Application Performance in the Frontera Acceptance Process

Richard Todd Evans
rtevans@tacc.utexas.edu
Texas Advanced Computing Center, University of Texas at Austin
Austin, TX, USA

ABSTRACT

In 2017 the NSF called for proposals for a High Performance Computing System that would replace its largest system at the time, Blue Waters, located at the National Center for Supercomputing Applications. The solicitation required proposals to include a performance analysis of eleven designated scientific application benchmarks, chosen to be representative of the workload the proposed system was to support. The performance analysis was expected to project time-to-solution improvements for each benchmark on the proposed system over Blue Waters. The projected performance improvements were then required to be used as a metric for system acceptance. In this paper, we present a candid accounting of the development of the application acceptance criteria used in the Texas Advanced Computing Center's proposal to this solicitation. We also describe the acceptance testing process on the resulting Frontera system and compare its scientific workload capacity to Blue Waters.

CCS CONCEPTS

• **General and reference** → **Design**; **Performance**; • **Computer systems organization** → *Parallel Architectures.*

KEYWORDS

HPC, acceptance, deployment, performance, benchmarking

ACM Reference Format:
Richard Todd Evans. 2020. Application Performance in the Frontera Acceptance Process. In *HPCSYSPROS20 '20: SIGHPC Systems Professionals Workshop, November 13, 2020, Atlanta, GA.* ACM, New York, NY, USA, 9 pages. https://doi.org/TBD

1 INTRODUCTION

The US National Science Foundation (NSF) supports the nation's open-research computing demands through computing systems awarded to HPC Service Providers such as the Texas Advanced Computing Center (TACC) on a competitive basis. Individual computing systems within the infrastructure are intended to support a certain profile of workloads, ranging from systems focused on small interactive data analysis to systems excelling at large-scale, highly parallelized simulations. In order to demonstrate a system's design is appropriate for supporting a given workload, acceptance criteria indicating application performance are developed and written into a HPC Service Provider's proposal responding to the system solicitation. These application performance acceptance criteria are used as one of many inputs into the decision to award a given proposal. Once the awarded system has been deployed, it must then pass the proposed acceptance criteria before it can enter into its operations phase.

The NSF solicitation that ultimately resulted in Frontera specified a set of scientific application benchmarks whose projected performance on the proposed system was required to be in included in responding proposals and used as acceptance criteria [8, 10]. Proposed systems were expected to provide "*at least two- to three-fold time-to-solution performance improvement over the current state of the art, the University of Illinois at Urbana-Champaign's (UIUC) Blue Waters system . . .*". This performance improvement was to be demonstrated through the Sustained Petascale Performance (SPP) benchmark suite, which was specifically chosen to "*. . . represent characteristics of the current and possibly future workloads, which consists of solving complex scientific problems and diverse computation techniques at high degrees of parallelism.*" [7]. The suite consists of eleven applications run on between 1,296 to 8,448 of Blue Waters 22,640 CPU-only XE nodes. In addition to these benchmarks, proposals were encouraged to include additional applications that are representative of emerging workloads in HPC environments. For this purpose, TACC added a machine learning benchmark, the performance of ResNet-50 on the Imagenet 100-category dataset implemented in the Caffe framework (version 1.0.3)[2, 4]. The 12 applications and general area of scientific discipline are shown in Table 1.

Our development of the application performance acceptance criteria was driven by two competing concerns. On the one hand, because the criteria must be met before the system could go into production, aggressive performance projections could be disastrous. On the other hand, being overly conservative would underestimate the capability of the proposed machine and make the proposal less competitive. Compounding these concerns, proposals and their application acceptance criteria were made one-and-a-half years in advance of system acquisition, and only estimates for the specifications of the actual hardware were known. In the case of Frontera, we used a cautious approach using lower bounds for estimated hardware capabilities combined with characterizing how these hardware capabilities influence application benchmark performance.

2 BENCHMARK CHARACTERIZATION

Over a time period of roughly 6 weeks in the Fall of 2017, TACC studied the SPP benchmark suite (plus the added Caffe benchmark) on the Intel Xeon 8160 ("Skylake") partition of Stampede2 as it was being deployed. This partition consists of 1,736 48-core nodes

HPCSYSPROS '20, November 13, 2020, Atlanta, GA

ACM ISBN TBD. . . $TBD
https://doi.org/TBD

Application	Discipline
AWP	Seismic
CACTUS	Relativistic Astrophysics
MILC	Particle Physics Lattice QCD
NAMD	Molecular Dynamics
NWChem	Chemistry
PPM	Astrophysics
PSDNS	Fluid Dynamics
QMCPACK	Quantum Chemistry
RMG	Electronic Structures
VPIC	Plasma Physics
WRF	Weather
Caffe	Machine Learning

Table 1: Applications used in acceptance benchmarks.

connected with Intel's 100 Gb/s Omni-Path fabric [3]. Each benchmark was run on a different range of node counts to characterize scaling behavior, although all of them were run on at least 1,024 nodes. Compiler versions and optimization flags were tested for each benchmark, including determining which vectorization level was most performant - scalar, 128b, 256b, or 512b. MPI task and OpenMP thread counts per node were also varied for each application until a runtime configuration that worked at all node counts and was reasonably performant was found. Compiler versions, MPI versions, and optimization flags used in this study can be located using the information in Section A.3.

