The ESIF-HPC-2 Benchmark Suite

Preprint

Christopher H. Chang[1], Ilene L. Carpenter[2] and Wesley B. Jones[1]

[1] *National Renewable Energy Laboratory*
[2] *Hewlett Packard Enterprise*

Presented at the Principles and Practice of Parallel Programming (PPoPP) 2020
San Diego, California
February 22-26, 2020

NREL is a national laboratory of the U.S. Department of Energy
Office of Energy Efficiency & Renewable Energy
Operated by the Alliance for Sustainable Energy, LLC

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at www.nrel.gov/publications.

Contract No. DE-AC36-08GO28308

Conference Paper
NREL/CP-2C00-76098
February 2020

The ESIF-HPC-2 Benchmark Suite

Preprint

Christopher H. Chang[1], Ilene L. Carpenter[2] and Wesley B. Jones[1]

[1] *National Renewable Energy Laboratory*
[2] *Hewlett Packard Enterprise*

Suggested Citation
Chang, Christopher H., Ilene L. Carpenter, and Wesley B. Jones. 2020. *The ESIF-HPC-2 Benchmark Suite: Preprint*. Golden, CO: National Renewable Energy Laboratory. NREL/CP-2C00-76098. https://www.nrel.gov/docs/fy20osti/76098.pdf.

NREL is a national laboratory of the U.S. Department of Energy
Office of Energy Efficiency & Renewable Energy
Operated by the Alliance for Sustainable Energy, LLC

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at www.nrel.gov/publications.

Contract No. DE-AC36-08GO28308

Conference Paper
NREL/CP-2C00-76098
February 2020

National Renewable Energy Laboratory
15013 Denver West Parkway
Golden, CO 80401
303-275-3000 • www.nrel.gov

NOTICE

This work was authored in part by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by the U.S. Department of Energy Office of Energy Efficiency and Renewable Energy Wind Energy Technologies Office. The views expressed herein do not necessarily represent the views of the DOE or the U.S. Government.

Cover Photos by Dennis Schroeder: (clockwise, left to right) NREL 51934, NREL 45897, NREL 42160, NREL 45891, NREL 48097, NREL 46526.

The ESIF-HPC-2 Benchmark Suite

Christopher H. Chang
Computational Science Center
National Renewable Energy Laboratory
Golden, CO USA
ORCID 0000-0003-3800-6021

Ilene L. Carpenter
Hewlett Packard Enterprise
Arvada, CO USA
ilene.carpenter@hpe.com

Wesley B. Jones
Computational Science Center
National Renewable Energy Laboratory
Golden, CO USA
wesley.jones@nrel.gov

***Abstract*—We describe the development of the ESIF-HPC-2 benchmark suite, a collection of kernel and application benchmark codes for measuring computational and I/O performance from single nodes to full HPC systems that was used for acceptance testing in our recent HPC procurement. The configurations of the benchmarks used for our system is presented. We also describe a set of "dimensions" that can be used to classify benchmarks and assess coverage of a suite systematically. The collection is offered cost-free as a GitHub repository for general usage and further development.**

Keywords—benchmark, HPC, procurement,

I. INTRODUCTION

Benchmarking as an activity has the fundamental goal of establishing a performance measurement for the object of the activity in a standard and reproducible way, often for the purpose of ranking this object against others of similar kind. The "object" could be a process, or an artifact that executes a process, within a context of external factors. So, a business process with the associated business policies and strategies can be benchmarked, as in the Six Sigma approach to process optimization [1]. Within computing, the object is normally a collection of hardware able to carry out instructions comprising the benchmark. "Performance" can entail throughput, efficiency, completion rate, or any metric the optimization of which brings benefit to a stakeholder. Finally, the measurement of such performance must be quantitative and must establish a field on which different objects may be compared evenly, understanding that any such comparison has constraints ("apples-to-apples"). The benchmark must encapsulate a precise statement of such constraints to make comparisons clear and fair.

The rapid progress in processor performance, once delivered mainly through clock frequency updates, has slowed. The end of Dennard scaling has led to a diverse set of architectural improvements and much greater use of concurrency to deliver performance improvements. This new diversity makes performance measurement and benchmarking critical to understand how and whether these new processors will deliver productivity to users. Within the domain of scientific computing, the floating-point operation reigns supreme, and the standard by which large-scale HPC systems have been benchmarked has been the solution of dense systems of linear equations as embodied by the high-performance Linpack package [2]. Recently, other performance metrics have been targeted as well, leading to ranked lists such as the Graph500 for graph-based operations [3] and the Green500 for a combined maximization of computation and minimization of energy consumption [4].

A. Value of a Standard Site-Tailored Set

However, the value and generalizability of Linpack results alone has been questioned. Unsurprisingly, once Linpack became a community standard, it fell victim to Goodhart's law[1], and the relationship between increases in Linpack performance and increases in performance of general floating-point applications began to diverge. Another concern arises from the complexity of scientific software. Ultimately, the scientist cares about the application, not its underlying algorithms, and taken at that relatively high level of abstraction, no single algorithmic benchmark can reasonably be expected to correlate highly with every application of interest. In other words, every application has a unique pattern of performance bottlenecks, such that creating a useful benchmark depends not on measuring one aspect of a system, but enough aspects to capture the complexity of both the system and the set of applications it executes.

