Experiences Evaluating Functionality and Performance of IBM POWER8+ Systems

Verónica G. Vergara Larrea(✉), Wayne Joubert, Mark Berrill, Swen Boehm, Arnold Tharrington, Wael R. Elwasif, and Don Maxwell

Oak Ridge National Laboratory, Oak Ridge, TN, USA
{vergaravg,joubert,berrillma,boehms,arnoldt,elwasifwr,maxwellde}@ornl.gov

Abstract. In preparation for Summit, Oak Ridge National Laboratory's next generation supercomputer, two IBM Power-based systems were deployed in late 2016 at the Oak Ridge Leadership Computing Facility (OLCF). This paper presents a detailed description of the acceptance of the first IBM Power-based early access systems installed at the OLCF. The two systems, Summitdev and Tundra, contain IBM POWER8+ processors with NVIDIA Pascal GPUs and were acquired to provide researchers with a platform to optimize codes for the Power architecture. In addition, this paper presents early functional and performance results obtained on Summitdev with the latest software stack available.

1 Introduction

The Collaboration of Oak Ridge, Argonne, and Livermore (CORAL) was launched in 2014 by the U.S. Department of Energy (DOE) [14]. The CORAL collaboration is led by Office of Science and National Nuclear Security Administration (NNSA) facilities which include the Oak Ridge Leadership Computing Facility (OLCF) at Oak Ridge National Laboratory (ORNL), the Argonne Leadership Computing Facility (ALCF) at Argonne National Laboratory (ANL), and Lawrence Livermore National Laboratory (LLNL). This joint effort between three DOE national laboratories aims to build high performance computing (HPC) technologies to support DOE's mission and procure next generation large-scale systems for each participating laboratory. Two distinct architectures were selected for CORAL, one based on Intel's manycore processors, and another based on IBM Power processors with NVIDIA Volta accelerators. As a result of

J.M. Kunkel et al. (Eds.): ISC High Performance Workshops 2017, LNCS 10524, pp. 254–274, 2017.
https://doi.org/10.1007/978-3-319-67630-2_20

CORAL three new systems will be deployed in the 2018 timeframe: Aurora at ANL, Summit at ORNL, and Sierra at LLNL. Aurora will be based on Intel's third generation Xeon Phi manycore architecture and is expected to have over 50,000 compute nodes and provide 180 PFLOPS [1]. Summit is ORNL's next generation supercomputer and will be based on IBM's POWER9 architecture with multiple NVIDIA Volta GPUs per node interconnected via NVLink. Summit is expected to have approximately 3,400 compute nodes and to deliver more than 5 times the performance of Titan, ORNL's flagship supercomputer today [11]. Sierra will also be based on IBM's POWER9 processors and NVIDIA Volta GPUs and is expected to have 4–6 times the performance of Sequoia, LLNL's production supercomputer [9].

In preparation for the arrival of Summit, the OLCF procured two early access (EA) systems, Summitdev and Tundra, which are one generation removed from Summit's architecture. The goal of the EA systems is to give researchers an opportunity to optimize their applications for the Power architecture and to use multiple GPUs per node. Summitdev is the main system supporting the Center for Accelerated Application Readiness (CAAR) efforts [2]. In addition, Summitdev will be used by several researchers as part of the Exascale Computing Project [5]. Tundra, on the other hand, will be used as an internal system to test new software and gain a better understanding of the IBM ecosystem.

The EA systems were installed in November of 2016. In order to ensure that each system would be able to fulfill its main purpose, the OLCF developed an acceptance test plan that focused on functionality and reflected the needs of the CAAR applications. Acceptance of the EA systems was completed in December 2016. Both Summitdev and Tundra were officially released to users in January 2017.

This paper describes the novel features of the hardware and software stack, the test plan and procedures used to accept the system, and early results obtained from running real-world applications and benchmarks on this new architecture. Section 2 describes the system configuration, followed by a high-level description of the acceptance test plan in Sect. 3. Sections 4–8 describe the benchmarks and applications used, why they were chosen, and present individual results. Section 9 discusses several challenges and lessons learned from accepting the EA systems. Finally, Sect. 10 presents initial conclusions based on the experience the ORNL team gained from using the IBM Power architecture.

2 System Configuration

Summitdev and Tundra are the two POWER8+ early access (EA) systems deployed at the OLCF in late 2016. The main building block for the EA systems is the IBM Power System S822LC server, which is the first IBM Power-based system to provide NVIDIA NVLink Technology. Summitdev is comprised of 54 IBM POWER8 S822LC compute nodes. It has access to an NFS file system that provides home directories, as well as two high performance parallel file systems: the OLCF's center-wide Lustre parallel file system, Spider 2 [24], and a dedicated

GPFS file system, Leto. The Tundra system is based on the same server offering as Summitdev but contains 18 compute nodes. Tundra has access to a separate NFS file system that provides home directories as well as project workspaces.