In three benchmarks we used a different version of the application than what was specified in the SPP suite to develop the acceptance criteria:

(1) The SPP version of RMG, 2.0.0, required a Cray environment. We used version 2.2.1 to avoid this limitation.
(2) We used a newer version of NAMD (2.13 versus 2.12) because the newer version was able to utilize 512b-wide vector instructions.
(3) The SPP version of WRF, 3.3.1, was replaced by WRF 3.6.1 due to the latter's more efficient use of OpenMP.

We also had to modify the VPIC source to use *Isend* rather than *Issend*, as the latter routine was not compatible with Intel MPI at the time of this study.

Performance data for the CPU, memory subsystem, communications and scaling, and IO was then collected for each benchmark at multiple node counts using their most performant binaries at constant MPI task and thread counts per node. We only collected time-averaged performance metrics in order to maintain low overhead, and therefore do not capture transient behaviors that could impact performance. We did, however, observe these benchmarks to have fairly constant workloads in time.

CPU performance data was collected using the `perf` subsystem in the Linux kernel. The memory subsystem data was collected using a TACC-built binary that samples the Integrated Memory Controller counters directly from PCI configuration space. We collected any communication data using the `I_MPI_STATS` mechanism available in Intel MPI.

This study was performed on a system undergoing deployment, with fluctuating node counts available, over a relatively short time frame. We don't claim it was comprehensive, but describe how it was used to motivate and justify our architecture decisions. We note below where we had incomplete or inconsistent data. The Caffe benchmark did not undergo the same analysis as the other benchmarks. Its analysis is described in Section 2.6.

2.1 CPU Data

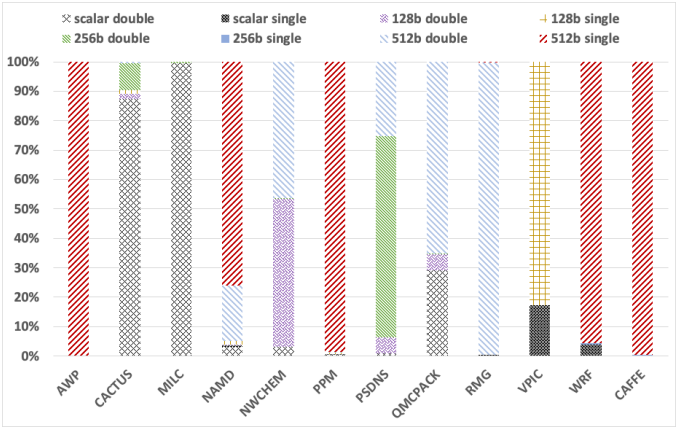

Figure 1: Percentage of flop instruction-type by benchmark.

Application	Min/Max Nodes	Min/Max $\frac{GFLOPS}{node}$	% Change
AWP	128/1366	130.7/134.5	3%
Cactus	1000/1700	8.9/6.3	-34%
MILC	128/1296	41.5/29.3	-34%
NAMD	512/1600	180.8/169.8	-6%
NWChem	800/1024	149.8/156.8	5%
PPM	704/1408	460.9/436.2	-5%
PSDNS	820/1640	35.6/26.4	-30%
QMCPACK	400/1600	115.2/88.1	-27%
RMG	576/1576	1008.3/517.4	-64%
VPIC	576/1536	219.1/213.1	-3%
WRF	400/1680	113.5/101.8	-11%

Table 2: Average GLOPS per node at minimum and maximum node counts.

Stampede2's Skylake nodes are capable of supporting single and double precision scalar and up to 512b-wide SIMD (vectorized) instructions. We measured the total number of single and double precision scalar, 128b-, 256b-, and 512b-wide vectorized floating point instructions used to execute each benchmark. This allowed us to determine the importance of vectorized instructions to each benchmark's performance along with the maximum possible per-node-performance of each benchmark. These instruction measurements can be seen in Fig. 1 in terms of percentage of each floating point instruction-type used by each application. Seven of the twelve benchmarks' floating point instructions were composed of at least 65% 512b-wide vector instructions. We concluded that 512b-wide

vector instructions were important for many of the benchmarks' performances.

We also computed the per-node-performance in terms of flop rates (*GFLOPS/node*) on every benchmark run. Table 2 shows the per-node-performance for each benchmark at the minimum and maximum node counts at which we performed measurements, along with the percent change. We drew three main conclusions from these data

(1) No benchmark run was CPU bound. A Skylake node is capable of roughly 2200 double precision *GFLOPS/node* with a well-tuned HPL binary [5]. Our highest measured per-node-performance is from the 576 node RMG run at 1008 *GFLOPS/node*. RMG, dominated by 512b double precision flops is well under the 2200 *GFLOPS/node* available to it. The benchmarks that are dominated by scalar instructions - Cactus and MILC - are only able to use roughly 1/8 the flops available to HPL, or 275 *GFLOPS/node*. These benchmarks are well under that limit as well.
(2) Only two benchmarks improve in per-node-performance with increasing node count, AWP at 3% and NWChem at 5%.
(3) Decreases in per-node-performance with increasing node count are due to a combination of increasing communication overhead and decreasing levels of cpu- and memory-level parallelism per node. Five applications experienced significant decreases (>11%) in per-node-performance when scaling from the minimum to maximum node counts.

