Updating the SPP Benchmark Suite for Extreme-Scale Systems

Gregory Bauer, Victor Anisimov, Galen Arnold, Brett Bode, Robert Brunner, Tom Cortese, Roland Haas, Andriy Kot, William Kramer, JaeHyuk Kwack, Jing Li, Celso Mendes, Ryan Mokos, and Craig Steffen
National Center for Supercomputing Applications
University of Illinois
Urbana, Illinois 61801, USA
{gbauer,anisimov,gwarnold,brett,rbtunner,tcortese,rhaas,andriy,wtkramer,jkwack2,lijing,cmendes,mokos,csteffen}@illinois.edu

***Abstract*—For the effective deployment of modern extreme-scale systems, it is critical to rely on meaningful benchmarks that provide an assessment of the achievable performance a system may yield on real applications. The Sustained Petascale Performance (SPP) benchmark suite was used very successfully in evaluating the Blue Waters system, deployed by Cray in 2012, and in ensuring that the system achieved sustained petascale performance on applications from several areas. However, some of the original SPP codes did not have significant use, or underwent continuous optimizations. Hence, NCSA prepared an updated SPP suite containing codes that more closely reflect the recent workload observed on Blue Waters. NCSA is also creating a public website with source codes, input data, build/run scripts and instructions, plus performance results of this updated SPP suite. This paper describes the characteristics of those codes and analyzes their observed performance, pointing to areas of potential enhancements on modern systems.**

Keywords-benchmark; performance; petascale; acceptance.

I. INTRODUCTION

Benchmarks for large-scale computing systems have a history that dates back to the origins of computing systems. Because performance is the major motivation for employment of these systems, it is necessary to create mechanisms that assess their usable, achievable performance in order 1) to compare different systems so that value and utility alternatives can be determined, 2) to assure systems from vendors can accomplish their mission and meet acceptance criteria, 3) to continue to assess the ability of a system to meet its mission in full service over its life-time, and 4) to encourage and enable vendors to further improve existing designs for the future.

For the effective deployment of modern post-petascale and upcoming exascale systems, it is critical to rely on effective and meaningful benchmarks that provide an assessment of the real and achievable performance a system may yield on real applications. It is the use of such applications, and in particular the scientific and engineering advances enabled by their execution, that justify the investments made in the acquisition of large-scale computational and data systems. Thus, it becomes essential to utilize robust benchmarks that realistically represent the expected workload of the computing systems.

The Sustained System Performance (SSP) method developed at Berkeley [1] has several advantages over traditional benchmarking approaches. Different implementations of this method have been used at multiple sites to assess the performance of systems. The Sustained Petascale Performance (SPP) metric [2] is the first heterogeneous implementation of SSP. It was used very successfully in evaluating the performance of the Blue Waters system, the largest machine ever built by Cray, and in ensuring that the system met its acceptance criteria. Despite its great value for Blue Waters acceptance and deployment, operation of the Blue Waters system over the past four years has shown that the original SPP suite needed an update. To enable this update and preserve the usefulness of SPP, NCSA has prepared a new version of the SPP suite. This new version removes codes that did not have significant use on Blue Waters, and adds applications associated with large allocations awarded by NSF that have run at scale recently.

In addition, to promote wide availability of this new SPP suite, NCSA is finalizing preparation of a public website that, besides the source code for each SPP application, hosts the input data for test cases. This website also contains instructions to build and run the codes, with corresponding example scripts and the results of executions conducted on Blue Waters. In summary, this updated SPP suite should enable a fair performance assessment of upcoming extreme-scale systems targeting execution of workloads from different scientific disciplines. More importantly, it will reflect the expected performance of codes that are actually relevant for their communities, allowing these communities to continue their trend of notable scientific discoveries observed in recent years.

The remainder of this paper is organized as follows. In Section II, we review major benchmarks currently available for large systems, including the original SPP suite that we used in 2012. Section III describes the components of the new SPP suite, with their performance results observed so far on Blue Waters and potential directions for improvements. Section IV contains details of the public website created by NCSA for distribution of the new SPP suite. Finally, Section V concludes our presentation, summarizing its major points and expected impact.

II. BACKGROUND

In this section, we briefly analyze the evolution of notable benchmarks for large systems, leading to the current status. We start discussing traditional benchmarks, and then review details of the SPP Benchmarks, which we used as a starting point for the updated version described in this paper.

A. *Traditional Benchmarks*

Comparison measures can range from assessing the simple, single method tests to using full application tests with a range of problem data sets. Simple, single method performance probes include LINPACK [3], GUPS [4], ping-pong [5], Mpptest [6], STREAM [7], and the recent HPCG [8]. Collections of tests, ranging from simple kernels such as HPCC [9] and Livermore FORTRAN Kernels [10], help expand the testing parameter space but provide limited inference for full applications. Pseudo applications (aka mini-Apps) such as NAS Parallel Benchmarks [11] and SPEC [12] come closer to representing applications but have other limitations, for example not including realistic I/O, or keeping I/O separated rather than integrated to the computational tests as one would desire.

Of course the best tests are the actual, complete applications, but they are site and community specific, can take very long times to run and are often difficult to project performance estimates to run on target systems before those systems exist. Most high-performance computing and data systems (HPCD) support a broad spectrum of users and applications. The resulting workload is too diverse to be represented by a small subset of simplified kernels. Multiple full application tests, such as those in many procurement calls for proposals, come closer to representing large workloads, but become non-trivial to implement on different systems and more difficult to interpret in a generalized manner.

The SPP suite addresses the limitations [13] of past methods while expanding the use of a scalable, cohesive, flexible benchmarking evaluation method for the future.

