming for excellent performance of the executed workloads. State-of-the-art hardware is deployed for scientists, engineers, and other researchers with ever-increasing performance capabilities. At the same time, these users solve computational challenges of ever-increasing sophistication, driving the demand for HPC systems.

While some expert users might understand the performance characteristics and hardware-software interplay of their applications, that is not generally the case for most users, system administration, HPC engineers, and support staff. Hence, dedicated applications with well-defined workloads are utilized to assess the performance of systems. These benchmarks have a long tradition in the HPC field, with the High-Performance LINPACK (HPL, Dongarra (1992)), used to rank supercomputers in the Top500[1], arguably the most prominent. But many benchmarks exist, focusing on diverse aspects of hardware and software of HPC installations: Individual niche benchmarks test dedicated hardware characteristics, other more integrated programs test interplay of different components, and some benchmarks entirely focus on software components. The choice is plentiful, but keeping track is hard.

1.1. Profile of benchmarks

For good reason, benchmarks are one of the key elements in the HPC researcher/engineer toolbox. Their benefit has many layers:

1.1.1. Clarity. Through the well-defined setup, workloads, and execution instruction, benchmarks allow objective, repeatable, and transparent performance measurements; the key is a clear – and ideally simple – metric.

1.1.2. Comparability. Analysis of various hardware installations with identical or similar benchmark execution makes the installations comparable and assessable by means of the benchmark's metric.

[1]Jülich Supercomputing Centre, Forschungszentrum Jülich, Jülich, Germany
[2]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA, USA
[3]Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA

Corresponding author:
Olga Pearce, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 7000 East Ave, Livermore, CA 94550-5507, USA.
Email: olga@llnl.gov

1.1.3. Durability. Through well-designed benchmarks, an accessible and continuous assessment of systems is enabled, allowing for tracking historical data and understanding technological improvements and trends.

1.1.4. Advancement. Modeling the complex interplay of hardware and software, benchmarks enable focused hardware research and application development towards ecosystem advancement; especially in relation to the theoretical capabilities of hardware (*peak performance*). A benchmark results database can create a competitive drive for system improvement.

1.1.5. Decisiveness. Ultimately, analyzing robust benchmarks across different hardware installations over time allows informed, objective decisions about system investment. Benchmarks are key in modern system procurements, where execution performance counts more than the hardware peak performance.

1.1.6. Validation. Well-defined benchmarks allow tracking resource utilization and performance regressions on in-production systems; for new systems, well-known benchmarks can be used to validate the system for production; they can also be used for stress-testing systems and support reliability of the system.

The benefit of individual benchmarks is amplified when they are collected into benchmark suites. Various workloads are combined to conduct holistic system characterization, usually with normalized, comparable metrics.

Benchmarks not only have value for HPC researchers and engineers, but also for HPC users. By having a well-defined version of their HPC application, users may track performance regressions in their program and understand performance limiters. By supplying benchmarks to the HPC community, users have a direct impact on system design and procurement decisions. In turn, seeing benchmark results for HPC systems, users have comparable baselines and can gain confidence in the capabilities of the system.

Of course, benchmarks have many limitations. Comparability crossing hardware generations and vendors is a hard task, as highly-optimized HPC applications optimize for specific hardware, which is a priori not directly transferable. A certain benchmark metric may be valuable, but might not convey general information about a system – especially synthetic benchmarks tend to measure specific aspects of a system, which have limited real-world applicability for more involved applications. Finally, creating a robust, repeatable, portable, versatile, stable, and clear benchmark is a very involved task, which requires significant effort.

1.2. Contribution

Because of their importance, the HPC field created a vast set of different benchmarks over the years. They range from simple synthetic benchmarks, over mini-apps of scientific applications, to full-blown large-scale applications with possibly large input files. Some benchmarks are collected into benchmark suites, typically created for system procurements, to replicate a desired measurement and workload mix.

Some benchmarks are well-known in the field – like HPL or STREAM (McCalpin, 1995) – while others are only known to a small subset. The benchmarks/suites may be published only on websites accompanying a procurement, or are hosted in a GitHub repository attached to a journal publication; finding them can be challenging.

To improve their visibility, findability, and, ultimately, benefit for the field, we contribute in this work a **collection of benchmarks and benchmark suites**. In the survey, we present each benchmark and suite with metadata, including license, URL for download, reference, notes, and categorization. The survey consists of a summarizing overview table, added at the end of this article, a supplemental table with all details and categories available alongside the article online, a dynamic website available online, and an evaluation of the collected material. For categorization, we develop a **Benchmark Taxonomy**, allowing for well-defined characterization of each benchmark and quick assessment of suitability. While taking great care to capture most of the benchmarks and suites of the field, it is likely that individual, potentially more niche benchmarks are missing. To that extent, we publish the benchmark survey in raw form[2] and the taxonomy keywords on GitHub as open source software/data for future extension. The interactive online version of the table is available at fzj-jsc.github.io/benchmark-survey/, the raw benchmark list is available at github.com/FZJ-JSC/benchmark-survey/, and the taxonomy definition at github.com/LLNL/benchpark/blob/develop/taxonomy.yaml.