2.2 Memory Subsystem Data

Application	Min/Max Nodes	Min/Max MBW [GB/s]
AWP	128/1366	$141^{+15.6}_{-1.6}/126^{+13.2}_{-9.2}$
Cactus	NA	NA
MILC	128/1296	$167^{+0.5}_{-0.5}/63^{+4.2}_{-1.5}$
NAMD	2500	46
NWChem	1536	43
PPM	NA	NA
PSDNS	820/1640	$89^{+2.1}_{-5.3}/62^{+2.0}_{-4.4}$
QMCPACK	400/1200	$75^{+2.8}_{-1.5}/49^{+3.3}_{-2.8}$
RMG	NA	NA
VPIC	576/1536	$96^{+1.3}_{-0.3}/90^{+0.44}_{-0.80}$
WRF	400/1680	$138^{+0.7}_{-1.8}/96^{+1.3}_{-2.2}$

Table 3: Average memory bandwidth per node at minimum and maximum node counts.

We collected the average memory bandwidth generated per node only for application benchmarks we expected to be sensitive to memory bandwidth, neglecting the likely insensitive Cactus, NAMD, NWChem, PPM, and RMG benchmarks from the measurements due to time limitations. The average memory bandwidth generated per node for the other six benchmarks is shown in Table 3 at the minimum and maximum node counts at which it was measured. We noticed some significant variation in the memory bandwidth across the nodes used within a run and included the upper and lower bounds for observed bandwidth. At the maximum node counts, the memory bandwidth variation across nodes was still within 10% of the average node's bandwidth for every benchmark - there were no extreme imbalances. The bandwidth numbers for NAMD and NWChem, shown in blue, were later measured on Frontera as a verification of our assumptions that they were bandwidth insensitive. We never measured the Cactus, PPM, and RMG memory bandwidths at scale.

Only one of these benchmark runs - MILC at 128 nodes - utilized greater than 80% of the available ~190 GB/s Skylake *sustained memory bandwidth* (memory bandwidth as measured by STREAM). The MILC benchmark, however, used just 33% at the maximum node count it was run on. This decrease in memory bandwidth usage with increasing node count is observed for every benchmark. It is due to the fixed benchmark problem size being partitioned across more nodes (i.e. strong scaling). With strong scaling, the per-node-memory footprint decreases and the amount of data that must be moved from memory to CPU either stays the same or, more likely, decreases. The reduction in memory bandwidth usage is then due to a combination of greater cache availability relative to the memory footprint, decreases in cpu- and memory-level parallelism, and communications overhead.

Assuming DDR3 memory has a similar memory bandwidth-latency curve to Stampede2's DDR4 memory, benchmark performance may be at most exponentially dependent on available bandwidth when using greater than 80%, linearly dependent when using 40%-80%, and constant when using less than 40% of the maximum sustained memory bandwidth [9]. We concluded from this loose constraint and measured data that using a processor with more memory bandwidth than the Skylake would not drastically improve the performance of these application benchmarks at the node counts we would eventually project them to run on. Only AWP, VPIC, and WRF could experience up-to linear improvement in performance with increasing available memory bandwidth at the maximum node counts they were measured on Stampede2. We note that if the benchmark problem sizes were larger, the associated increases in per-node-memory footprint could have changed the observed behavior and our conclusions.

2.3 Roofline Data

In this subsection we illustrate the scaling behavior of the six benchmarks where both CPU and memory subsystem data are available with a Roofline plot. In Fig. 2 we display the *GFLOPS/node* versus the arithmetic intensity, $flops/byte$, of the six applications. The light-blue line with a discontinuity around 1.4 $flops/byte$ bounds the maximum per-node performance based on the arithmetic intensity of the benchmarks, assuming only scalar floating-point instructions are used (as stated in Section 2.1 the total flops available to an application depends on its mix of floating-point instruction types). The data measured from maximum node count runs for each benchmark are plotted with crosses in Fig. 2, while the data from minimum node count runs are small circles connected to those crosses by a line of the same color. The single measurements for NAMD and NWCHEM on Frontera are also shown and plotted by lone crosses.

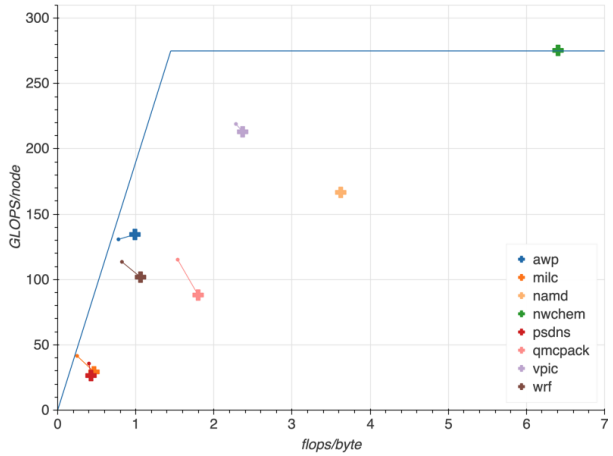

Figure 2: Roofline plot for benchmarks at minimum and maximum node counts measured

The Roofline Model quantifies how well a benchmark on a given hardware configuration is utilizing the flops and memory bandwidth available to it based on its arithmetic intensity and floating-point instruction mix. It does not give insight to the causes of sub-maximal usage, whether it's lack of cpu- and memory-level parallelism or communications overhead. In our study, AWP, MILC, and PSDNS are utilizing within 10% of the memory bandwidth-limited flops at their minimum node counts, with MILC using within 1%. This is in qualitative accordance with what we observed in Section 2.2, where MILC at 128 nodes is the only benchmark run using >80% of the memory bandwidth per node, and therefore only its performance is potentially strongly limited by memory bandwidth. MILC's arithmetic intensity at the maximum node count is 87% greater than at the minimum, likely the cause of its dramatic shift from memory bandwidth sensitive to insensitive regimes.