B. *Sustained System Performance (SSP) Method and the Sustained Petascale Performance (SPP) Tests*

The Sustained System Performance (SSP) Method [1] has several advantages over traditional benchmark methods. First, it focuses on real time to solution for real work rather than rates that may be artificially inflated. Second, it typically represents full applications that include both scientific and defensive I/O (e.g., as used for checkpointing). Third, it can determine not only individual measures, but it can also composite the results into a single, meaningful measure for how much real work (aka sustained performance) a system is truly capable of producing. The composite work potential can then be used in value calculations to assess alternative options. Different implementations of the SSP method have been used at multiple sites to assess the performance of systems.

The Sustained Petascale Performance (SPP) metric [2] is the first heterogeneous implementation of SSP and was used very successfully in evaluating the performance of the Blue Waters system (1.3 sustained petaflops). The original SPP suite was composed of 12 full codes and 14 input sets that represented a large portion of the NSF frontier science computational workload. The SPP extended the SSP method to be able to assess a system with heterogeneous mixes of different types of nodes and varying implementations, which has also influenced other sites using SSP methods.

Because Blue Waters is a hybrid machine, containing two types of nodes (XE, with dual x86 processors, and XK, combining an x86 processor and a GPU), each set of nodes of a given type has the SPP contribution calculated independently, and those sustained measures are summed to obtain the full-system SPP value. In summary, SPP is a time-to-solution metric that represents end-to-end runs, including I/O, pre and post-processing, etc.

The following mix of application codes and respective areas, corresponding to a significant representation of the NSF workload originally expected for Blue Waters, was selected for the initial SPP acceptance tests:

- NAMD – biology (molecular dynamics)
- MILC, CHROMA - particle physics (quantum chromodynamics)
- VPIC, SPECFEM3D - geophysical science
- WRF - atmospheric science
- PPM - astrophysics
- NWCHEM, GAMESS - quantum chemistry
- QMCPACK - materials science

Four of those applications (NAMD, CHROMA, QMCPACK, and GAMESS) had versions to run either on the CPUs (AMD Interlagos processors) or on the GPUs. The remaining six applications were not yet GPU-enabled, and ran exclusively on the CPUs.

To confirm that the set of SPP codes originally selected were good representatives of the actual workload, Fig.1 shows the distribution of Blue Waters usage across the various science and engineering areas during the period between July 2013 and August 2014. Each of the five areas in the figure with the largest consumption, which jointly account for 85% of the total system usage, has a corresponding code among the selected SPP benchmarks listed above. Thus, we can claim that our original SPP selection was fully successful in characterizing the initial Blue Waters workload.

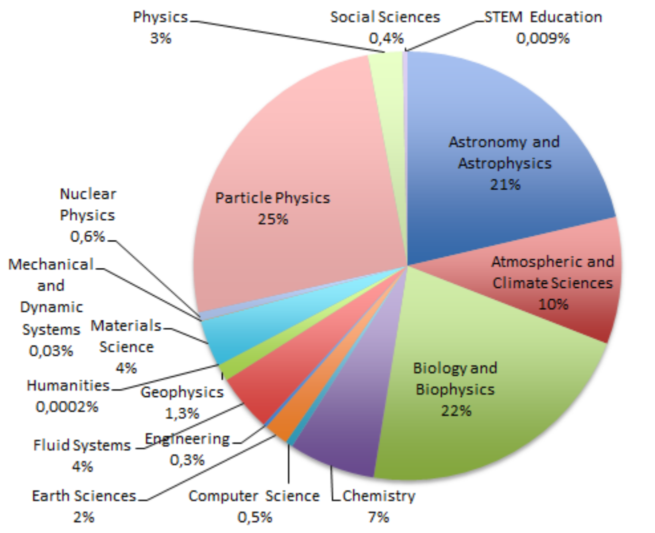

Figure 1 – Blue Waters usage distribution by science area according to the node-hours (Jul'2013-Aug'2014).

C. *Performance of Original SPP Codes on Blue Waters*

During Blue Waters acceptance, in 2012, optimized versions of the SPP applications were executed and the performance of their full runs (i.e., including overheads of typical production runs, like I/O, checkpointing, etc) were recorded. Meanwhile, for each code, an accounting of the floating-point operations executed in each case was conducted. The precise accounting mechanism employed varied across the applications. For some of them, the source code was instrumented with PAPI [14], and the flop count was read directly from the hardware performance counters at the end of the execution. For other applications, an actual count of operations was manually determined for basic segments of the code, and the total number of operations was obtained by measuring the number of times that each segment executed in the full run. After establishing a flop count at the desired configuration for each code, the applications were executed on Blue Waters without any instrumentation added, to measure their actual execution times without instrumentation overheads.

Table I shows the performance measured, in 2012, for each of the SPP applications that were executed on Blue Waters' XE nodes (i.e., nodes with x86 CPUs only). Besides the flop count and execution times, which were mentioned in the previous paragraph, the table also shows the number of nodes that were used in each execution. With that information, it is possible to derive the average node performance, in GF/s. It is also possible to estimate, assuming ideal scaling, the performance that would be obtained for that given code running on the entire set of XE nodes in Blue Waters. While not all applications were ready to scale to the entire system, this scaled estimate is still useful to assess the effectiveness of each code as a component of the mix of codes that typically run on Blue Waters.

Given the individual applications' performance at scale, corresponding to the last column in the table, one can determine the value of the SPP metric by computing the geometric mean of those values. For the values in the table, the geometric mean is 1.063 PF/s, indicating that Blue Waters is capable of running a production workload consisting of a certain mix of those applications at a sustained rate beyond one Petaflop/second. It must also be noted that, according to previous reports [15], some of these applications were indeed run at scale on Blue Waters and achieved more than 1 PF/s of sustained performance, individually.