Furthermore, the set of applications in which a community is interested varies by site. At the National Renewable Energy Laboratory (NREL), our HPC systems host a diverse set of applications ranging over quantum chemistry and materials science, molecular dynamics, fluid dynamics, multiphysics, and complex energy systems optimizations. While diverse, these applications are still grounded in floating-point computation. However, they do not primarily solve dense linear systems as a bottleneck. Thus, there is perceived value in a collection of benchmark codes and run configurations ranging from kernels to full applications that reflect the specific needs of the energy research community at NREL.

B. ESIF-HPC-2

The following describes the process undertaken to create an initial benchmark suite for NREL's most recent large-scale HPC procurement, dubbed ESIF-HPC-2. This new system, named Eagle and released to general production early in 2019, is a follow-on to the Peregrine system that has served as NREL's and DOE/EERE's primary HPC resource [6]. The system was specified as predominantly x86_64-based, with node-local persistent storage, high-reliability shared filesystem storage, and large shared parallel filesystem storage. Subsets of nodes were also specified for visualization, data-intensive computing, and accelerated computing as limited testbeds for emerging workloads. The set of benchmarks chosen and the rationale for them is discussed, and a strategy for a public release of the current snapshot and future plans are considered.

[1] Usually expressed as "When a measure becomes a metric, it ceases to be a good measure." [5]

II. Methods

A. Purposes

Benchmarking activity can serve a number of purposes within a production environment, including

1. acting as a quantitative performance discriminator for choosing an optimal system;
2. enabling responsive vendor system design;
3. validating the delivered system;
4. ensuring burst reliability of system at speed (*e.g.*, stress testing);
5. continuous verification of the system in production;
6. providing information for detailed understanding of requirements to achieve performance; and,
7. setting expectations for future runs that cannot be done on a current system.

For the ESIF-HPC-2 procurement, points 1-3 were of primary concern. Of course, as the system moves online, points 4 and 5 will become crucial. Points 6 and 7 are indirect but important outcomes of benchmarking activity. In production, the hardware and software of an HPC system is generally viewed in an abstract and coarse-grained manner, in which one may not be able to exploit the full capability of hardware performance. The efforts of those who are charged with actively pursuing peak performance in a competitive environment can demonstrate what is needed to obtain such performance. Furthermore, any HPC procurement will involve acquiring a system that reflects the substantial pace of technological evolution of the marketplace. Without constant exposure to this evolution, one can become accustomed to one's present environment, such that it can be difficult to design projects and goals appropriate to the current state of technology. Benchmark results can serve to update that thinking, oftentimes dramatically given the period of system refreshes (in our experience, ~3-5 years) and the pace of hardware and software performance improvement.

B. Dimensions

As a component of initial discussions and in an attempt to provide some systematic structure for this activity, an initial set of dimensions was proposed. The idea behind this discussion was to think of a particular benchmark as residing in a space of features (Table 1). Thinking in this way would have the primary goals of preventing duplication and an over-abundance of benchmark choices in one region of this space, and making manifest neglected considerations. A more precise formulation might be in terms of subspaces and dimensions within them, with some breakdown as correlations exist between subspaces. For example, "scalable" parallel scope may often go hand-in-hand "network" as a hardware subsystem and "memory-to-memory" data transfer (*i.e.*, MPI communication as a key component of the benchmark), implying some non-orthogonality among these dimensions. This exercise is similar in spirit to the analysis expressed as "Ogres" [8] for big data benchmarking, with "facets" and "views" there replaced by "dimensions" and "subspaces" here.

It should be recognized that for some dimensions, there is a degree of judgment involved in assigning values. For example, "algorithm" in particular can include a well defined and well recognized computational process (*e.g.*, LU factorization), or a less defined but well recognized category (*e.g.*, dense linear algebra). One could conceivably extend this to complex collections that involves many different actual computational actions, but signify such a common process that they can be reliably grouped together as a unit (*e.g.*, computational fluid dynamics in terms of grid computation and sparse linear algebra, although the distinction between structured and unstructured grids can make such a grouping fuzzy if a benchmark uses one or the other, like Nalu). Primarily, these categories are simply examples of an increasing degree of abstraction as the benchmark progresses along the "software scope," and there is a degree of non-orthogonality to be expected between these two dimensions. Similarly, one can debate the degree of task coupling, and precisely what type of data transfer is being exercised. To a degree, the answers are a matter of intent and circumstance; nonetheless, we believe there is a value in having a space in which to place benchmarks, whether as points or figures of greater extent.

For the specific purposes of the ESIF-HPC-2 procurement, development focused on sampling along what might be considered traditional benchmarking dimensions of hardware subsystem, parallel scope, and data transfer, with some mind paid to software scope. These dimensions are natural first foci, in that they do not require detailed knowledge of implemented algorithms, can rely on a number of standard packages (*e.g.*, STREAM[9], Intel MPI Benchmarks), and provide a reasonable high-level snapshot of system performance for decisions on hardware acquisition. Sampling of other dimensions certainly occurred, but was incidental. Consideration of the other dimensions will be of interest in guiding future benchmark development, particularly for the other purposes to which benchmarking may be put.

TABLE I. A provisional set of dimensions or features, that define a benchmarking space.