The six benchmarks that we had measurements for can be seen to increase in arithmetic intensity with strong scaling. This increase is potentially responsible for the decreasing memory bandwidth utilization with increasing node count and could result in improved per-node-performance due to improved cache reuse. However, only AWP experiences even a small improvement. The decrease in other benchmarks' per-node-performance is likely because this caching effect is offset by losses in cpu- and memory-level parallelism and communication overhead. We note decreasing the available cache per node without a corresponding increase in memory bandwidth could be disastrous for the performance of AWP, WRF, MILC, and PSDNS, even at large node counts, as their arithmetic intensity could decrease enough that they become strongly memory bandwidth limited.

2.4 Communications and Scaling Data

We characterized the communication patterns of some of the benchmarks by collecting the percentage of time spent in each participating MPI routine at various scales. Other benchmarks were characterized by prior knowledge or inspection of the application. Our qualitative characterization of each benchmark's communication pattern is summarized in Table 4. Our quantitative MPI data has unfortunately been lost to staff turnover and time. We also note in this table which benchmarks were able to efficiently use OpenMP, as applications that efficiently use OpenMP can require fewer MPI tasks and hence have lower communication overhead.

Application	Communication Characterization
AWP	point-to-point
Cactus	efficient OpenMP, large message collectives
MILC	frequent 8B allreduce
NAMD	infrequent small message alltoall
NWChem	global arrays, large message collectives
PPM	point-to-point
PSDNS	frequent small message alltoall
QMCPACK	efficient OpenMP, infrequent allreduce
RMG	efficient OpenMP, large message collectives
VPIC	point-to-point
WRF	efficient OpenMP, point-to-point
Caffe	infrequent allreduce

Table 4: Communication characteristics by benchmark.

As previously stated, we ran each benchmark on a different range of node counts. Node counts and runtimes for every benchmark run can be seen in the scaling plots of Fig. 3, where the measured data points are red dots and the x- and y-axes are node counts and runtimes respectively. The node counts we used for each benchmark were based on

- How flexible the application decomposition was. Some benchmarks required specific task counts on specific node counts to ensure efficient partitioning of the problem.
- How much memory the benchmark problem required. All of the benchmarks had a large enough memory footprint that they could not be run under a certain node count.
- How stable the application benchmark was. NWChem hung at node counts greater than 1,024 on Stampede2.
- How close we were to achieving a three-fold improvement over Blue Waters for the application on available node counts. The AWP, MILC, and RMG benchmarks were able to achieve a three-fold time-to-solution improvement over Blue Waters on Stampede2.

In Fig. 3 it can be seen that the runtimes for every benchmark consistently decreased with increasing node count. Some benchmarks scaled much more efficiently than others as shown in the per-node-performance numbers in Table 2.

Our examination of the flops and memory bandwidths for each benchmark at different node counts gave us confidence that the benchmarks were run in node ranges where no large effects due to caching or a sudden removal of a significant bottleneck, such as memory bandwidth, would appear (with the possible exception of the 128 node MILC run). This suggested it was reasonable to use Amdahl's law to model the strong scaling behavior of each benchmark,

$$T = T_0(1-p) + \frac{T_0 p}{N},$$

where T is the runtime, T_0 is the runtime on a single node, p is the fraction of the workload that is parallelizable over nodes, and N is