Table I – Results from SPP application executions on Blue Waters in 2012

Application	Flop Count	Number of Nodes	Time (s)	Node Rate (GF/s)	BW Rate (PF/s)
VPIC	$1.83\text{x}10^{18}$	4,608	5,811.0	68.26	1.65
QMCPACK	$4.71\text{x}10^{17}$	4,800	1,852.0	52.98	1.28
NAMD	$8.29\text{x}10^{17}$	5,000	5,432.4	30.50	0.74
WRF	$1.05\text{x}10^{18}$	4,560	8,931.0	25.81	0.62
MILC	$4.73\text{x}10^{17}$	4,116	5,099.5	22.52	0.54
PPM	$2.57\text{x}10^{19}$	8,256	46,848.0	66.45	1.61
SPECFEM3D	$6.30\text{x}10^{18}$	5,419	16,918.4	68.70	1.66
NWCHEM	$5.95\text{x}10^{18}$	5,000	24,852.2	47.87	1.16

III. New Sustained Petascale Performance Benchmark Suite

To ensure that the SPP Benchmark suite remains relevant, it must be updated periodically such that it always contains codes that are actually important for the scientific community. Hence, while codes with low usage may be removed, new codes with emerging adoption by users of a given area may be considered for incorporation to the suite.

Our analysis of the components in the SPP suite was strongly based on a detailed study of the Blue Waters' observed workload from April 2013 to September 2016, conducted by researchers from University at Buffalo, SUNY [16]. That study indicated, among other things, that the top five most used applications accounted for more than 50% of Blue Waters cycles during the period analyzed. In particular, NAMD was the application with most execution cycles, indicating that it must certainly continue to participate in the SPP suite.

The selection of codes for the new SPP benchmark suite was also influenced by NCSA's own measurements indicating that, between June 2015 and May 2016, the five disciplines with most cycles spent on Blue Waters were these, respectively: biophysics (18.9%), elementary particle physics (16.4%), stellar astronomy and astrophysics (12.6%), chemistry (8.1%), and molecular biosciences (6.1%). Hence, unlike other benchmarks that focus on specific performance aspects, the updated SPP suite, described below, will continue to reflect the capability of a given system to handle workloads that cover a variety of scientific areas.

A. *Description of the New SPP Codes*

The codes composing the new SPP suite, and their corresponding scientific areas, are the following:

- AWP-ODC: a code to study wave-propagation in earthquakes;
- CACTUS: a problem-solving environment for astrophysics;
- MILC: a collection of codes to study quantum chromodynamics (QCD);
- NAMD: a code for molecular dynamics simulations;
- NWCHEM: a suite of computational chemistry tools;
- PPM: a code for gas dynamics simulations;
- PSDNS: a code for direct numerical simulations of 3D unsteady turbulent fluid flows;
- QMCPACK: a Quantum Monte Carlo code for many-body ab initio simulation of atoms;
- RMG: a code for electronic structure calculations and modeling of materials and molecules;
- VPIC: a code to follow the movement of charged particles;
- WRF: a mesoscale numerical weather prediction code.

The programming languages and parallelization paradigm used by each of these SPP applications are shown in the table below, while a general description of each code is provided in the following subsections.

Table II - Programming language and parallelization paradigm for codes in the updated SPP suite.

Application	Language	Parallelization
AWP-ODC	Fortran, C++	MPI
CACTUS	Fortran, C, C++	MPI+OpenMP
MILC	C, C++	MPI+OpenMP
NAMD	C++	Charm++
NWCHEM	Fortran, C, C++	Global Arrays
PPM	Fortran	MPI+OpenMP
PSDNS	Fortran	MPI+OpenMP+CAF
QMCPACK	C, C++	MPI+OpenMP
RMG	Fortran, C, C++	MPI+pthreads
VPIC	C++	MPI+OpenMP
WRF	Fortran	MPI+OpenMP

AWP-ODC

The Anelastic Wave Propagation, AWP-ODC [17], independently simulates the dynamic rupture and wave propagation that occurs during an earthquake. Dynamic rupture produces friction, traction, slip, and slip rate information on the fault. The moment function is constructed from this fault data and used to initialize wave propagation.

A staggered-grid finite difference scheme is used to approximate the 3D velocity-stress elastodynamic equations. The user has the choice of modeling dynamic rupture with either the Stress Glut (SG) or the Staggered Grid Split Node (SGSN) method. The user also has the choice of two external boundary conditions that minimize artificial reflections back into the computational domain: the absorbing boundary conditions (ABC) of Cerjan and the Perfectly Matched Layers (PML) of Berenger.

CACTUS

Cactus [18] is a publicly available, community driven, open-source problem solving environment designed for scientists and engineers. Its modular structure easily enables parallel computation across different architectures and collaborative code development between different groups. Zelmani is a core collapse supernova code built on top of Cactus. It employs the adaptive mesh refinement driver Carpet, the metric solver McLachlan and the hydrodynamics code GRHydro of the Einstein Toolkit and the curvilinear grid code Llama. Zelmani provides neutrino radiation transport and complex equations of state required to simulate core collapse supernovae.

For this new SPP suite, version **ET_2016_05** of the Einstein Toolkit and version 45bb0d7 and c50f88b of Zelmani, checked out on December 26, 2016, are used. The code was modified to reduce startup time in grid creation and data reading and those changes were partially incorporated in the current development branch of the Einstein Toolkit.

MILC

The benchmark code MILC [19] represents part of a set of codes written by the MIMD Lattice Computation (MILC) collaboration used to study quantum chromodynamics (QCD), the theory of the strong interactions of subatomic physics. It performs simulations of four dimensional SU(3) lattice gauge theory on MIMD parallel machines. "Strong interactions" are responsible for binding quarks into protons and neutrons and holding them all together in the atomic nucleus. QCD discretizes space and evaluates field variables on sites and links of a regular hypercube lattice in four-dimensional space time. Each link between nearest neighbors in this lattice is associated with a 3-dimensional SU(3) complex matrix for a given field.