Subspace	Type	Dimensions
Hardware subsystem	Categorical	processor, memory, storage, network
Parallel scope	Ordinal	serial, MT/MP[1] single node, multi-node, many-node/scalable
Software scope	Ordinal	kernel, mini-application, full application, workflow
Degree of task coupling	Ordinal	loose, medium, tight
Data transfer	Categorical	Cache-core, memory-core, LFS-memory, NFS-memory, PFS-memory, external-memory, memory-memory[2]
Performance type	Categorical	Maximum application, sustained
Algorithms	Categorical	See [7], [8]
Interactive productivity	Categorical	Job scheduler delay, GUI latency, framework support

[1] Abbrevations: MT/MP, multi-threaded or multi-process; LFS, local filesystem; NFS, networked serial filesystem; PFS, parallel filesystem; GUI, graphical user interface

[2] Here LFS, NFS, PFS, and external refer to filesystem types and localities to, e.g., DRAM. Memory-memory refers to transfer between separate address spaces through, e.g., MPI.

C. Identification

With a set of purposes in mind and an understanding that benchmarks should primarily focus on serial and parallel performance of key hardware, a set of benchmarks was identified. These mainly fell into two categories: community-standard kernel benchmarks, and application benchmarks [10] reflecting the bulk of our current system's node-hour usage. The former included STREAM, high-performance Linpack, Intel MPI Benchmarks, IOR, mdtest, and Bonnie++. For consideration of graphical processing unit (GPU) performance, we also included the SHOC benchmark [11]. The latter category of benchmarks reflected our internal workload analysis, which shows that most of NREL's computing has, and is projected to continue to, focused on (a) materials and chemical electronic structure, (b) large-scale computational fluid dynamics and fluid-structure interaction, and (c) classical atomistic molecular dynamics. The set of benchmarks included as part of the ESIF-HPC-2 suite is listed in Table 2.

Bonnie++: Although the question was raised in external review what Bonnie++ could add to results from properly configured IOR, we elected to include this I/O kernel in the suite for two reasons. First, it has in the past been a standard test that can easily test local filesystems on nodes and produce results directly comparable with past results. Second, we have observed in the past that Bonnie++ runs in an overall way that reflects realistic I/O workflows of computational scientists who are less HPC-aware, and exercises resources in a corresponding way that other benchmarks like IOR may not. For example, testing of network-attached filesystems will probably reflect CPU and network rather than disk limits per se [12], and user-level calls may well engage interactions with default buffer sizes [13]; nevertheless, this will reflect what our average user will see. By default, Bonnie++ will detect DRAM size of the machine on which it runs, and should use a dataset large enough to prevent client-side caching. Runs were single-node, and were requested to use a single core, half the available cores on the node, and all cores on the node with synchronization in the case of multiple threads (-y), and to skip per-character tests (-f).

IOR: This benchmark (currently along with mdtest hosted at https://github.com/hpc/ior) was included as our primary parallel filesystem test. The I/O parameters in which we were most interested were read vs. write; sequential vs. random access, which was controlled through transfer size settings of 4 MB and 4 kB, respectively, as well as explicitly requesting random offsets for the latter (-z); and, performance range over a single process, half the available cores, and all available cores on the test system. Tests were configured only for the POSIX API, file-per-process access, a single segment, and a block size equal to the total file size, which was mandated simply to be a multiple of available DRAM to prevent caching [14]. This configuration leaves out certain differentiated cases (*e.g.*, MPI-IO on a shared file with certain access patterns), but given that a well established parallel filesystem technology was anticipated and that POSIX has shown similar performance to MPI-IO in certain circumstances [15], it was felt simplicity outweighed the additional overhead of completeness for our purposes. The chosen configuration was also not chosen to mirror a particular application I/O pattern, as the target system must serve a diverse set of current and future applications with indeterminate I/O patterns. Overall, our IOR benchmark was designed for easy interpretation and to stress the filesystem in ways compatible with our observed workload (which is predominantly high-throughput, small node-count jobs).

mdtest: To test metadata performance of the high-performance parallel filesystem, creating, statting, and removing files in three basic run configurations were requested.

1. 2^{20} (1,048,576) files in a single directory;
2. 2^{20} files in 2^{20} directories, with each directory held by only one MPI rank; and,
3. A single file split among N ranks

Four run configurations were requested for each test,

1. a single rank,
2. m_1 ranks on a single node yielding optimal performance,
3. m_n ranks over n nodes yielding optimal global performance for the system; and,
4. N×C ranks over all N nodes of a given type, with C hardware cores/node (*i.e.*, a fully packed system).

TABLE II. BENCHMARKS IN THE ESIF-HPC-2 SUITE AND PARAMETERS EXPLORED.