1.3. Structure

The rest of the paper is structured as follows. In section 2 we concisely assess the state-of-the-field and discuss related work. In section 3, we present the Benchmark Taxonomy. Finally, in section 4 we present the benchmarks in table form. Some observations and evaluations are discussed in section 5. In section 6, we conclude our paper.

2. Related work

Efforts to publicly post performance-focused benchmark results relating to deployed HPC systems range from the long-running Top500 and aligned Green500[3] and IO500[4], to sub-discipline benchmarking such as Machine Learning Commons (MLCommons, Mattson et al., 2020; ML Commons, 2023) and HPL-MxP[5]. Notable is also https://OpenBenchmarking.org, which openly collects results from

the Phoronix Test Suite[6] and schema-compliant further benchmarks; the focus is on end-user devices.

There are numerous efforts to make benchmarking easier, by encoding build/run/evaluation rules for entire suites of benchmarks; examples include Pavilion (LANL Pre-Team, 2019), Reframe (Karakasis et al., 2020), JUBE (Breuer et al., 2022), Ramble (Jacobsen and Bird, 2023), and Benchpark (Pearce et al., 2023). While they showcase somewhat different approaches, the proliferation of such efforts underscores the vital importance – and difficulty – of benchmarking.

There have of course been attempts at surveying the HPC benchmarks – *meta-benchmarking*, in a sense. Here we list just a few of the recent ones. A survey of convergence of big data, HPC, and ML systems has been done by Ihde et al. (2022) and cites 25 benchmarks/suites (our paper covers the HPC-specific ones from this list). In similar spirit list Thiyagalingam et al. (2022) a number of scientific ML benchmarks and present the SciMLBench framework. Some works argue that although we already have too many benchmarks in the ML space (Zhang et al., 2019) (citing 54 ML benchmarking papers), we still need more - while needing convergence. This underscores the need for the community awareness of existing benchmarking work, better benchmark characterization, and, hopefully, collaboration to both improve the quality of benchmarks, and understand their applicability.

3. Taxonomy

Benchmarks come in different flavors with different execution profiles, focus points, dependencies, and other properties. While comparing well-known benchmarks colloquially is easy, a thorough, structured comparison is more involved – especially for more niche and specialized benchmarks.

In our survey, we identify the different flavors and develop a systematic approach for characterization of benchmarks, considering a variety of different aspects of the individual benchmarks. This **Benchmark Taxonomy** consists of a number of *categories* under which benchmarks can be sorted and which group different *entries* of the similar kind. A combination of multiple category-entry pairs allows for fine-grained description of each benchmark.

Categories range from the application domain, where a benchmark originates from (like astrophysics), over the method employed for execution (like FFT), and the employed programming language (like Fortran), to detailed aspects like memory access characteristics (like regular access).

The categories are presented in Figure 1 with all currently collected entries (normalized to lower-case). The list of possible entries is likely not complete for every possible workload; it rather represents the result of our collection, augmented with other obvious entries.

To allow for extension of the taxonomy and support future work building up on it, the raw data is available in concise YAML form on GitHub. The schema employed uses the categories for keys, and individual entries available as a list for the values; for example `communication: [mpi, nccl]`.

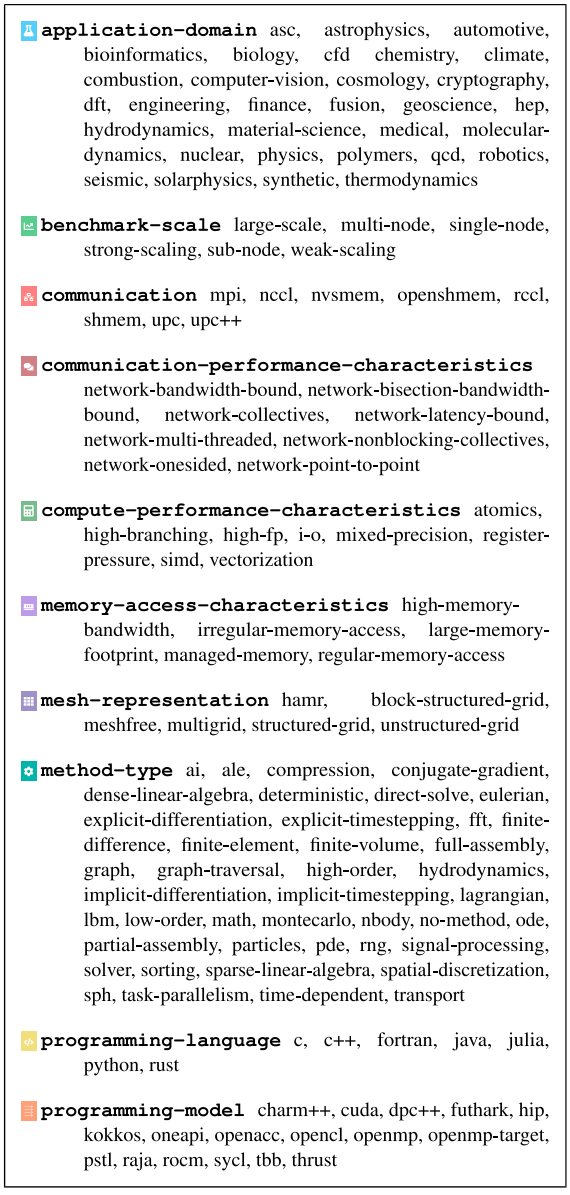

Figure 1. Taxonomy overview with top-level *categories* (printed in bold with added symbol) and each multiple *entries*.