The MILC collaboration has produced application codes to study several different QCD research areas, only one of which is used in this SPP suite. This code generates lattices using rational function approximations for the fermion determinants using the Rational Hybrid Monte Carlo (RHMC) algorithm and implementing the HISQ action.

NAMD

NAMD [20] is an application for performing classical molecular dynamics simulations of biomolecules that is able to scale to 100 million atoms on hundreds of thousands of processors. Interactions between atoms in the simulation are parameterized based on the species of each atom and its chemical role. The forces on all atoms are integrated by the explicit, reversible, and symplectic Verlet algorithm to simulate the dynamic evolution of the system with a timestep of 2 fs.

The force field includes bond forces among groups of 2-4 atoms, Lennard-Jones forces, and short-range and long-range electrostatics. All of these forces, except the long-range electrostatics, are localized and scale well on large distributed computers. The long-range electrostatics are computed using the FFT-based particle-mesh Ewald (PME) algorithm, which requires computing two 3-D FFTs every time step. Since NAMD is designed to overlap various force computations, with increasing numbers of compute nodes, the local computation time decreases to the point where the all-to-all communication for the FFT stage dominates execution time.

The science problem is a 100M atom chromatophore simulation, running a constant-pressure equilibration at 298K. The benchmark input file was assembled by equilibrating the chromatophore of the purple bacterium *Rodobacter Sphaeroids*. This provides a science input set appropriate for scaling NAMD to petascale systems.

NWCHEM

NWChem [21] provides scalable computational chemistry tools that are able to treat large scientific computational chemistry problems, and efficiently use available parallel computing resources, from conventional workstation clusters to high-performance parallel supercomputers. The NWChem development community represents a consortium of developers led by the EMSL team located at the Pacific Northwest National Laboratory (PNNL) in Washington State. The NWChem development strategy focuses on delivering essential scientific capabilities to its users in the areas of kinetics and dynamics of chemical transformations, which occur in gas phase, at interfaces, and in condensed phase.

The benchmark test involves an electronic structure calculation of Guanine dimer at Coupled Cluster Singles and Doubles excitation including perturbative Triple excitations known as CCSD(T) level of theory. The computation employs correlation-consistent aug-cc-pvtz basis set. The outcome of the computation is the total energy of the molecular system at the fixed geometry representing a stationary solution of the Schroedinger equation.

PPM

The piecewise parabolic method (PPM) [22], developed initially for gas dynamical simulations including shocks and discontinuities in stellar processes, has been applied to the problem of inertial confinement fusion (ICF). The ICF problem, like the star problems, involves turbulent mixing due to instabilities at multi-fluid interfaces, and in both problems the details of this mixing affect the combustion that results. The ICF problem exercises all the features of the PPM codes, including strong shocks and very elaborate treatments of unstable multifluid interfaces, while at the same time performing an important scientific problem that has a highly transient character.

The ICF process is initiated by shining 192 powerful laser beams on a container in which a small capsule is suspended. The spherical capsule confining the hydrogen fuel is driven radially inward by the very high pressure of the surface material heated by the laser light. This inward motion exceeds the speed of sound in the unshocked capsule and fuel materials, and therefore it takes place very rapidly. In just a few sound crossing times of the system, the compression of capsule and fuel is complete. In this problem, the time step is set by the speed of sound in the hot gases. Thus, only a relatively small number of time steps are needed. The test case is a modification of the ICF test problem devised by Youngs in 2008. These modifications increase the radial compression that it produces from about a factor of 4 to about a factor of 10. This is still much less than is needed for inertial confinement fusion, but it is closer, and the test problem is still feasible to specify and set up. The emphasis is on the mixing of the capsule and fuel materials at the inner surface of the capsule.

All the fluids are treated as ideal monatomic gases. Shock Mach numbers are then very high, which of course would ionize the material. This detail is not especially relevant to the unstable behavior of the capsule-fuel boundary. Instead, the focus is the interface instability in the context of a strongly converging flow field.

PSDNS

PSDNS [23] is a highly parallelized application code used for performing direct numerical simulations (DNS) of three-dimensional unsteady turbulent fluid flows, under the assumption of statistical homogeneity in space. A system of partial differential equations expressing the fundamental laws of conservation of mass and momentum is solved using Fourier pseudo-spectral methods in space and explicit Runge-Kutta integration in time. The pseudo-spectral approach requires multiple FFTs in three directions per time step, resulting in communication-intensive operations due to transposes involving collective communication among processors. However, remote-memory addressing techniques, such as Co-Array Fortran, have been found to be helpful on Blue Waters.

Simulation outputs can include flow-field information stored at a large number of grid points, as well as the trajectories of infinitesimal fluid elements, which are tracked over sustained periods of time for the study of turbulent dispersion. The current code has enabled a production simulation of turbulent flow using 4 trillion grid points on 262,144 CPU cores.

For the SPP benchmark, PSDNS is initialized with a simple sinusoidal velocity field, and solves for the velocity field variables using the 4th order Runge–Kutta method. Its I/O and checkpoint are performed at the frequency specified in the input file.

QMCPACK

QMCPACK is a Quantum Monte Carlo (QMC) code [24] used for many-body ab initio simulation of atoms, molecules, and solid materials. Open-source and written in C++, QMCPACK solves the many-body Schrödinger equation by stochastically sampling the configuration domain. A Variational Monte Carlo (VMC) algorithm can be used either to obtain the results directly or to quickly find a “ballpark” estimate of the solution, which is then refined by a Diffusion Monte Carlo (DMC) algorithm. In both configurations, a number of VMC walkers (each a complete state representation) randomly move through the energy domain each time step.