Benchmark	Parameter Summary
STREAM Triad	Threads = 1, numcores, maximum performance; Default memory and 60% of total on-node
HPL	2^N node runs up to ½ and all nodes; all runtime parameters tunable
SHOC Triad	1 or 2 GPUs; BusSpeed and Triad tests
Bonnie++	1, ½, all cores on node; 2×DRAM; local and NFS
IOR	1, maximum, all nodes; 1.5×DRAM; NFS; POSIX + MPI-IO; 1 file, 1 file/process
mdtest	1, all, and maximum performing # nodes; 1 or 2^{20} files, 1 file/directory or 1 directory; create/stat/remove
Intel MPI Benchmarks	$2^{\{1,6-10\}}$ nodes; 0, 64k, 512k, 4M messages; PingPong, SendRecv, Exchange, 0B Barrier, Uniband, Biband, Allreduce, Allgather, Alltoall
HPGMG-FV	$2^{\{6-10\}}$, N/4, N/2, N ranks; 27-unit box, 8 boxes/rank
Nalu	$2^{\{1,7-10\}}$ and N nodes; 256 mesh
VASP	$2^{\{4-8\}}$, 320 ranks; GW small unit-cell band structure, and Γ-only catalysis
LAMMPS	$2^{\{0,2,4,6,7\}}$ nodes, 35% salt solution, 7e5 – 5e7 atoms
Gaussian	1 & 2 nodes; Mn-aquo DFT single point, 175 e-/520 BF
HiBench	5 nodes; Hadoop & Spark, Wordcount, Sort, Bayes, K-means, DFS I/O Enhanced; "Gigantic" problem size

HPL and HPGMG-FV: Two additional benchmarks listed in Table 2 deserve further comment. It has been documented thoroughly that HPL, the primary determinant of Top 500 rankings, has become increasingly disconnected from the practical performance achieved on production workloads in HPC [16], [17]. HPL has been engineered to highlight the raw floating-point performance of highly parallel systems, but the kernel's computational intensity (Flop/byte) is high enough [16] that it fails to exercise other hardware subsystems that can bottleneck parallel codes on application problems, *e.g.*, memory and interconnect latency and bandwidth. For that reason, even though HPL was included to provide a "standard candle," we included the finite-volume formulation of High Performance Geometric Multigrid [17]. This benchmark provides a cleanly packaged body of code that exercises the hardware subsystems in a more balanced way [18], and directly reflective of computational patterns found in modern large-scale parallel algorithms. The finite volume implementation exercises the memory subsystem's bandwidth more so than the CPU floating-point units or caches.

Application Benchmarks: Gaussian, VASP, LAMMPS, and Nalu: At the time of the ESIF-HPC-2 procurement, the bulk of the application load on our first cluster, Peregrine [6], comprised physics codes including molecular and materials electronic structure, classical molecular dynamics, and computational fluid dynamics. The choices of Gaussian[19] and VASP[20]–[23] were based primarily on existing workflows on Peregrine. LAMMPS[24] had the benefits of a GPLv2 license, and a focus on performance at scale. Nalu is a CFD code likewise targeting massively parallel systems, and is representative of applications supporting important engineering work at NREL. The differing formulations of these benchmarks reflected some finer points of our intent. For Gaussian, the intent was to ensure that the program's shared and distributed memory runtime models were simply *supported*, and to *assess* performance relationships. For VASP on the other hand, the intent was to *optimize* performance by highlighting system design points against a critical workflow, and to assess problem-specific performance scaling against Peregrine. LAMMPS and Nalu had similar aims, but a focus on nodes rather than ranks forced a minimum degree of internode communication, and so highlighted interconnect performance as opposed to memory capacity (which the memory-constrained problem in bench1 of the VASP benchmark could emphasize).

HiBench: The ESIF-HPC-2 procurement activity recognized that the world, and the scientific and engineering activity within it, is becoming more data-intensive. In recognition, a part of the requested resources included nodes that could be deployed for "big data" types of work. To this end, the open-source HiBench [25] was included to ascertain the differential value of new hardware against our current environment. We were specifically interested in Hadoop and Spark performance on some basic patterns over large data sets, *e.g.*, binning (in HiBench: Wordcount), sorting (Sort), clustering (K-Means), and classification (Bayes), as well as a basic measurement of Hadoop I/O (enhanced DFSIO). Given the current and foreseen near-future workloads at NREL, we requested performance runs on a cluster of 5 data nodes and 1 name node, using HiBench's pre-defined "gigantic" set of data.

D. Development workflow

The Git revision control system was employed to enable simple content development and issue tracking. Content was based in an institutional GitHub, permitting private development and simple management of ancillary concerns such as access control. A Git Organization was created as a containing entity for the benchmarks, each of which was assigned its own repository. Each repository was made write-accessible to the Organization Owner roles, as well as the assigned developer. Development decisions were left largely to the developers themselves, beyond an understanding that the default markdown README.md file (present by default in every Git repo) would serve as the primary vehicle for instructions specific to each benchmark. Questions and issues were tracked through the standard GitHub mechanism, enabling provenance, locality (*i.e.*, a single place to find items), and persistence for future reference. Development via Git also permits facile ongoing development and public release.

E. Testing

The Git system provides a convenient means to establish parallel co-existing repository versions through its branching mechanism. By branching each repo to a "3PT" version for third-party testing, instructions could be customized for this purpose and preserving the same mechanism that final benchmark users would employ (*i.e.*, start at README.md) without requiring error-prone change and reversion.

Third-party testing was intended to ensure that a user reasonably familiar with and skilled in the use of the computing environment could (a) acquire the benchmark materials, (b) build the required binary components by following the instructions in README.md, (c) run the benchmark according to the instructions provided, and (d) interpret reporting instructions adequately to complete a subset of desired reporting requirements for each benchmark. To these ends, a set of testers mostly disjoint with the development team was chosen. In some cases, the test team was simply a permutation of the development team, but in any case such that no person who developed a benchmark was involved in testing. Issues were reported through the Git tracking system, and results were then checked into the 3PT repo branch by the third-party testers.