Each category has an associated symbol and color, which is used in *tag* form within the full version of the table (Table A1) and in the interactive online version. An example tag for a benchmark utilizing MPI communication is mpimpi. In the machine-readable raw form within the benchmark list, the category-entry-combination is key-value-combined with a colon: `programming-model:cuda` (for cudacuda).

Table 1. Overview version of benchmark survey.

*(Suite-)*Name	Description, included benchmarks (if suite)
ATS-5	Suite for the procurement of the fifth version of the advanced technology system by LANL/NNSA with representative workloads of the centers. Contained benchmarks: Branson, AMG2023, Parthenon-VIBE, MLMD, UMT, MiniEM, SPARTA, LAMMPS
benchpark	The suite comes with re-implementations of well-known benchmarks/benchmark suites in a variety of programming models Contained benchmarks: AD, AMG2023, BabelStream, branson, GENESIS, GPCNet, GROMACS, HPCG, HPL, IOR, kripke, laghos, LAMMPS, MDTest, MiniEM, OSU micro-benchmarks, Parthenon-VIBE, phloem, quicksilver, QWS, RAJAPerf, remhos, SMB, STREAM
CORAL-2	The suite comes with re-implementations of well-known benchmarks/benchmark suites in a variety of programming models Contained benchmarks: HACC, NEKBONE, QMCPACK, LAMMPS, AMG, kripke, quicksilver, PENNANT, BDAS, DLS, CMB, STREAM, stride, MLDL, IOR-MDTest-simul-FTree, CLOMP, pynamic, RAJAPerf, E3SM, VPIC, laghos, ParallelIntegerSort, havoq
HeCBench	A collection of simple heterogeneous computing benchmarks, aligned in categories Contained benchmarks: Automotive benchmarks, bandwidth benchmarks, bioinformatics benchmarks, computer vision and image processing, cryptography, data compression and reduction, data encoding, decoding, or verification, finance, geoscience, graph and tree, language and kernel features, machine learning, math, random number generation, search, signal processing, simulation, sorting, robotics
HPC Challenge	A benchmark suite of 7 benchmarks that measures a range of memory access patterns. Contained benchmarks: HPL, DGEMM, stream, PTRANS, RandomAccess, FFT, b_eff
JUPITER benchmark suite	The JUPITER benchmark suite is used for procurement of the JUPITER exascale system and consists of application and synthetic benchmarks. The vast majority of the benchmarks focus on GPU execution. The application benchmarks come in base category (usually executing on 8 nodes, each 4 GPUs) and high-scaling category (using between 500 and 650 nodes, each 4 GPUs). Contained benchmarks: Amber, arbor, chroma LQCD, GROMACS, ICON, JUQCS, nekRS, ParFlow, PIConGPU, quantum ESPRESSO, SOMA, MMoCLIP, Megatron-LM, ResNet, DynQCD, NAStJA, Graph500, HPCG, HPL, IOR, LinkTest, OSU micro-benchmarks, STREAM, STREAM (GPU)
NERSC-10	This suite represents scientific workflows: Simulation of complex scientific problems at high degrees of parallelism, large-scale analysis of experimental or observational data, machine learning, and the data-flow and control-flow needed to couple these activities in productive and efficient workflows. Contained benchmarks: Optical properties of materials workflow, materials by design workflow, metagenome annotation workflow, lattice QCD workflow, DeepCAM AI workflow, TOAST3 cosmic microwave background workflow
OLCF-6	Suite for procurement of the next ORNL supercomputer (post-exascale), developed to capture the programming models, programming languages, numerical motifs, fields of science, and other modalities of investigation expected to make up the bulk of the usage upon deployment. Contained benchmarks: LAMMPS, M-PSDNS, MILC, QMCPACK, FORGE, workflow
RAJAPerf	The RAJA performance suite is designed to explore performance of loop-based computational kernels of the sort found in HPC applications. In particular, it is used to compare runtime performance of kernels implemented using RAJA, and the same kernels implemented using standard or vendor-supported parallel programming models directly (such as CUDA and ROCm). Contained benchmarks: STREAM, PolyBench, LCALS, halo communication, basic patterns, application kernels, algorithms
Rodinia	No updates since a long time; customized BSD-3-clause license Contained benchmarks: Leukocyte, heart wall, MUMmerGPU, CFD solver, LU decomposition, HotSpot, back propagation, needleman-wunsch, kmeans, breadth-first search, SRAD, streamcluster, particle filter, PathFinder, Gaussian elimination, k-nearest neighbors, LavaMD2, myocyte, B + tree, GPUDWT, hybrid sort, Hotspot3D, huffman
SPEC ACCEL	Commercial suite with distinct execution instructions, focusing on accelerators. Current version: v1.3. Individual benchmarks are combined here per category for brevity. Contained benchmarks: SPEC ACCEL_OCL, SPEC ACCEL_OACC, SPEC ACCEL_OMP
SPECaccel 2023	Commercial suite with distinct execution instructions, focusing on accelerators. Update of the preceding SPEC ACCEL suite, extending selected benchmarks. Contained benchmarks: 403.stencil, 404.lbm, 450.md, 452.ep, 453.clvrleaf, 455.seismic, 456.spF, 457.spC, 459.miniGhost, 460.ilbdc, 463.swim, 470.bt