If the VMC-DMC setup is used, the VMC walkers are sampled to create walkers for the DMC phase. The output is the lowest energy state within a statistical uncertainty, which can be reduced by taking more samples (i.e., using more walkers).

RMG

RMG [25] is a DFT-based electronic structure code developed at North Carolina State University. The original version was written in 1993-1994 and it has been updated and expanded continuously since that time. It uses real-space meshes to represent the wavefunctions, the charge density, and the ionic pseudopotentials. The real-space formulation is advantageous for parallelization, because each processor can be assigned a region of space, and for convergence acceleration, since multiple length scales can be dealt with separately. Both norm-conserving and ultrasoft pseudo-potentials (UPPs) are allowed. For both efficiency and accuracy, the implementation of UPPs uses three grids, for the wave functions, the charge density and the short-ranged projectors associated with UPPs.

To further reduce the grid density, high-order Mehrstellen discretizations were developed, as an alternative to central difference discretization of the kinetic energy operator. The Mehrstellen discretization employs a weighted sum of the wavefunction and potential values to improve the accuracy of the discretization of the entire differential equation, not just the kinetic energy operator. For a given discretization order, it is also significantly shorter range than central difference formulas, decreasing communication needs in a parallel implementation.

VPIC

To simulate plasma, the Vector Particle-In-Cell (VPIC) code [26] follows the movement of charged particles in simulated electric and magnetic fields that interact with the particles. VPIC integrates the relativistic Maxwell-Boltzmann system in a linear background medium for multiple particle species, in time with an explicit-implicit mixture of velocity Verlet, leapfrog, Boris rotation and exponential differencing based on a reversible phase-space volume conserving 2nd order Trotter factorization. VPIC can be used in studies of magnetic reconnection of high temperature plasmas (H^+ and e^-).

Magnetic reconnection is an energy conversion process that occurs within high temperature plasmas, and often produces an explosive release of energy as magnetic fields are reconfigured and destroyed. In space and astrophysical plasmas, the onset of magnetic reconnection occurs within intense current layers, where the magnetic field rapidly rotates. Most simulations of this process focus on these thin layers, but there are a variety of possibilities to choose from for the initial conditions. While most studies have focused on the Harris-type equilibrium that is relevant to the Earth's magnetosphere, there has been growing interest in the past few years on force-free current sheets that are thought to be more relevant to the solar atmosphere, as well as many astrophysical problems.

The objective of kinetic simulations is to understand the three-dimensional evolution of force-free current layers. One important first step is to understand the 3D evolution of tearing modes – a type of plasma instability that spontaneously produces magnetic reconnection while giving rise to topological changes in the magnetic field.

WRF

The Weather Research and Forecasting (WRF) model [27] is a next-generation mesoscale numerical weather prediction system designed to serve both operational forecasting and atmospheric research needs. WRF is suitable for a broad spectrum of applications across scales ranging from meters to thousands of kilometers. The WRF-ARW core is based on an Eulerian solver using a third-order Runge-Kutta time-integration scheme coupled with a split, explicit second-order time integration scheme.

For the SPP tests, WRF is configured to conduct a simulation of Hurricane Sandy, using a two-level nested grid: a coarser grid over the entire three-dimensional domain, and a finer grid on parts of that domain. The simulation mode, such as the number of time steps to be executed, is controlled via parameters configured in an input namelist.

Table III – Input data characteristics for each SPP code

Application	Compact Input	Large Input
AWP-ODC	128^3 mesh	5600x2800x1024 mesh
CACTUS	6.7x10^7 grid points	2.8x10^9 grid points
MILC	36x36x36x72 lattice	72x72x72x144 lattice
NAMD	2 ps simulation of 100 million atoms	20 ps simulation of 100 million atoms
NWCHEM	32 atoms, 4 ccsd iterations	32 atoms, 20 ccsd iterations
PPM	$1{,}280^3$ zone mesh	$5{,}120^3$ zone mesh
PSDNS	$2{,}048^3$ grid points	$8{,}192^3$ grid points
QMCPACK	5.12x10^4 Monte Carlo samples	2.56x10^6 Monte Carlo samples
RMG	302 water molecules	4,096 atoms
VPIC	$1{,}200^3$ grid, 8.64x10^{10} particles	$1{,}536^3$ grid, 1.16x10^{12} particles
WRF	9120x9216x48 grid, 900 T-steps	9120x9216x48 grid, 9,000 T-steps

B. *Input Problems for the New SPP Codes*

For each code in the new SPP suite, two input test cases were defined. First, a compact test case, intended to be run on approximately a hundred nodes of Blue Waters. This compact case's primary goal is not to enable assessment of a system's performance. It simply allows verification of the proper building of the code and its correct execution with small problems on a given system. Meanwhile, a second, larger test case is designed to run on more than a thousand nodes. For Blue Waters, this may correspond to hundreds of thousands of cores. The precise predefined number of processors employed varies across codes, and together with the compact and large input test cases, are sufficient to represent production workloads used by the teams employing the applications.

Table III lists characteristics of the input data used by each SPP application, for both the compact and large test cases.

C. *Executions of New SPP Codes on Blue Waters*

To provide a performance assessment of the new SPP suite on Blue Waters, we have been testing the codes with the inputs defined in the previous subsection. Our initial executions were done with the compact inputs, to verify the build process for each code, debug compiling and linking problems, and ensure that all codes would execute properly on the system. These preliminary tests were conducted with a few hundred Blue Waters nodes, to ensure that the codes could run quickly and to provide an end-to-end exercise of their utilization on Blue Waters.