Additional input was provided by an external review panel comprised of established experts in leadership-class high-performance computing. This valuable process allows for integration of leading expertise in the field, a diversity of experience with procurement in multiple institutional settings, and up-to-the-minute knowledge of technology trends and status. In this particular scenario, external review served as both a sanity check and a credibility multiplier, adding to our confidence in the scope and coverage of the benchmarking process.

III. Results

A. Benchmark Collection

Categories. In terms of the dimensions outlined in Table 1 above and the assignments in Table 3, the initially released benchmark suite described here occupies a limited subspace of hardware subsystem, parallel scope, and software scope. This might be considered a fairly traditional choice for purposes of

(a) system procurement (b) for general-purpose HPC (c) within a limited technological scope (*e.g.*, where the available high-speed networking protocols have similar latency and bandwidth figures). Other dimensions, like task coupling or data transfer, might have greater predictive value for systems with more focused application domains or a broader palette of technology choices, or for other purposes like stress testing.

B. Definitions of Dimensions

Hardware subsystem. Each subsystem (processor, memory, storage, and network) contributes to a benchmark if it is at least likely to contribute substantively to the overall rate limitation. So, even though STREAM Triad involves floating-point operations, it is limited primarily by memory bandwidth. Although HPL is at least partially limited by interconnect [26], it predominantly reflects processing power (vide supra). "Memory" limitation can also include data movement between memory pools on a single server, *e.g.*, from host to accelerator. Application benchmarks are not so clearly limited by a single hardware factor, and the particular formulation of these benchmarks determines which factors were included. So, whereas the VASP tests as formulated exercise all the subsystems besides storage, Gaussian is only formulated to validate functionality of single- and multi-node capabilities, and Nalu runs were constructed to exercise all four subsystems.

Parallel scope. This category reflects the main focus of each benchmark as we have constructed it, not all of its capabilities. For example, HPL can of course be run at smaller scales, but we (and arguably, the HPC community as a whole) primarily use it to test computation at scale. Similarly, IOR and mdtest may inform single-node performance, but our concern was more focused on I/O performance as jobs scale up. Our four categories were defined as

a. Serial: no explicit shared memory or distributed memory parallelism. Any computational parallelism (*e.g.*, pipelining, SIMD) would be discovered and implemented by the compiler.

b. Multi-threaded or multi-process parallelism on a single node. We have defined a "node" as a single hardware entity not requiring communication through an external switch (*e.g.*, Infiniband) to enable inter-process communication. This class reflects much of the observed demand at NREL, nicely balancing parallel acceleration through a limited scaling regime and overall throughput of completely independent jobs for, *e.g.*, high-throughput parameter exploration.

c. Multi-node parallelism. A second class of demand we have observed involves codes and problems which do not scale especially well, but for which there is enough acceleration over < 10 nodes to, *e.g.*, fit a job inside a walltime limit, or to acquire intermediate results which will guide further exploration.

d. Scalable. This is the traditional HPC category of work, involving carefully parallelized code and a single substantial computation with memory or time demands prohibiting solution on smaller numbers of nodes. At NREL, these have typically involved classical molecular dynamics or fluid dynamics-heavy multiphysics runs.

Software scope. Community benchmarks have historically focused on either single hardware subsystems (processing, memory, I/O), or single algorithms (matrix multiplication, FFT, etc.). We labeled this benchmark type a "kernel." Some more recent collections, *e.g.*, Mantevo [27], have expanded into collections of kernels with common co-occurrence, which we have labeled "mini-applications." Although these two categories can capture most of the information needed for a performance specialist to draw conclusions regarding hardware and toolset suitability, they still represent an indirect measure of performance versus representative problems that are commonly seen in a particular facility. Thus, we have extended our collection to include "full applications." Obviously, these cannot capture everything that a particular application could do, nor would comparisons arising from such a comprehensive analysis provide much direct value (*i.e.*, small vs. big data). However, as formulated they represent common job types important to NREL's research community. For example, the VASP benchmark includes two job types: a small unit-cell, high-accuracy many-body calculation reflecting work on semiconductor band structure, and a large unit-cell, DFT GGA slab calculation more common for catalysis studies. Finally, a straightforward extension toward evolving demands couples many applications together, where bottlenecks may involve file-based communication, software stack complexity, or simply usability. This "workflow" category is less a hardware benchmark, and more focused on the ability of an HPC ecosystem to scale up to heterogeneous computing models.

Task coupling. HPC traditionally focuses on what we consider here as "tight" task coupling, with a large message rate (hence latency-sensitive) and often a large message volume as well (hence bandwidth-sensitive). NREL's workload is composed of a mixture of workflows, many of which fall outside of this paradigm. We also considered this dimension as primarily concerned with tasks as analogous to MPI ranks or processes (*e.g.*, with a separate address space, unique process and thread group ID) as opposed to threads (*i.e.*, sharing an address space, sharing process and/or thread group IDs). So from this viewpoint, STREAM has a single task; HPL is primarily bound by FPU and cache performance and thus might represent medium coupling; and, IMB, HPGMG, Nalu, VASP, and LAMMPS would be considered tightly coupled. This measure was not defined to be particularly quantitative, but rather to understand how a benchmark collection might be biased toward one or the other end of coupling strength.