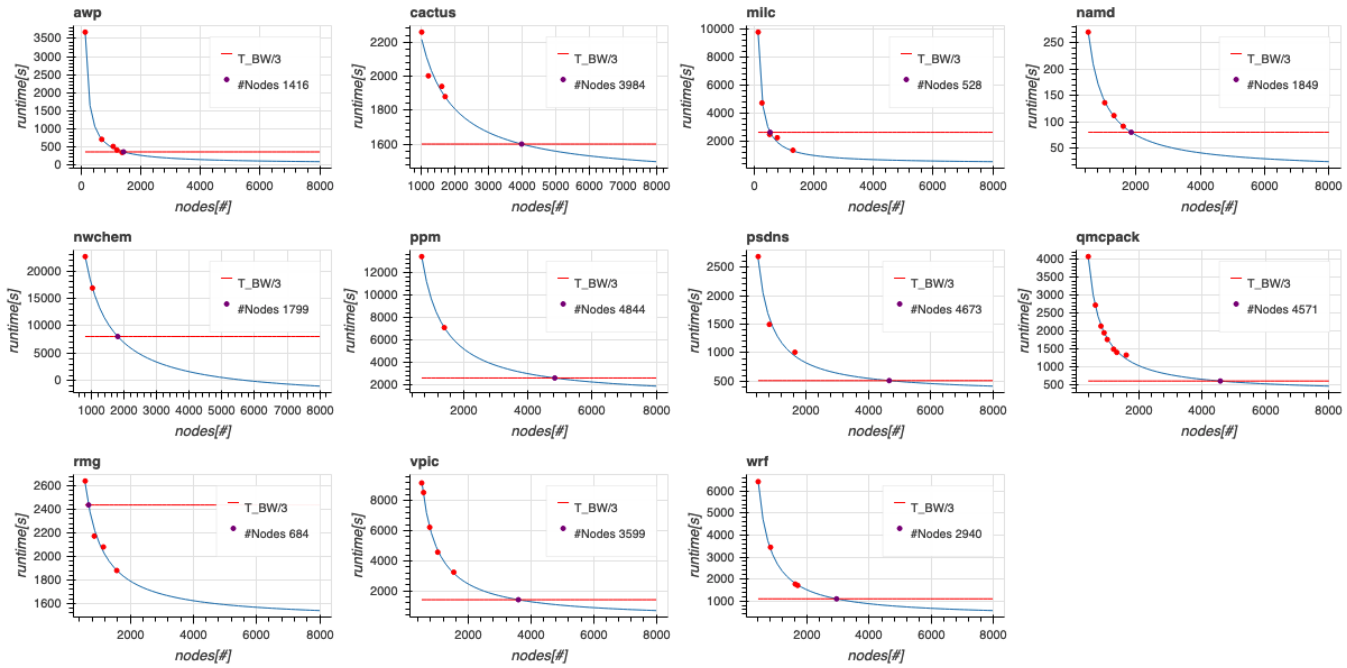

Figure 3: Scaling plots, model fits, and extrapolations by benchmark.

the variable node count. The blue line shown in each plot of Fig. 3 is the curve fit of Amdahl's law to the runtimes for each benchmark, where we determined two parameters in the fit: $A = T_0(1-p)$ and $B = T_0 p$. We expected this model to adequately capture the strong scaling behavior until the increases in communication overhead grew significant relative to the on-node processing time. We did not observe this adverse effect in any of the benchmarks at the node counts measured on Stampede2.

Although we did not quantitatively measure the goodness of fit for each curve, the curves can be seen to qualitatively fit the data reasonably well for each benchmark. The only suspicious scaling curve was that for NWChem, where the only two points available scale superlinearly. The effect of this is that the curve indicates a negative runtime at very high node counts. We expect additional data points would have corrected this, however we did not have time to investigate.

We used the fit parameters, A and B, to interpolate or extrapolate each benchmark's strong scaling curve to the number of nodes that would be required to achieve a three-fold reduction in runtime over Blue Waters. The purple dots at the intersection of the red lines and blue curves in Fig. 3 indicate at what node count this would be achieved. The further the extrapolation required to achieve the three-fold performance improvement, the greater the opportunity for communication overhead on Frontera to grow large enough to overwhelm any performance gains made by further parallelization.

2.5 Filesystem IO Data

The WRF and PSDNS benchmarks had IO phases that occupied a significant fraction of their runtime. These phases were observed to scale linearly with the filesystem bandwidth available to the application. The IO time can be attributed to the serial term, $A = T_0(1-p)$, in our model.

2.6 Caffe Projection

The Caffe benchmark performance on Blue Waters was estimated by measuring how many images per second could be performed on a single XE node, and then assuming perfect scaling to 1,024 nodes (i.e. multipling single-node performance by 1,024) to obtain a total number of images per second performance. This total number of images per second was then used to compute the total time required to complete the benchmark on Blue Waters on 1,024 nodes. We then ran the Caffe benchmark on 1,024 Stampede2 nodes and used that runtime for our Frontera projection.

3 ACCEPTANCE CRITERIA

Our acceptance criteria consisted of a specified runtime and node count at which that runtime would be achieved for each benchmark. If the measured runtime on the new system was achieved with fewer nodes than projected that was considered acceptable. If the measured runtime on the new system was lower than the projected runtime that was also considered acceptable.

Based on our benchmark study described in the previous section, we determined that a system consisting entirely of Intel Xeon 8160 Skylake nodes with an Omni-Path interconnect would be able to achieve a two- to three-fold performance improvement for each application at less than or equal to the number of nodes used on Blue Waters and fit within the solicitation's budget. This allowed us to make conservative proposals for the acceptance criteria based on the assumptions that we could deploy an Intel Xeon processor-based system where the processor would have at least as many

Application	BW Node[#]	BW Runtime[s]	Frontera Node[#]	Frontera Runtime[s]	$\frac{BW\ Runtime}{Frontera\ Runtime}$
AWP	2048	1228	1336	335	3
CACTUS	4096	4800	2400	1753	2.7
MILC	1296	7916	1296	1364	5.8
NAMD	4500	242	2500	62	3.9
NWChem	5000	24160	5000	8053	3
PPM	8448	7790	5000	2540	3
PSDNS	8192	1538	3235	769	2
QMCPACK	5000	1832	2500	916	2
RMG	3456	7310	700	2410	3
VPIC	4608	4218	4608	1170	3.6
WRF	4560	3289	4600	749	4.4
Caffe	1024	3310	1024	1203	2.75

Table 5: Frontera's application acceptance criteria by benchmark.

flops, as much cache, and as much memory bandwidth per node as Stampede2's Skylake nodes and have an interconnect at least as capable as Omni-Path. At the time we developed the acceptance criteria, we were expecting Frontera to be composed of nodes with ~10% more flops and memory bandwidth and at least as much cache as the Stampede2 nodes. We also expected Frontera's interconnect to be roughly comparable to Omni-Path in terms of performance.