Next, we started to test the codes with the large inputs. Due to differences, for some of the codes, between the inputs being used now and those that had been used in 2012, the observed performance in the current tests may be different from the values obtained in the past. Additionally, we have not yet completed a precise determination of the flop counts for all codes. Hence, we simply present here the execution times observed with these codes so far, under the large inputs. These results are in Table IV, and present some variation in the execution durations. The longest execution is for NWCHEM, taking more than six hours, while the shortest execution is for NAMD, which takes slightly more than four minutes.

The largest job among the tests reported in Table IV is for PPM, which executes on 8,448 nodes, corresponding roughly to 88 Blue Waters cabinets, or nearly 36% of all XE cabinets in the system. The smallest job is for MILC, with 1,296 nodes, in approximately 14 cabinets, or 6% of the XE cabinets. The average number of nodes used in the tests of Table IV is 4,655, corresponding to 20% of all XE nodes in Blue Waters. Although many of these codes from the updated SPP set have versions ready or in preparation for GPU execution (namely, AWP-ODC, MILC, NAMD, NWCHEM, PPM, QMCPACK, RMG), we are focusing our initial tests on their CPU versions.

D. Potential Directions for Performance Improvements

Although all of the current SPP applications have been in use by their corresponding communities on Blue Waters, the specific code versions that we chose for the SPP suite may not yet be fully optimized for the problem cases that we picked. Meanwhile, on other machines newer than Blue Waters, there may be optimization opportunities that might lead to improved performance for some of the codes. In this sub-section, we provide, for some of the SPP codes, general directions where optimization efforts could be worth exploring.

CACTUS Improvements

Executions of CACTUS with the defined problem sizes for the SPP suite are such that a considerable part of the runtime is spent on I/O. The code uses HDF5, hence one path for optimization could be to employ optimized parallel I/O. Also, because the code is communication-intensive, a network-aware, system-specific placement of the executing ranks (and corresponding threads) might lead to improve performance.

Table IV – Execution times of new SPP applications with large input on Blue Waters

Application	Number of Nodes	Time (s)
AWP-ODC	2,048	855
CACTUS	4,096	4,800
MILC	1,296	7,916
NAMD	4,500	242.2
NWCHEM	5,000	22,201
PPM	8,448	7,790
PSDNS	8,192	1,538
QMCPACK	5,000	1,765
RMG	3,456	7,310
VPIC	4,608	4,218
WRF	4,560	10,260

MILC Improvements

Given that specific parts of MILC are numerically intensive, the use of *fused multiply-add* (FMA) instructions could lead to superior performance for those parts – more precisely, on functions of the *su3* library. In fact, the version of MILC used in 2012 was accelerated by the FMA4 instructions of the AMD Interlagos processors on Blue Waters. Similar techniques could be adopted by employing SIMD extensions of current processors. One could also explore non-uniform domain decompositions to achieve better load balance, although this should be done carefully so that no numerical instabilities arise. Additionally, since MILC has communication-intensive phases, some kind of rank reordering to better explore the underlying interconnection network might alleviate its communication bottlenecks. This rank-reordering optimization had been used in the 2012 runs with a very positive impact on the observed application performance.

NAMD Improvements

Future optimization work for NAMD includes GPU-offloading of bonded terms and integration targeting Oak Ridge's Summit system and extension of the AVX-512 vectorization currently in use on KNL for Argonne's Aurora system [28]. NAMD is an early-science project on both platforms, and new Charm++ network layers, optimized for the interconnections of those machines, are also in development. More generally, the design of the code will need to move from its legacy thread-centric design to something more process-centric, to reduce overhead on increasingly high core-count or GPU-dependent machines.

NWCHEM Improvements

The 6.1 version of NWCHEM that was used in 2012 for the SPP test relied on the DMAPP [29] network interface for Global-Arrays [30]. The DMAPP layer provided superior performance but was recently discarded by NWCHEM developers due to bugs in the implementation; the current version is entirely based on MPI communication level for Global-Arrays. This improved the portability of the code to different hardware architectures but resulted in some communication performance drop on Cray machines. In general, optimizing the network-specific layer in Global-Arrays will lead to improved performance. In addition, numerically intensive parts of NWCHEM would benefit from optimizations such as vectorization on processors with improved support for SIMD instructions. Significant potential remains for algorithmic optimizations in the CCSD portion of the code, as initiated in previous work [31], which could provide additional performance improvement.

PPM Improvements

Given PPM's structure of dividing the processors in *teams*, with one processor in the team designated as a server in charge of all I/O activities for the team, it is important to optimize the flow of data between team members and the team server. This can be accomplished by careful mapping of the executing processors of a team, via node selection and rank reordering in the submission scripts for the execution. Additionally, compiler optimizations can improve compiler-generated vectorized code and high performance math functions, which can lead to better computational performance of the team members.

QMCPACK Improvements

In 2012, QMCPACK had been optimized on Blue Waters with the use of fused multiply-add FMA4 instructions for the AMD Interlagos processor and with a special version of the *libm* library. The code can also be optimized by using mixed precision in some of the parts dominated by numerical computation.

VPIC Improvements

The computational performance of VPIC could be improved by compiler vectorization and adoption of fused multiply-add instructions, and by other optimizations to eliminate extra data copies. Some form of dynamic load balance for the particles could also potentially improve performance, although this could lead to higher memory consumption in some nodes. To improve communication performance, optimal node selection and rank reordering in the job scripts can be employed. VPIC executions produce a significant amount of output data, hence techniques to optimize I/O performance, in general, might lead to improved performance.

WRF Improvements

While our WRF runs of 2012 had many ranks reading input data from multiple files, the recent runs employed a single input file. A natural I/O optimization is to turn on the multiple-reader scheme again. Similarly, rank reordering had been employed in 2012, and it can be adopted again for systems that provide such capability. This can lead to improved performance on WRF's nearest neighbor communication phase, which occurs on every time-step of the execution. In addition, improved compiler vectorization in the processing of vertical columns, where most of the physics in the code is computed, could result in better overall performance.