Data transfer. Except for programs that would reside entirely in registers, benchmarks will involve some degree of data transfer that is characteristic and may be rate-limiting. HPL runs of substantial size may be configured to hold all data in cache, and so we would characterize the data transfer character of this benchmark as cache-core (although some data obviously must be exchanged between nodes). I/O benchmarks are primarily concerned with storage, and so would be most characterized as involving local (LFS), network-attached Ethernet (NFS), or parallel high-performance (PFS)-memory transfer. This dimension is intended to indicate where one expects the primary bottleneck to reside.

Performance type. Maximum application performance in the context of a benchmark reflects an absolute ceiling that software

or algorithms of that type should be able to achieve on the system with the, *e.g.*, compiler flags used. This is distinct from theoretical peak performance, *i.e.*, assuming that computational units are the sole rate-limiting step and are saturated with work, as well as sustained performance for a loaded system. The last (which we denote simply as "sustained" for brevity) is intended to take into account the effects of multiple loads accessing shared resources in an average case scenario—neither the ideal case of a maximum application performance benchmark, nor pathological cases like every running job requiring the resource concurrently. A sustained performance benchmark can be useful in setting expectations of a system in production.

Algorithms. These reflect typical atomic computational tasks within an application run—neither synthetic microkernels nor full applications. They "constitute classes where membership in a class is defined by similarity in computation and data movement." [7] So, not so much "LU decomposition" across any matrix type, but dense vs. sparse matrix operations. In our case, the HPL and HPGMG benchmarks might be considered examples of dense linear algebra and unstructured grid (given cycles through multiple layers of regular grids, one might anticipate a degree of irregular data access somewhere between structured and unstructured grids overall), respectively.

This classification is not universally assignable to benchmarks—microkernel and application benchmarks would fall outside of it. Thinking of it as a dimension, then, for classification purposes those benchmarks would have a null projection onto it (*i.e.*, reside in a subspace orthogonal to algorithmic classification).

Interactive productivity. The ESIF-HPC computing facility supports a variety of work styles beyond bulk batch computing. One critical need that it serves is creation and modification of meshes, which is often most easily done interactively through a GUI. More generally, intermediate visualization (as opposed to final generation of publication-quality graphics) can dramatically accelerate discovery by tapping into the highly evolved human visual system's abilities. Our strategy has been to provide heavily provisioned nodes with hardware rendering capability, which has served our user community well. However, it would be beneficial to quantify this type of workload. While we define the dimension here, the benchmark suite does not currently contain such a component. We do anticipate creating one involving remote access large-mesh creation and refinement workflows. Such a test would exercise not only the central compute capabilities, but also graphics rendering and the peripheral networking connecting a data center with the outside world. Indeed, having a formalized benchmark like this would have caught an issue with graphics hardware branding that was only discovered after the Eagle system went into production.

C. Private vs. public release

In the course of developing the benchmark suite, we kept repository access relatively restricted to a group of developers with mutual trust. Our concerns were to make contributors comfortable enough to take more initiative than they might otherwise in front of a public audience. Nevertheless, the potential impact of this work is severely limited without a public release, and in the worst case will lead to duplication of effort should someone that would have used the repository is forced to redevelop the same capability. We have thus released the suite at https://github.com/NREL/ESIFHPC2. Licensing information is included in each benchmark's top-level README.md file. We hope that the materials will serve as a convenient starting point for those tasked with similar evaluations, the source for subsequent improvements, a base for improvements submitted through pull requests, and our own primary store of knowledge for future benchmarking efforts.

IV. Discussion

A. Value of a formalized process

Benchmarking is of growing importance as hardware and software grow in complexity and diversity. While there are already industry standard collections (*e.g.*, SPEC), we sought to create a cost-free collection that can be easily obtained, adapted, and built to benchmark multiple elements of an HPC computer system (hardware compute performance, memory and network latency/bandwidth, I/O subsystems, and key application performance). In terms of the ESIF-HPC-2 benchmarking activity, the creation and execution of a formal benchmarking effort certainly served to increase our own knowledge and competence in assessing an HPC system, as well as to socialize the importance of performance measurement beyond the core group of contributors. Because these benchmarks were ultimately used by vendors without easy access to the contributors, internal third-party assessment was crucial in filling in instructional holes created by contributors' familiarity with the benchmark. Finally, assessment by external HPC experts was invaluable in improving the benchmarks and filling in knowledge gaps. In the end, we were able to quantify the performance of Eagle precisely in terms of individual subsystems, at scale, across multiple important application categories, under sustained load, and for emergent workloads.