(continued)

Table 1. (continued)

*(Suite-)*Name	Description, included benchmarks (if suite)
SPEChpc	SPEChpc collects its benchmarks into suites of different workload sizes: tiny, small, medium, large. Each size targets different number of tasks and higher memory usage. Contained benchmarks: LBM D2Q37, SOMA, tealeaf, cloverleaf, minisweep, POT3D, SPH-EXA, HPGMG-FV, miniWeather
UEABS	The unified european application benchmark suite is a set of 13 application codes maintained by PRACE. The last release was in 2022, and it is probably not maintained anymore. Contained benchmarks: Alya, Code_Saturne, CP2K, GADGET, GPAW, GROMACS, NAMD, NEMO, PFARM, QCD, quantum ESPRESSO, SPECFEM3D, TensorFlow
HPL	High performance linpack
HPCG	High performance conjugate gradients is a complement to linpack (HPL)
PolyBench	A benchmark suite of 30 numerical computations with static control flow, extracted from operations in various application domains. Last commit in 2018.
STREAM	Simple synthetic benchmark program that measures sustainable memory bandwidth (in MB/s) and the corresponding computation rate for simple vector kernels.
BabelStream	STREAM in many models for many devices; also available: Julia, rust, scala, java
PTRANS	Matrix transpose
RandomAccess	GUPS (giga updates per second)
FFT	1d discrete fourier transforms
b_Eff	MPI benchmark for measuring effective accumulated bandwidth in a network; several message sizes, communication patterns and methods used.
LCALS	Livermore compiler analysis loop suite, a collection of loop kernels based, in part, on historical livermore loops benchmarks. Original website currently not reachable.
Graph500	Linpack for graph problems. Breadth first search (BFS).
Rodinia (julia)	Julia-port of the rodinia benchmarks, with single and multi-threaded implementations and Julia + CUDA; has the rodinia license (customized BSD-3-clause)
Rodinia (DPC++)	DPC++-translation of rodinia benchmarks (SYCL tag added for visibility)
Rodinia (SYCL)	SYCL implementations of rodinia benchmarks, currently deprecated (integrated into/maintained through HeCBench)
OSU Micro-benchmarks	A vast collection of network-related micro-benchmarks.
SPEC MPI 2007	MPI-targeted benchmark suite. Last update: 2009 (v2.0). It features the following benchmarks: 104.milc, 107.leslie3d, 113.GemsFDTD, 115.fds4, 121.pop2, 122.tachyon, 125.RAxML, 126.lammps, 127.wrf2, 128.GAPgeofem, 129.tera_tf, 130.socorro, 132.zeusmp2, 137.lu, 142.dmilc, 143.dleslie, 145.lGemsFDTD, 147.l2wrf2
GPCNet	Global performance and congestion network tests.
SPEC OMP 2012	OpenMP-focused benchmark, successor of SPEC OMP 2001 benchmark. Includes the following benchmarks: 350.md, 351.bwaves, 352.nab, 357.bt331, 358.botsalgn, 359.botsspar, 360.ilbdc, 362.fma3d, 363.swim, 367.imagick, 370.mgrid331, 371.applu331, 372.smithwa, 376.kdtree
benchFFT	FFTW's FFT benchmark
MLPerf HPC	MLPerf training: HPC collects 4 HPC-related benchmarks: Climate segmentation (DeepCAM), cosmology parameter prediction (CosmoFlow), quantum molecular modeling (DimeNet++), protein folding (AlphaFold2).
IO500	The IO500 benchmark captures user-experienced I/O performance with a variety of workloads.
Fiber miniapp	Suite of miniapps with: CCS QCD, FFVC, NICAM-DC, mVMC, NGS analyzer, MODYLAS, NTChem, FFB. Different licenses. Last update: 2015.
Lulesh	Suite of proxy apps for 3D Lagrangian hydrodynamics on unstructured mesh. Current version is 2, but no update has happened in quite some time.

4. Benchmark survey

Table 1 presents a shortened overview of the surveyed benchmarks[7], augmenting the vast complete overview Table A1 available as supplemental material. Over 180 individual benchmarks and 13 benchmark suites were collected. The main future-proof point of access is the online version of the table at fzj-jsc.github.io/benchmark-survey/ with features for dynamic filtering and sorting, as well as automatically incorporating updates from the community.