IV. Website for SPP Distribution

To promote wide usage of the SPP benchmarks in the HPC community, NCSA is making the codes available[1] from a general benchmarking page on its website, under the URL

https://bluewaters.ncsa.illinois.edu/benchmarks

This site contains links to a page specific to the SPP codes and inputs. From that page, one can download (i) the source codes for all SPP applications, (ii) build scripts for Blue Waters, (iii) usage instructions, and (iv) input data for both the compact and large cases. For some of the codes, the data must be retrieved using Globus Online. The page also lists our preliminary SPP results obtained on Blue Waters. We plan to keep updating this page as we optimize the codes and improve their observed performance.

The site above also has links to other kinds of test codes, such as I/O and low-level benchmarks. NCSA staff periodically run some of these tests, together with part of the SPP codes, as regression tests for Blue Waters. The executions of these tests are fully automated using an infrastructure, based on Jenkins [32], which can submit jobs to Blue Waters, process their outputs and archive their results for historical analyses.

V. Conclusion

High performance is the major motivation for employment of large-scale systems. There are several benefits in using benchmarks to assess the performance of existing systems, such as comparing competing offers, evaluating whether a deployed system achieves the performance level promised by the vendor, and verifying that software updates have not impacted system performance negatively. Those assessments can also guide vendors in the design of future systems. Traditional benchmarks have been used for some of these tasks, but since they typically focus on a limited set of aspects of the systems, they may fail in revealing the expected performance of a given system with real workloads.

For the Blue Waters acceptance in 2012, NCSA had used a collection of benchmarks consisting of actual scientific applications. That collection, named the Sustained Petascale Performance (SPP) benchmarks, covered a variety of important scientific areas expected to use Blue Waters. The performance of the system on those benchmarks was measured with a method called Sustained System Performance (SSP), which considers end-to-end application executions, including overheads such as I/O, checkpointing, and initialization phases. Using the SPP benchmarks, it was possible to determine that Blue Waters was capable of sustaining petascale performance on a wide set of applications that were relevant for its potential user community.

Some of the SPP benchmarks continue to be used regularly by NCSA staff to verify system behavior, ensuring that there is no performance regression after major software updates. Meanwhile, NCSA's detailed monitoring of the system has revealed that some of the original SPP applications had much more use than others, while new applications, outside the SPP set initially, emerged as popular applications among Blue Waters users. Hence, to keep the SPP suite relevant and representative of the actual Blue Waters' workload, NCSA is updating the suite by replacing some of its components. This update was guided by a detailed study of all jobs executed on Blue Waters for more than three years since its initial deployment.

In this paper, we have presented the components of the updated SPP benchmark suite, describing the major characteristics of the codes and the results of their execution on Blue Waters, with current directions being considered for

[1] NCSA has agreements with code owners to distribute the versions used in the SPP, even for the applications that are not open-source originally.

performance improvements. The major goal of this updated set is to provide a collection of applications, with appropriate input datasets, that represent the typical workload from the scientific community. This workload is likely to be dominant also on other modern large-scale systems aimed at supporting open science. Potential directions for performance improvements common to many of the SPP applications include compiler vectorization, for numerical sections, and careful placement of tasks, to promote better communication between executing nodes.

The paper also described NCSA's efforts to promote adoption of the SPP suite, by sharing the source codes of the applications, including input datasets and build scripts, with the scientific community, via a public website. For the future, NCSA envisions keeping this website updated with results observed with the SPP codes on other systems, and expanded with other codes as the SPP suite evolves with time. Jointly, the SPP suite and its corresponding website should become a valuable asset for the high-performance computing community, both from a user's standpoint, with performance results of a given code on different systems, and from a vendor's or system administrator's perspective, offering a one-stop location where several production-level codes from different scientific areas can be obtained.

Acknowledgment

We thank Jim Phillips, from the University of Illinois, for helpful comments and suggestions for improvement of our text. This work is part of the Blue Waters sustained-petascale computing project, which is supported by the US National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.

References

[1] W. T. Kramer, "PERCU: A Holistic Method for Evaluating High Performance Computing Systems," University of California at Berkeley, Berkeley, 2008.

[2] G. H. Bauer, T. Hoefler, W. T. Kramer and R. A. Fiedler, "Analyses and Modeling of Applications Used to Demonstrate Sustained Petascale Performance on Blue Waters," in *Annual Meeting of the Cray Users Group*, Stuttgart, 2012.

[3] J. Dongarra, "Performance of Various Computers Using Standard Linear Equations Software," University of Tennessee, Knoxville TN, 37996, 1985.

[4] J. Earl, C. G. Willard and D. Goldfarb, "IDC Workstations and High-Performance Systems Bulletin," in *HPC User Forum: First Annual Meeting Notes*, 2000.

[5] "MVAPICH Ping-Pong Benchmark," [Online]. Available: https://mvapich.cse.ohio-state.edu/svn/mpi-benchmarks/branches/OMB-3.1/osu_latency.c. [Accessed 2008].

[6] W. Gropp and E. L. Lusk, "Reproducible Measurements of MPI Performance Characteristics," in *6th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Barcelona, 1999.

[7] "Streams Benchmark," [Online]. Available: http://www.cs.virginia.edu/stream/. [Accessed 2008].

[8] M. A. Heroux and J. Dongarra, "Toward a New Metric for Ranking High Performance Systems," Sandia National Laboratory, Albuquerque, 2013.

[9] J. Dongarra and P. Luszczek, "HPC Challenge: Design, History, and Implementation Highlights," in *Contemporary High Performance Computing: From Petascale Toward Exascale*, Boca Raton, Taylor and Francis, 2013, pp. 13-30.