We had met or exceeded a three-fold improvement over Blue Waters on Stampede2 for the AWP, MILC, and RMG benchmarks and therefore simply used our measured or interpolated runtimes and node counts as the acceptance criteria for these benchmarks. We then aggressively set our acceptance criteria to greater than a three-fold improvement for three benchmarks - NAMD, VPIC, and WRF. We were confident based on our data that these would scale well on Frontera. We conservatively extrapolated to fewer nodes than required for a three-fold improvement for three benchmarks - Cactus, PDSNS, and QMCPACK - where we were not confident about the scaling efficiency and would have had to extrapolate long node distances to achieve a three-fold improvement. PPM's point-to-point dominated communication and scaling data were encouraging; however, we had just two data points so only extrapolated to a three-fold improvement. In the case of NWChem, where the scaling data was nonsensical, we simply gave ourselves a large buffer in terms of acceptance criteria. In NWChem's case we projected the same number of nodes as was used on Blue Waters, far more than we expected to be required.

The projected node counts and run times we used for acceptance criteria are shown in columns 4 and 5 of Table 5. All runtimes used as acceptance criteria were projected to be run at node counts less than used on Blue Waters. The node counts and runtimes measured on Blue Waters along with the projected speedup on Frontera are also shown.

4 ACCEPTANCE TESTING AND RESULTS

From May through August 2019, TACC studied and measured the SPP suite's performance on Frontera's 8,008 Intel Xeon 8280 ("Cascade Lake") 56-core nodes as the system was deployed. SPP benchmarks requiring little communication - QMCPACK and Caffe - or predominantly point-to-point communication - AWP, PPM, VPIC, and WRF - required minimal MPI tuning and were able to run with reasonable performance with multiple MPI implementations from the start. Application benchmarks relying on frequent collective communications - CACTUS, MILC, NAMD, NWChem, PSDNS, and RMG - required exploration and tuning of the available MPI implementations to run to completion. Their performance was initially very poor - worse than on Stampede2. These difficulties with collectives arose because Frontera was the first large-scale deployment of Mellanox's Infiniband HDR interconnect, and no MPI implementation had yet been tuned for HDR at scale [6]. While the nominal performance of HDR (100 Gb/s and <1μs single-hop latencies for Frontera's HDR-100 HCAs) is comparable to Infiniband EDR and OmniPath, HDR has support for the extended Infiniband `verbs` API and expects to use new connection types such as *dynamic connection*.

TACC investigated each collective-intensive benchmark's performance using up to three different, evolving MPI implementations: Intel MPI, MVAPICH2-X, and Mellanox's HPC-X. The initially available Intel MPI versions would crash or hang for collective-intensive applications on more than 30 nodes. TACC worked with the Intel MPI developers to get an updated engineering release of Intel MPI 19 that was able to run most of the benchmarks effectively at scale with additional manually tuning of some of the collective algorithms used for specific message sizes. These changes were incorporated into later versions of Intel MPI. Before a reasonably stable Intel MPI was available though, we were able to meet some of our application benchmark acceptance criteria by working with the MVAPICH2 team to tune MVAPICH2-X. Their work was specifically critical to running NWChem and WRF in a performant and stable manner. We were never able to run HPC-X in a stable manner on our software stack and its tendency to leave running processes after its termination resulted in disfavoring its use.

In Table 6 we display the ratios for the projected to actual runtimes and node counts for each benchmark. The Runtime Ratio is the projected runtime (shown in column 5 of Table 5) over the measured runtime on Frontera. The Node Ratio is the number of nodes projected to be required (shown in column 4 of Table 5) over the number of nodes which were used on Frontera. The runtime and

Application	Runtime Ratio	Node Ratio	Speedup vs. BW
AWP	1.03	1	3.2
CACTUS	1.22	1	3.3
MILC	1.64	1	9.5
NAMD	1.03	1	4.0
NWChem	1.26	3.26	3.8
PPM	1.17	1.04	3.6
PSDNS	1.41	1.58	2.8
QMCPACK	2.76	1	5.5
RMG	1.04	1.02	3.2
VPIC	1.19	1.13	4.3
WRF	1.18	1.09	5.2
Caffe	1.15	1	3.2

Table 6: Ratios of projected to measured runtimes and node counts on Frontera and speedup versus Blue Waters (BW).

node ratios must be greater than or equal to 1 for the acceptance criteria to be unequivocally met.

All of our acceptance criteria were met, with 9 out of 12 of the benchmarks running within 30% of our projected runtime and 10 out of 12 using within 15% of our projected node counts. The 30% or less runtime discrepancies can be attributed to a combination of architectural differences between the Stampede2 and Frontera systems including: 50% more flops available per node, 10% more memory bandwidth available per node, and improved interconnect performance. We did in fact only expect a 10% increase in Frontera's per-node flop capability versus Stampede2's, but received an additional 35% lift arising from a higher nominal frequency (2.7 GHz) than expected (2.1 GHz).