The shortened overview in Table 1 first lists suites of benchmarks with commentary about the suite and contained benchmarks, and then free-standing benchmarks with respective notes. The complete table Table A1 and the online version

contain more details. Each table entry consists of the most relevant information for a benchmark/suite:

4.1. Name

The self-given name of each benchmark/suite is used for identification; when ambivalent, the more commonly used term was taken

4.2. Taxonomy

Tags Building up on the Benchmark Taxonomy of section 3, each entry is characterized by a set of taxonomy tags. For suites, tags common to (nearly) all contained benchmarks are promoted to top-level tags of the suite itself and are not separately displayed for each individual benchmark.

4.3. License

As the scope of applicability of a benchmark is closely related to its license, an effort is made to collect the information here. For brevity, each license is shown in symbol form using the following keys: None No license; Free Free (no (clear) license specified, but freely available); Prop Proprietary/Custom; MIT MIT; BSD-2/3/4-Clause; GPL-2.0/3.0; LGPL-2.1 (logo abbreviates 2.1 to 2); Apache-2.0; MPL-2.0; CC-BY-4.0.

4.4. Reference

For easy access, a URL for each benchmark is provided in *link* form.

4.5. Notes

In the notes, comments and details about the benchmarks/suites are added, as well as names of benchmarks of suites-within-suites (HeCBench, SPEC ACCEL) and older suites (SPEC MPI, SPEC OMP). If available, a scientific publication is added as reference at the end of the notes. But only a minority of benchmarks expose readily available publications.

The completeness of this metadata is highly dependent on the offered information at the source of the data; i.e. only if clear description is provided, it could be added as well-formed metadata. Especially, the taxonomy tags referring to workload profiles are highly reliant on the available data. In addition, the entries are augmented with the authors' knowledge of the workload, building up on their experiences in the field. The authors gladly welcome contributions by the community on GitHub to further extend this list.

The collected data is available in machine-readable, raw YAML form on the accompanying GitHub repository (Herten et al., 2025). This allows for easy extension; and also for future developments building up on the data. The YAML schema uses the following keys for each entry: `name`, `tags`, `license`, `url`, `ref`, `notes`, `benchmarks`; Listing 1 gives an example. Except for `tags`, which expects a list of taxonomy tags (see Figure 1), every key is accompanied by a single value. The `benchmarks` key is only present for benchmark suites, and under it the individual benchmarks are listed with the same schema. Freestanding benchmarks outside of suites are summarized under a top-level key `benchmarks`; again, for each entry, the default schema follows.

```yaml
benchmarks:
    hpl:
        name: HPL
        url: https://www.netlib.org/benchmark/hpl/
        license: BSD-4-Clause
        ref: 10.1145/141868.141871
        tags: [application-domain:synthetic,
        ↪  programming-language:c,
        ↪  math-libraries:blas,
        ↪  method-type:dense-linear-algebra]
        note: "High Performance Linpack"
```

Listing 1: Example benchmark definition in YAML schema.

5. Evaluation of survey

5.1. Overview and highlights

In the complete survey, 13 benchmark suites were recorded. They can be roughly sorted into three groups. References are omitted here, as they can be neatly found in Table A1 and online.

5.1.1. System procurements. These suites were created for large system procurements and used for evaluation of large installations. They are arguable the most thoroughly documented benchmarks surveyed, as they document the needs and requirements of whole user communities to system integrators. Nearly all benchmarks in this list are GPU-accelerated and combine application benchmarks (or mini-apps thereof) with synthetic benchmarks.

OLCF-6 is the suite for the next-generation supercomputer to be hosted at Oak Ridge National Lab, the RFP closed in autumn 2024. ATS-5 is the suite for the next supercomputer at Los Alamos National Lab, due to be installed in 2026/2027. NERSC-10 is the benchmark suite for the successor of the Perlmutter system at NERSC, to be

deployed in 2026. The JUPITER Benchmark Suite Herten et al. (2024) was used for the procurement of JUPITER, currently being built at Jülich Supercomputing Center. The CORAL-2 benchmark suite was used for the acquisitions of Frontier and El Capitan, hosted at Oak Ridge National Lab and Lawrence Livermore National Lab, respectively; the systems are already operational.

5.1.2. Research and community collections. Grown out of endeavors of research institutes and partly supported by the community, a number of benchmark suites has emerged which focus on certain niches. For example RAJAPerf Pearce et al. (2025), which started as a suite to verify and showcase the RAJA programming model Beckingsale et al. (2019a), but now incorporates other parallel programming models in comparative fashion; the focus is on typical parallel patterns and simple algorithms. RAJAPerf includes other suites, for example LCALS and Polybench. HeCBench is similar and includes a vast amount of simple computational benchmarks in a variety of different GPU programming models. In the table, we only list the categories, as the suite contains more than 400 individual programs. Another well-known collection of benchmarks for heterogeneous machines is the Rodinia benchmark suite. It is frequently used in the community for many performance investigations. The future of Rodinia is unsure, as the benchmark currently appears unmaintained. The case is similar for UEABS (*Unified European Applications Benchmark Suite*) by the European PRACE project. The suite captures many well-known applications with detailed execution instructions and input data, but appears not maintained anymore. The HPC Challenge benchmark suite combines many synthetic benchmarks, and also shares results on their website; but the project seems currently abandoned.