[10] F. McMahon, "The Livermore Fortran Kernels: A computer test of numerical performance range," University of California,, Livermore, CA, 1986.

[11] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan and S. Weeratunga, "The NAS Parallel Benchmarks," NASA Ames, Moffett Field, CA, March 1994.

[12] "SPEC Benchmarks," 2000. [Online]. Available: http://www.spec.org. [Accessed 2008].

[13] W. T. Kramer, "Keynote Talk: Top500 versus Sustained Performance – Or the Top Problems with the TOP500 List – And What to Do About Them," in *The 21st International Conference on Parallel Architectures and Compilation Techniques - PACT*, Minneapolis, 2012.

[14] P. J. Mucci, S. Browne, D. Christine and G. Ho, "PAPI: A portable interface for hardware performance counters," in *Proceedings of the Department of Defense HPCMP Users Group Conference*, 1999.

[15] C. L. Mendes, B. Bode, G. H. Bauer, J. R. Muggli, C. Beldica and W. T. Kramer, "Blue Waters Acceptance: Challenges and Accomplishments," in *Annual Meeting of the Cray Users Group*, Napa, CA, 2013.

[16] M. D. Jones, J. P. White, M. Innus, R. L. DeLeon, N. Simakov, J. T. Palmer, S. M. Gallo, T. R. Furlani, M. Showerman, R. Brunner, A. Kot, G. Bauer, B. Bode, J. Enos and W. Kramer, "Workload Analysis of Blue Waters," SUNY, Buffalo, NY, 2017.

[17] Y. Cui, K. B. Olsen, T. H. Jordan, K. Lee, J. Zhou, P. Small, D. Roten, G. Ely, D. K. Panda, A. Chourasia, J. Levesque, S. M. Day and P. Maechling, "Scalable Earthquake Simulation on Petascale Supercomputers," in *Proceedings of Supercomputing*, New Orleans, LA, 2010.

[18] T. Goodale, G. Allen, G. Lanferman, J. Massó, T. Radke, E. Seidel and J. Shalf, "The Cactus Framework and Toolkit: Design and Applications," in *Proceedings of the 5th International Conference on Vector and Parallel Processing-VECPAR*, Berlin, 2002.

[19] S. Gottlieb, "Benchmarking and tuning the MILC code on clusters and supercomputers," *Nuclear Physics B - Proceedings Supplements*, Vols. 106-107, pp. 1031-1033, 2002.

[20] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Christophe, R. D. Skeel, L. Kalé and K. Schulten, "Scalable molecular dynamics with NAMD,"

Journal of Computational Chemistry, vol. 26, pp. 1781-1802, 2005.

[21] M. Valiev, E. J. Bylaska, N. Govind, K. Kowalski , T. P. Straatsma, H. J. van Dam, D. Wang, J. JNieplocha, E. Apra, T. L. Windus and W. A. de Jong, "NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations," *Comput. Phys. Commun,* vol. 181, p. 1477, 2010.

[22] P. R. Woodward, "The PPM Compressible Gas Dynamics Scheme," *Implicit Large Eddy Simulation: Computing Turbulent Fluid Dynamics,* pp. 130-146, 2007.

[23] R. Fiedler, N. Wichmann, S. Whalen and D. Pekurovsky, "Improving the performance of the PSDNS pseudo-spectral turbulence application on Blue Waters using coarray Fortran and task placement," in *Proceedings of the Annual Meeting of the Cray Users Grou*, Napa, CA, 2013.

[24] J. Kim, K. Esler, J. McMinis and D. M. Ceperley, "Quantum Monte Carlo algorithms: making most of large-scale multi/many-core clusters," in *Conference Series - Scientific DIscovery throughAdvanced Computing (SciDac)*, Chattanooga, TN, 2010.

[25] S. Moore, E. Briggs, M. Hodak, W. Lu and J. Bernholc,, "Scaling the RMG Quantum Mechanics Code," in *Proceedings of the Extreme Scaling Workshop - BW-XSEDE*, Urbana, IL, 2012.

[26] K. J. Bowers, B. J. B.J. Albright, L. Yin, W. Daughton, V. Roytershteyn, B. Bergen and T. J. Kwan, "Advances in petascale kinetic simulations with VPIC and Roadrunner," *Journal of Physics: Conference Series,* vol. 180, 2009.

[27] J. Michalakes, J. Dudhia, T. Henderson, J. Klemp, W. Skamarock and W. Wang, "The Weather Reseach and Forecast Model: Software Architecture and Performance," in *Proceedings of the 11th ECMWF Workshop on the Use of High Performance Computing In Meteorology*, Reading, UK, 2004.

[28] H. A. Nam, "Upcoming Machines - The Path to Exascale," Los Alamos National Laboratory, Argonne, 2016.

[29] M. Bruggencate and D. Roweth, "DMAPP - an API for one-sided program models on Baker systems," in *Annual Meeting of the Cray Users Group*, Edinburgh, UK, 2010.

[30] J. Nieplocha, R. J. Harrison and R. J. Littlefield, "Global Arrays: A nonuniform memory access programming model for high-performance computers," *The Journal of Supercomputing,* vol. 10, no. 2, pp. 169-189, 1996.

[31] V. M. Anisimov, G. H. Bauer, K. Chadalavada, R. M. Olson, J. W. Glenski, W. T. Kramer, E. Aprà and K. Kowalski, "Optimization of the Coupled Cluster Implementation in NWChem on Petascale Parallel Architectures," *J. Chem. Theory Comput.,* vol. 10, no. 10, pp. 4307-4316, 2014.

[32] R. D. Budiardja, T. Bouvet and G. Arnold, "Application-Level Regression Testing Framework using Jenkins," in *Annual Meeting of the Cray Users Group*, Redmond, WA, 2017.