Hardware. Each compute node has two IBM POWER8 processors running at 2.860 GHz in normal operation and at 3.492 GHz in turbo mode. Each processor has 10 cores, each capable of up to 8-way simultaneous multithreading (SMT), i.e., each core supports up to 8 hardware threads. Each CPU is connected to two NVIDIA Tesla P100 Pascal GPUs via NVIDIA NVLink Technology which provides a bandwidth of 80 GB/s from the CPU to the GPUs and between GPUs. Each NVIDIA Tesla P100 GPU is capable of delivering 5.3 TFLOPS of double precision, 10.6 TFLOPS of single precision, and 21.2 TFLOPS of half precision floating point performance. Furthermore, the P100 GPU is the first accelerator to use High Bandwidth Memory 2 (HBM2) and includes four vertical stacks of four memory dies totaling 16 GB of HBM2 memory and providing 732 GB/s peak memory bandwidth [7]. The compute node also has 256 GB of DDR4 memory capable of 340 GB/s peak memory bandwidth as well as one 1.6 TB Non-Volatile Memory (NVMe) device [12,20]. All compute nodes are interconnected via Mellanox EDR InfiniBand in a full fat-tree network that provides two links each with 100 Gbps bandwidth between compute nodes. Figure 1 shows the high-level structure of the compute node architecture used in Summitdev and Tundra.

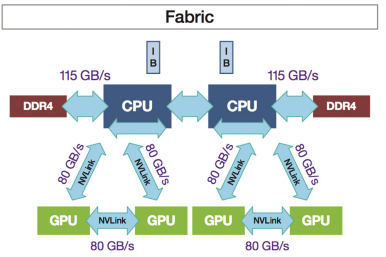


Fig. 1. IBM POWER8 S822LC compute node obtained from [20].

Software Stack. IBM provides a specialized software stack that targets their HPC offerings. The products included in IBM's HPC software stack as well as IBM's partners, NVIDIA and PGI, are described in Table 1. After Summitdev and Tundra were accepted and IBM's HPC software stack was officially released, both systems were upgraded to use the generally available (GA) software. The versions used in production are listed under the "GA" column in Table 1. Since then, the software stack has continued to mature, and the systems now have newer pre-release versions of certain components. The current versions available on the EA systems are listed under the "Production" column.

Table 1. EA systems software stack

Feature	Product	Acceptance	GA	Production	Vendor
Batch scheduler	Spectrum LSF	10.1.0[a]	10.1.0.1	10.1.0.1	IBM
MPI library	Spectrum MPI	10.1.0.2[b]	10.1.0.2	10.1.0.2	IBM
Math libraries	ESSL	5.5[b]	5.5	5.5	IBM
Compilers	XL C/C++	13.1.5[b]	13.1.5	14.1.0[b]	IBM
	XL Fortran	15.1.5[b]	15.1.5	16.1.0[b]	IBM
	PGI	16.10[a]	17.1	17.3	PGI
	clang (LLVM C/C++)	3.8.0[b]	3.8.0[b]	3.8.0[b]	IBM
	xlflang (LLVM Fortran)	4.0.0[b]	4.0.0[b]	4.0.0[b]	IBM
	GCC	4.8.5	4.8.5	4.8.5	RedHat
CUDA support	CUDA Toolkit	8.0.44 – 1[b]	8.0.54	8.0.54	NVIDIA
	CUDA Driver	361.103	361.107	375.51	NVIDIA
Parallel file system	Spectrum Scale (GPFS)	4.2.1.2	4.2.1.2	4.2.3	IBM

[a] Patched version.
[b] Beta version.

3 Acceptance Test

The OLCF developed a comprehensive acceptance test (AT) plan to verify the functionality of the EA systems. The AT plan contains three test phases: a hardware test (HWT), an I/O test (IOT), and a functionality test (FT). The AT on the EA systems took approximately three days to complete.

Hardware Test (HWT). The HWT is designed to ensure that all the hardware components are functioning correctly. This is accomplished by running vendor-provided hardware diagnostics as well as the HPL High Performance Linpack benchmark both to identify slow nodes and to ensure that each node meets or exceeds expected performance levels. The HWT also includes system administration tasks that are commonly needed in production. First, the full system is rebooted twice to ensure that it can be put back into production in a reasonable amount of time. Then, an MPI application is used to start multi-node jobs across the system and a node failure is simulated. The test is considered successful if the node failure only impacts the job that was allocated on that node. If all tests in the HWT phase succeed, the IOT is started.

I/O Test (IOT). As previously mentioned, Summitdev has access to a small GPFS file system called Leto and to the Spider 2 Lustre file system. For the IOT, only Leto was tested given that the OLCF will have a center-wide GPFS file system in the Summit timeframe. The IOT basic functionality tests include measuring metadata performance with the *mdtest* benchmark, measuring I/O

bandwidth of POSIX, HDF5, and MPI-IO using the *IOR* benchmark, and creating a 10 TB file in the file system. If no issues are observed during the IOT, the FT starts.