5.1.3. (Semi-)commercial offerings. The Standard Performance Evaluation Corporation, SPEC, provides different benchmark suites targeted for a variety of use-cases. Relevant for HPC are SPEChpc, offering a number of benchmarks in different workload sizes, SPEC ACCEL, last updated 2019, and focusing on different GPU benchmarks (with OpenCL, OpenMP, and OpenACC), and SPECaccel 2023, updating the previous suite and using OpenMP- and OpenACC-accelerated applications for benchmarking. Further benchmarks relevant to HPC exist. The benchmark setups are closed source and can be commercially acquired.

The choice for individual benchmarks is plentiful – be it freestanding or integrated into a suite. Famous and well-used benchmarks are HPL, which is particular compute-intensive and used for the Top500 ranking, HPCG, a conjugate gradient benchmark with data access patterns seen in applications, STREAM, which measures memory bandwidth with simple data movement routines, BabelStream, a STREAM implementation for a variety of GPUs, the Ohio State University Micro-Benchmarks, which test communication libraries (for example MPI), MLPerf, one of the few AI/ML benchmark in the suite, or IOR, a tool to determine I/O performance also used for the IO500.

Well-curated benchmarks are scarce, and the silent sunsetting of established suites is a loss for the field. It appears challenging to keep pace with the fast-moving update cycle in HPC, resulting in some benchmarks discontinued to various degrees. The authors hope that with this work, not only an overview can be gotten, but also awareness raised for the trove of choice currently existing.

5.2. Statistical evaluation

Albeit identification of characterizing aspects was at times challenging, many taxonomy tags could be attached to the benchmarks. The approach of using a YAML scheme for the taxonomy and the survey allows for some first, imperfect[8] evaluation of the state-of-the-practice. For the following numbers, taxonomy tags common to benchmarks in a suite have been, temporarily, added again to the benchmarks themselves. Categories can, of course, be present multiple times for individual benchmarks, for example if a benchmark is available in both OpenMP and CUDA.

Over 400 times a programming model was identified, with OpenMP being the most prominent parallelization choice (158 entries), followed by CUDA (94 times); OpenMP Target (48), OpenACC (43), and HIP (31) follow. Not claiming perfect representativeness, the results still appear to reflect the current trends in the field: CPU-focused benchmarks almost exclusively utilize OpenMP for CPU-parallelization and GPU benchmarks mostly use CUDA for GPU-parallelization. A programming languages could be identified 222 times, with C and C++ equally the most prominent ones (80 entries each). Fortran has 50 entries, Python only 10. Despite Fortran's importance in HPC, the vast majority of benchmarks use C/C++. Surprising is the little count of Python benchmarks, being the driving force in many sub-domains of HPC. The most used benchmark scale is *single-node* with 88 entries; *multi-node* follows with 52 entries. A focus on intra-node tests can be seen, removing network effects and related implementation/evaluation complications. Application information is available 122 times, with *synthetic* being the front-runner (18), *physics* (17), *molecular dynamics* (MD, 17), and *computational fluid* dynamics (CFD) and *climate* following (both 12). A slight focus appears to be on synthetic benchmarks, determining intricate and specific hardware features inspired by actual application workloads. Physics, MD, CFD, climate research are traditional HPC use-cases, which are expectedly represented in the survey. 92 benchmarks were identified to use MPI for

communication, with much distance to NCCL (5). MPI is the de-facto standard for communication in HPC, which can clearly be seen.

The taxonomy categories referring to more involved benchmark details are harder to identify and require thorough description or deeper knowledge of the benchmarks. Characteristics of memory access could be identified 48 times, of communication performance 40 times, and of compute performance 28 times.

6. Conclusions

Benchmarks are essential tools in the HPC field, enabling well-defined and comparable assessments of HPC systems. They characterize hardware features and software capabilities in an objective manner, support the advancement of the field, and are key for investment decisions.

In the presented work, we collect available HPC benchmarks and benchmark suites and survey them in a concise and comparable manner. An overview with strongly reduced level of detail is available in Table 1. The much longer full survey, including characterization tags, is available as supplemental material in Table A1 and online (interactive). 13 benchmark suites and over 180 benchmarks could be collected with detailed metadata, like associated license, references, notes, and characterization. For the latter, we create a Benchmark Taxonomy (Figure 1) to describe different aspects of the individual benchmarks and suites and enable easy comparison. The raw data of the survey and the taxonomy are available for further extension and collaboration as open source software on GitHub, feeding directly into the interactive website.

The benchmark suites are either created for large system procurement (like Frontier, El Capitan, or JUPITER), or come from parts of the community (like RAJAPerf). Without claiming representativeness, we attempt a first evaluation of the collected taxonomy tags and find, for example, OpenMP appears to be by far the most-used programming model for parallelization on the CPU. On the GPU, CUDA is the main model.