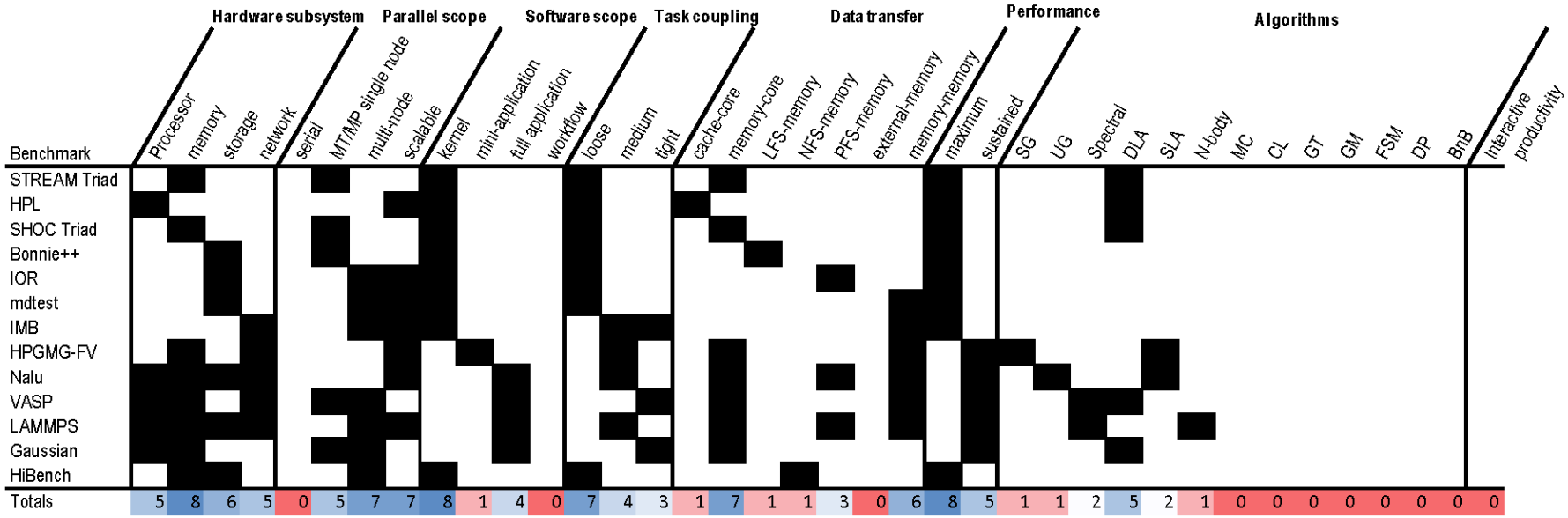
B. Future targets

Computing systems are undergoing rapid evolution as the traditional increase in transistor counts is less able to translate directly into compute performance, and as new paradigms like machine learning begin to take mindshare from model-based technical computing. The reliance on accelerators for leading-edge performance going forward and the rise of data science are undoubtedly harbingers for many technical advances in the near future, and our benchmark suite will need to adapt to these accordingly to stay relevant. By formulating and thinking along "dimensions", we are able to identify gaps systematically and grow the suite to ensure balance and add new information.

Acknowledgment

We thank the following people for development and testing: Shreyas Ananthan, Matt Bidwell, Marc Henry de Frahan, Ray Grout, Ross Larsen, Hai Long, Monte Lunacek, Caleb Phillips, Avi Purkayastha, Matthew Reynolds, Jon Rood, Jeff Simpson, Harry Sorensen, Stephen Thomas, and Deepthi Vaidhynathan This work was authored by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding was provided by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government.

TABLE III. ESIF-HPC-2 BENCHMARK FEATURES. EACH COLUMN IS A FEATURE, AND EACH ROW CAN BE VIEWED AS A VECTOR IN ONE-HOT ENCODING, WITH BLACK DENOTING THAT THE FEATURE IS PRESENT OR RELEVANT TO THAT BENCHMARK (A VALUE OF 1) AND WHITE THAT IT IS NOT (A VALUE OF 0). THE SUMS ALONG THE BOTTOM REFLECT HOW MUCH THAT FEATURE IS REPRESENTED IN THE SUITE, COLOR-CODED FROM BLUE (HIGHLY REPRESENTED) TO RED (NOT REPRESENTED).



References

[1] T. Pyzdek, "Benchmarking," in *The Six Sigma Handbook: A Complete Guide for Green Belts, Black Belts, and Managers at All Levels*, Revised and Expanded., New York: McGraw-Hill, 2003, pp. 91–96.

[2] J. J. Dongarra, P. Luszczek, and A. Petitet, "The LINPACK Benchmark: Past, Present, and Future," University of Tennessee, Knoxville, TN, Dec. 2001.

[3] R. C. Murphy, K. B. Wheeler, B. W. Barrett, and J. A. Ang, "Introducing the Graph 500," presented at the Cray Users Group, Edinburgh, 2010, p. 5.

[4] W. Feng and K. W. Cameron, "The Green500 List: Encouraging Sustainable Supercomputing," *Computer*, vol. 40, no. 12, pp. 50–55, Dec. 2007, doi: 10.1109/MC.2007.445.

[5] D. Manheim and S. Garrabrant, "Categorizing Variants of Goodhart's Law," *arXiv.org*, p. 1803.04585v3, Apr. 2018, doi: https://arxiv.org/pdf/1803.04585.pdf.

[6] K. Regimbal, I. Carpenter, C. Chang, and S. Hammond, "Peregrine at the National Renewable Energy Laboratory," in *Contemporary High Performance Computing: From Petascale to Exascale*, vol. 2, J. S. Vetter, Ed. Taylor & Francis, 2015, pp. 163–184.