In the cases of the three benchmarks which exceeded our projected runtimes by more than 30% - MILC, PSDNS, and QMCPACK - we believe we've identified the majority of factors contributing to the discrepancies in projected versus measured performance:

- The MILC benchmark runtime is dominated by 8B MPI Allreduce collectives. We observed a 30% improvement to this MPI collective's performance at large node counts with fully subscribed nodes on Frontera's HDR interconnect compared with Stampede2's Omnipath interconnect. Accounting for this network improvement our overestimate would have been 1.15.
- I/O was 40% of the PSDNS runtime on a filesystem capable of 60GB/s such as Frontera's `SCRATCH1` Lustre filesystem. Moving this application benchmark to Frontera's DDN IME flash-based filesystem reduced IO to 10% of the runtime [1]. If this had been taken into account our overestimate would have been 1.2.
- QMCPACK's data structures were refactored between the development of our projections using the original SPP benchmark version (3.0.0), and our Frontera measurements using the latest version (3.7.0). The latest version enabled vectorization and more efficient memory access patterns. A factor of 2 improvement in performance was reported by the developers due to this change. Accounting for this our overestimate is 1.4.

The two applications which required more than 15% fewer nodes than we proposed were PSDNS and NWChem.

- PSDNS performance was extrapolated to a node count where it was expected to obtain two-fold performance improvement. We did not understand when making our projections that PSDNS communication is much more efficient when run with task and node counts of powers of 2. We used the more efficient task and node counts for the acceptance testing.
- As stated in Section 3, the NWChem benchmark's node requirements were set artificially high since we did not have confidence in its scaling model. In fact, while our model broke down at high node counts, it turned out to be reasonably accurate for the modest extrapolation in node count that was required. 1536 nodes were ultimately required compared to the 1800 nodes indicated by the model.

5 BLUE WATERS VS. FRONTERA CAPACITY

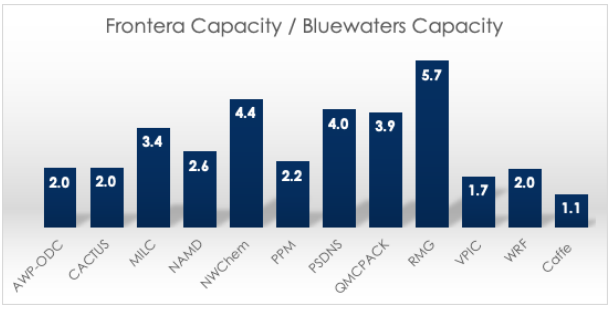

Figure 4: Frontera to Blue Waters capacity by benchmark.

Frontera was measured to provide at least a two- to three-fold reduction in runtime for each individual benchmark and thus satisfies the NSF solicitation's original performance expectations. It is, however, worth comparing the capacity of Frontera to Blue Waters, as Frontera is its replacement. We assume the SPP benchmark suite can be used as a proxy for these workloads given that was the purpose behind its design.

The capacity for each system for each benchmark a in runs per hour is

$$capacity_a = \frac{total\ nodes}{runtime_a \times node\ count_a}$$

where the *total nodes* is the total number of nodes for each system - 8,008 for Frontera and 22,640 (only the XE nodes are included) for Blue Waters - the $runtime_a$ is the runtime in hours for benchmark a, and $node\ count_a$ is the number of nodes on which $runtime_a$ was obtained. The ratio of Frontera's capacity to Blue Waters' capacity by application is shown in Fig. 4. The individual benchmark $runtime_a$s and $node\ count_a$s can then be combined to produce a *SPP capacity* for the entire benchmark suite

$$SPP\ capacity = \frac{total\ nodes}{\sum_a runtime_a \times node\ count_a}$$

which measures how much throughput of the entire SPP benchmark suite can be performed by a given system in an hour.

The *SPP capacity* of each system and their ratio is shown in Table 7. We also show the *SPP per node* capacity in this table. We note that while Frontera has only three times the capacity of Blue Waters, it provides this with nearly one-third the node count, resulting in

	Blue Waters	Frontera	$\frac{Frontera}{BlueWaters}$
SPP Capacity	0.27	0.81	3
SPP per node	1.2e-5	1.01e-4	8.5
HPL	7.1 PF	23.5 PF	3.3

Table 7: Capacity of Blue Waters versus Frontera.

an almost nine-fold improvement of throughput per node. The last row shows High Performance Linpack (HPL) performance of the XE partition of Blue Waters (using its theoretical peak performance as HPL performance was never reported for Blue Waters) and what was measured on Frontera. Frontera provides about a factor of 3 increase in HPL performance, in accordance with its capacity increase.

6 SUMMARY

We reviewed the study that drove the development of the application performance acceptance criteria for Frontera. We described the acceptance testing and results, where all acceptance criteria were met or exceeded. In 9 out 12 of our benchmarks, our projections were within 30% of our measurements on Frontera. Probable causes of these relatively small discrepancies and the larger discrepancies for the other 3 benchmarks are discussed. Overall capacity of Blue Waters and Frontera are then compared. Using the SPP application benchmarks as a proxy for the scientific workloads Frontera is expected to support, Frontera has three times the scientific capacity of Blue Waters at one-third the node count.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation through the ACI-1134872 Stampede, OAC-1540931 Stampede2, ACI-1953575 XSEDE, and OAC-1854828 Frontera awards. Special thanks goes to the TACC staff members that contributed to this effort: John Cazes, Kevin Chen, Victor Eijkhout, Lei Huang, Lars Koesterke, Hang Liu, Si Liu, Cyrus Proctor, Ian Wang, and Zhao Zhang

A ARTIFACT DESCRIPTION APPENDIX: [APPLICATION PERFORMANCE IN THE FRONTERA ACCEPTANCE PROCESS]

A.1 Abstract

This appendix describes how to use the software included in the artifact to construct the performance projections described in this paper. The artifact includes application source code, build and run scripts, and benchmark inputs and outputs. It also includes the code used to perform the fits to the performance measurements and extrapolations to our performance projections.