In the future, we expect to further extend the Benchmark Taxonomy, adding, for example, further application domains, methods, or libraries; we also consider extending it by status-related information to cover functionality and topicality of benchmarks, since some defunct benchmarks were collected. Although being thorough in our review of available benchmarks, we are certain that some benchmarks escaped our attention. We expect to extend the benchmark survey in the future to benchmarks not yet included in the current data; in hopes of help by the community online.

Acknowledgements

The authors would like to thank Jens Domke for his thoughts relating to significant benchmarks for this work.

ORCID iDs

Andreas Herten https://orcid.org/0000-0002-7150-2505
Olga Pearce https://orcid.org/0000-0002-1904-9627
Filipe S. M. Guimarães https://orcid.org/0000-0002-5618-6727

Funding

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the LLNL-LDRD Program under Project No. 24-SI-005 (LLNL-JRNL-2001672).

Declaration of conflicting interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Supplemental Material

Supplemental material for this article is available online.

Notes

1. https://www.top500.org.
2. The data is available in simple, machine-readable YAML form, for which we developed a fitting schema. The YAML data is the single source of truth of which the PDF and HTML tables are generated from; all required scripts are also part of the repository.
3. https://top500.org/lists/green500.
4. https://io500.org/.
5. https://hpl-mxp.org.
6. https://www.phoronix-test-suite.com/.
7. Due to its size, the table is inserted at the end of this article.
8. Of course, this is not the core scope of the paper. We do not claim completeness in the added taxonomy tags, but merely survey. Also, we do not resolve into "suites-of-suites" (HeCBench).

References

Beckingsale DA, Scogland TR, Burmark J, et al. (2019a) RAJA: portable performance for large-scale scientific applications. In: 2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC), Denver, CO, USA, 22 November 2019.

Beckingsale DA, Scogland TR, Burmark J, et al. (2019b) Raja: portable performance for large-scale scientific applications. In: 2019 IEEE/ACM International Workshop on

Performance, Portability and Productivity in HPC (P3HPC), Denver, CO, USA, 22 November 2019.

Berendsen H, van der Spoel D and van Drunen R (1995) Gromacs: a message-passing parallel molecular dynamics implementation. *Computer Physics Communications* 91(1–3): 43–56.

Breuer T, Lührs S, Smolenko A, et al. (2022) Jube. (Version 2.5.1); 2.5.1. https://juser.fz-juelich.de/record/917408

Che S, Boyer M, Meng J, et al. (2009) Rodinia: a benchmark suite for heterogeneous computing. In: 2009 IEEE International Symposium on Workload Characterization (IISWC), Austin, TX, USA, 4–6 October 2009.

Chunduri S, Groves T, Mendygral P, et al. (2019) Gpcnet: designing a benchmark suite for inducing and measuring contention in hpc networks. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. New York, NY: ACM.

Deakin T, Price J, Martineau M, et al. (2018) Evaluating attainable memory bandwidth of parallel programming models via babelstream. *International Journal of Computational Science and Engineering* 17(3): 247.

Dongarra JJ (1992) Performance of various computers using standard linear equations software. *ACM SIGARCH Computer Architecture News* 20(3): 22–44.

Dongarra J, Heroux MA and Luszczek P (2015) High-performance conjugate-gradient benchmark: a new metric for ranking high-performance computing systems. *The International Journal of High Performance Computing Applications* 30(1): 3–10.

Herten A, Achilles S, Alvarez D, et al. (2024) Application-driven exascale: the JUPITER benchmark suite. In: SC24: International Conference for High Performance Computing, Networking, Storage and Analysis, Los Alamitos, CA, USA, 17–22 November 2024.

Herten A, Pearce O and Guimaraes F (2025) Benchmarks survey GitHub. https://github.com/FZJ-JSC/benchmark-survey

Hornung R, Keasler J and Gokhale M (2011) Hydrodynamics challenge problem. https://doi.org/10.2172/1117905

Ihde N, Marten P, Eleliemy A, et al. (2022) A survey of big data, high performance computing, and machine learning benchmarks. In: Nambiar R and Poess M (eds) *Performance Evaluation and Benchmarking*. Cham: Springer International Publishing, 98–118.

Jacobsen D and Bird B (2023) Ramble: a flexible, extensible, and composable experimentation framework. *HPC Tests Workshop at the ACM/IEEE International Conference on High Performance Computing, Network, Storage, and Analysis (SC|23)*. Denver, CO, USA: ACM.

Jin Z and Vetter JS (2023) A benchmark suite for improving performance portability of the sycl programming model. *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. New York, NY: IEEE.

Karakasis V, Manitaras T, Rusu VH, et al. (2020) Enabling continuous testing of hpc systems using reframe. In: Juckeland G and Chandrasekaran S (eds) *Tools and Techniques for High Performance Computing*. Cham: Springer International Publishing, 49–68.

Komatitsch D and Tromp J (2002) Spectral-element simulations of global seismic wave propagation-i. Validation. *Geophysical Journal International* 149(2): 390–412.