[7] K. Asanović *et al.*, "The Landscape of Parallel Computing Research: A View from Berkeley." EECS, University of California at Berkeley, 18-Dec-2006.

[8] G. Fox, S. Jha, J. Qiu, S. Ekanayake, and A. Luckow, "Towards a Comprehensive Set of Big Data Benchmarks," in *Big Data and High Performance Computing*, vol. 26, 2015, pp. 47–66.

[9] J. D. McCalpin, "Memory Bandwidth and Machine Balance in Current High Performance Computers," *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pp. 19–25, 1995.

[10] B. Armstrong, H. Bae, R. Eigenmann, F. Saied, M. Sayeed, and Y. Zheng, "HPC Benchmarking and Performance Evaluation with Realistic Applications," presented at the SPEC Benchmark Workshop, University of Texas, Austin, TX, 2006, pp. 1–11.

[11] A. Danalis *et al.*, "The Scalable Heterogeneous Computing (SHOC) Benchmark Suite," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, Pittsburgh, PA, 2010, pp. 63–74, doi: https://doi.org/10.1145/1735688.1735702.

[12] R. Bourbonnais, "Decoding Bonnie++," 15-Dec-2008. [Online]. Available: https://blogs.oracle.com/roch/decoding-bonnie. [Accessed: 27-Apr-2018].

[13] B. Gregg, "Active Benchmarking: Bonnie++," *Brendan Gregg's Blog*, 08-Feb-2014. [Online]. Available: http://www.brendangregg.com/ActiveBenchmarking/bonnie++.html. [Accessed: 12-Sep-2018].

[14] H. Shan, K. Antypas, and J. Shalf, "Characterizing and Predicting the I/O Performance of HPC Applications using a Parameterized Synthetic Benchmark," 2008, pp. 1–12, doi: 10.1109/SC.2008.5222721.

[15] J. Borrill, L. Oliker, J. Shalf, and H. Shan, "Investigation of Leading HPC I/O Performance using a Scientific-Application Derived Benchmark," in *High Performance Computing, Networking, and Storage Conference*, 2007, doi: 10.1145/1362622.1362636.

[16] J. Kwack and G. H. Bauer, "HPCG and HPGMG Benchmark Tests on Multiple Program, Multiple Data (MPMD) Mode on Blue Waters-A Cray XE6/XK7 Hybrid System," *Concurr. Comput. Pract. Exp.*, vol. 30, no. 1, p. e4298, Jan. 2018, doi: 10.1002/cpe.4298.

[17] M. F. Adams, J. Brown, J. Shalf, B. Van Straalen, E. Strohmaier, and S. Williams, "HPGMG 1.0: A Benchmark for Ranking High Performance Computing Systems," Lawrence Berkeley National Lab, LBNL-6630E, 1131029, May 2014.

[18] V. Marjanović, J. Gracia, and C. W. Glass, "HPC Benchmarking: Problem Size Matters," presented at the 7th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems, Salt Lake City, 2016, pp. 1–10, doi: 10.1109/PMBS.2016.006.

[19] M. J. Frisch *et al.*, *Gaussian 16, Rev. B.01*. Wallingford, CT: Gaussian, Inc., 2016.

[20] G. Kresse and J. Furthmüller, "Efficient Iterative Schemes for ab Initio Total-Energy Calculations Using a Plane-Wave Basis Set," *Phys. Rev. B*, vol. 54, no. 16, pp. 11169–86, 1996.

[21] G. Kresse and J. Furthmüller, "Efficiency of ab-Initio Total Energy Calculations for Metals and Semiconductors using a Plane-Wave Basis Set," *Comput. Mater. Sci.*, vol. 6, pp. 15–50, 1996.

[22] G. Kresse and J. Hafner, "*Ab initio* Molecular Dynamics for Liquid Metals," *Phys. Rev. B*, vol. 47, no. 1, pp. 558–561, 1993, doi: 10.1103/PhysRevB.47.558.

[23] G. Kresse and J. Hafner, "*Ab initio* Molecular-Dynamics Simulation of the Liquid-metal-Amorphous-semiconductor Transition in Germanium," *Phys. Rev. B*, vol. 49, no. 20, pp. 14251–14271, Mar. 1994, doi: 10.1103/PhysRevB.49.14251.

[24] S. Plimpton, "Fast Parallel Algorithms for Short-Range Molecular Dynamics," *J. Comput. Phys.*, vol. 117, no. 1, pp. 1–19, Mar. 1995, doi: 10.1006/jcph.1995.1039.

[25] Rui Han, Lizy Kurian John, and Jianfeng Zhan, "Benchmarking Big Data Systems: A Review," *IEEE Trans. Serv. Comput.*, vol. PP, no. 99, pp. 1–18, 2017, doi: 10.1109/TSC.2017.2730882.

[26] Y. Deng, P. Zhang, C. Marques, R. Powell, and L. Zhang, "Analysis of Linpack and Power Efficiencies of the World's TOP500 Supercomputers," *Parallel Comput.*, vol. 39, no. 6–7, pp. 271–279, 2013, doi: 10.1016/j.parco.2013.04.007.

[27] M. A. Heroux *et al.*, "Improving Performance via Mini-Applications," Sandia National Laboratories, Albuquerque, NM, SAND2009-5574, 2009.