A.2 Description

A.2.1 Check-list (artifact meta information).

- **Program:** scientific application benchmarks from the Blue Waters Sustained Petascale (SPP) Benchmark Suite, SPP_Scaling.ipynb
- **Compilation:** scripts used to compile SPP benchmark applications
- **Data set:** inputs used in SPP benchmark applications
- **Run-time environment:** CentOS 7.3 (kernel 3.10.0-513), CentOS 7.4 (kernel 3.10.0-693)
- **Hardware:** 1,736 dual socket servers with Intel Xeon Scalable Processor 8160 connected by Intel Omni-Path. 8,008 dual socket servers with Intel Xeon Scalable Processor 8280 connected by Mellanox HDR Infiniband
- **Run-time state:** Multi-user mode (runlevel=3)
- **Execution:** Node counts with MPI and OpenMP tasks specified in job submission scripts
- **Output:** Timing output is specified in application README files, raw data output is available upon request
- **Experiment workflow:** Specified by general README, application READMEs, and job submission scripts
- **Experiment customization:** Node counts, MPI and OpenMP task configuration, and compiler optimizations can be modified
- **Publicly available?:** Intel Compiler License is required to compile with Intel Compilers, Globus Online account is required to download the SPP benchmarks and inputs, PPM source code requires acknowledging and agreeing to certain conditions and must be directly obtained from the SPP website

A.2.2 How software can be obtained . Applications, inputs, build scripts, run scripts and scaling software are included as artifacts to this paper.

A.2.3 Hardware dependencies. Intel CPU based systems with a high speed interconnect are required to reproduce results. A minimum of 512 nodes with 192 GB of memory per node are required to run every benchmarks.

A.2.4 Software dependencies. Software dependencies for each application are described in the README file for each benchmark. A SLURM job scheduler is expected to be used for benchmark submission. Python 3 with numpy, scipy, bokeh, and jupyter packages are required to perform the scaling curve fits and extrapolations.

A.2.5 Datasets. Input data sets are included as artifacts to this paper. Their use is indicated in each application's README file.

A.3 Installation

Compilation is described in each application's README file. Build scripts are provided where appropriate.

A.4 Experiment workflow

Runtime configurations are included as job scripts in the artifacts. Location and use of job scripts are described in each application's README file.

A.5 Evaluation and expected result

Timing outputs from runs on Stampede2 are included in the artifacts. Their location and interpretation are provided in each application's README.

A.6 Experiment customization

Job scripts can be modified to use different node and task configurations. Build scripts can be modified to use different compiler options.

REFERENCES

[1] DataDirect Networks. 2018. Overcoming Flash Performance Barriers At-Scale. https://www.ddn.com/download/ddn-ime-overcoming-flash-performance-barriers-at-scale/?wpdmdl=41002&refresh=5f53c185165e71599324549.

[2] ImageNet. 2012. ImageNet Large Scale Visual Recognition Challenge 2020. http://www.image-net.org/challenges/LSVRC/2012/, Last accessed on 2020-08-30.

[3] Intel. 2017. Intel Omni-Path Architecture Customer Resource Center: Overview. https://www.intel.com/content/www/us/en/design/products-and-solutions/networking-and-io/fabric-products/omni-path/overview.html, Last accessed on 2020-08-30.

[4] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093* (2014).

[5] John D. McCalpin. 2018. HPL and DGEMM Performance Variability on the Xeon Platinum 8160 Processor. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis* (Dallas, Texas) *(SC '18)*. IEEE Press, Article 18, 13 pages.

[6] Mellanox. 2019. Introducing 200G HDR Infiniband Solutions. https://www.mellanox.com/pdf/whitepapers/WP_Introducing_200G_HDR_InfiniBand_Solutions.pdf, Last accessed on 2020-08-30.

[7] National Center for Supercomputing Applications. 2017. SPP-2017 Benchmark Codes and Inputs. https://bluewaters.ncsa.illinois.edu/spp-benchmarks, Last accessed on 2020-08-30.

[8] National Science Foundation. 2017. Towards a Leadership-Class Computing Facility - Phase 1. https://www.nsf.gov/pubs/2017/nsf17558/nsf17558.htm, Last accessed on 2020-08-30.

[9] M. Radulovic, D. Zivanovic, D. Ruiz, B. Supinski, S. McKee, P. Radojkovic, and E. Ayguadé. 2015. Another Trip to the Wall: How Much Will Stacked DRAM Benefit HPC?. In *MEMSYS '15*.

[10] Dan Stanzione, John West, R. Todd Evans, Tommy Minyard, Omar Ghattas, and Dhabaleswar K. Panda. 2020. Frontera: The Evolution of Leadership Computing at the National Science Foundation. In *Practice and Experience in Advanced Research Computing* (Portland, OR, USA) *(PEARC '20)*. Association for Computing Machinery, New York, NY, USA, 106?111. https://doi.org/10.1145/3311790.3396656