LANL Pre-Team (2019) Pavilion framework. https://github.com/hpc/pavilion2

Luszczek PR, Bailey DH, Dongarra JJ, et al. (2006) The hpc challenge (Hpcc) benchmark suite. *Proceedings of the 2006 ACM/IEEE conference on Supercomputing* 213: 1.

Madec G, Bell M, Blaker A, et al. (2023) Nemo ocean engine reference manual. https://zenodo.org/record/8167700

Mattson P, Cheng C, Diamos G, Coleman C, et al. (2020) MLPerf training benchmark. *Proceedings of Machine Learning and Systems* 2: 336–349.

McCalpin JD (1995) Memory bandwidth and machine balance in current high performance computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA)*. New York, NY: IEEE, 19–25.

ML Commons (2023) MLPerf. https://mlcommons.org/en/

Pearce O, Scott A, Becker G, et al. (2023) Towards collaborative continuous benchmarking for HPC. *Proceedings of the SC '23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis, SC-W '23*. New York, NY, USA: Association for Computing Machinery, 627–635.

Pearce O, Burmark J, Hornung R, et al. (2025) Raja performance suite: performance portability analysis with caliper and thicket. *Proceedings of the SC '24 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*. New York, NY: IEEE Press, 1206–1218.

Phillips JC, Hardy DJ, Maia JDC, et al. (2020) Scalable molecular dynamics on cpu and gpu architectures with namd. *The Journal of Chemical Physics* 153(4): 044130.

Plimpton S, Brightwell R, Vaughan C, et al. (2006) A simple synchronous distributed-memory algorithm for the hpcc randomaccess benchmark. In: 2006 IEEE International Conference on Cluster Computing, Barcelona, Spain, 25–28 September 2006.

Springel V, Pakmor R, Zier O, et al. (2021) Simulating cosmic structure formation with the gadget-4 code. *Monthly Notices of the Royal Astronomical Society* 506(2): 2871–2949.

Thiyagalingam J, Shankar M, Fox G, et al. (2022) Scientific machine learning benchmarks. *Nature Reviews Physics* 4(6): 413–420.

Zhang Q, Zha L, Lin J, et al. (2019) A survey on deep learning benchmarks: do we still need new ones? In: Zheng C and Zhan J (eds) *Benchmarking, Measuring, and Optimizing*. Cham: Springer International Publishing, 36–49.

Author biographies

Dr. Andreas Herten is a researcher at Jülich Supercomputing Centre (JSC) of Forschungszentrum Jülich

(FZJ). He is a joint lead of the Novel System Architecture Design division, in which he heads the Accelerating Devices Lab. His research focuses on enabling applications for new hardware, especially GPUs and other accelerators, collaborating closely with vendors and users. He enjoys investigating programming models for performance and reproducible benchmarking of hardware-software ecosystems. Recently, he was responsible for the benchmarks used for assessment of JUPITER, the first European exascale supercomputer, where the thought for this survey was ignited. Andreas is a physicist by training, receiving his doctorate from Ruhr-Uni Bochum/FZJ (institute IKP) for research on usage of highly-parallel algorithms on GPUs in hadron physics. He joined JSC in 2015 to continue enabling applications on GPUs. Since then, he is involved in a variety of third-party funded projects for HPC enablement and performance improvement, and participated in multiple HPC system procurements.

Dr. Olga Pearce is a computer scientist in the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory. She created Benchpark, an open collaborative repository for reproducible specifications of HPC benchmarks and cross-site benchmarking environments, and Thicket, an open source toolkit for Exploratory Data Analysis (EDA) of parallel performance data. Olga leads benchmarking for Advanced Technology Systems, the Performance Analysis and Visualization for Exascale Project, and Performance Modeling in the Fractale SI. Her research interests include HPC architectures and simulations, parallel algorithms and programing models, system software, performance analysis and optimization. Olga has been at LLNL since 2007. She received the NSF graduate fellowship in 2006, Lawrence Scholar Fellowship in 2009, and joined CASC as technical staff in 2014. Olga received the LLNL Deputy Director's Science & Technology award (2015), and LLNL awards for developing the RAJA performance portability model (2018), porting and optimization of codes on LLNL's first accelerated supercomputer (2019), developing GPU capabilities of the Next-Gen Multiphysics code (2021), response to the National Academies of Sciences RFI (2023), and acceptance of El Capitan (2022, 2024). Olga helped create the SC Student Cluster Challenge in 2008, started a joint appointment at Texas A&M University as the Associate Professor of Practice in the Computer Science and Engineering in 2021, and serves as a co-chair of the Salishan Conference on High-Speed Computing. Olga received her Ph.D. in Computer Science from Texas A&M University, and her B.S. in Computer Science and Mathematics (dual major) from Western Oregon University.

Dr. Filipe Guimarães is a Computational Physicist who earned his PhD in 2011. He joined FZJ in 2014 as a researcher in the PGI-1 institute, specializing in condensed matter physics. In 2020, Filipe transitioned to JSC, where he became a member of the support group of the Application Optimization lab. His work focuses on providing advanced technical support and optimizing computational workflows for scientific applications. Filipe is also one of the developers of LLview, a powerful tool for monitoring and visualizing high-performance computing resources.