Functionality Test (FT). The FT phase includes tests to evaluate the functionality of compilers, math and I/O libraries, MPI implementation, and tools. To accomplish this a set of miniapps, benchmarks, and real-world applications is used. These were selected to ensure high coverage of features commonly used by scientific application developers. Table 2 summarizes the codes used during the FT phase and each code's test objectives.

Table 2. FT benchmarks, miniapps, and applications

Test	Purpose
Intel MPI Benchmarks	MPI bandwidth and latency
OpenMP 3.1 verification and validation suite	OpenMP 3.1 specification
CUDA & GPU Direct tests	CUDA, CUDA Fortran CUDA MPS, and GPU Direct
NVLink Tests	CPU↔GPU, GPU↔GPU bandwidth
SPEC OMP2012	OpenMP 3.1 functionality and performance
SPEC ACCEL ACC suite	OpenACC 1.0 functionality and performance
SPEC ACCEL OMP suite (pre-release)	OpenMP 4.5 functionality and performance
ScaLAPACK tests	Parallel dense linear algebra (DLA) operations
Minisweep	Radiation transport miniapp with OpenMP 3.1 and CUDA support
NUCCOR kernels	Nuclear physics miniapp; DLA operations using: LAPACK, OpenBLAS, ESSL; programming models: OpenMP 3.1, OpenMP 4.5, OpenACC
XRayTrace	Ray propagation miniapp; uses: C++11 threads, OpenMP, OpenACC, CUDA
Nekbone	CORAL benchmark; simulates Nek5000
HACCmk	CORAL benchmark; simulates HACC
QBOX	CORAL benchmark; first-principles molecular dynamics application
GTC	Gyrokinetic 3D particle-in-cell application

During the FT phase, each code is compiled with each target compiler. Then, a single job for each test is submitted. Once each test has completed successfully at least once, the entire set of tests is launched continuously for a period of at least 8 h. During that period, any job failure is investigated and classified.

The test phase is considered complete if there are no job failures, or in the event that there are failures, if the root cause for each failure has been identified and a remediation or a fix exists.

4 Benchmarks

Several benchmarks and standard tests were used to evaluate the functionality of the EA systems. A set of performance tests was also used to verify that the hardware met vendor specifications; this set included bandwidth intra-node tests (i.e., between node components) and inter-node tests (i.e., between compute nodes).

GPU Specific Tests. A set of tests (i.e., CUDA tests) was created to ensure correct functionality of the NVIDIA CUDA driver, the NVIDIA CUDA Toolkit, and the P100 GPUs. The set includes modified versions of the NVIDIA CUDA Toolkit code samples [4], in particular the "Simple References" examples.

Specific tests were also developed to evaluate GPUDirect capabilities (i.e., GPU_Direct tests), NVLink Technology (i.e., NVLink tests), and Unified Virtual Addressing (NVLink UVA tests). Three simple GPUDirect codes were created: *PingPong*, *Stencil*, and *Collective*. Variations of each code were also created to test CUDA MPS, CUDA Fortran, and managed memory. Tests with 1, 4, and 25 compute nodes were created for each code, and each was built with XL, PGI, GCC, and LLVM. Three additional tests were created to measure the host-device and device-device bandwidths as well as ensure proper functionality of each device. These tests are based on the *bandwidthTest*, *deviceQuery*, and *topologyQuery* code samples provided in the "Utilities" section of the CUDA SDK examples. The *bandwidthTest* example was also extended to include support for unified memory. The bandwidth values measured using the NVLink tests are shown in Fig. 2.

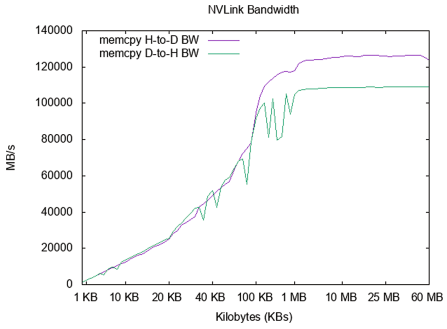


Fig. 2. NVLink host-device and device-host bandwidth using memcpy and unified memory.

During Summitdev's acceptance, the CUDA, GPU_Direct, and NVLink tests were able to identify several issues on the system. The Collective CUDA Fortran tests triggered a bug in the CUDA driver that caused the GPU to hang instead of returning an appropriate error. The Collective OpenACC test was able to identify an issue in how a pre-release version of Spectrum MPI was launching OpenACC code. The NVLink tests identified a performance regression between CUDA drivers as well as find two Summitdev compute nodes in which peer-to-peer access between GPUs 2 and 3 was not functioning correctly.

MPI Tests. The Intel MPI Benchmarks suite (IMB) [6] provides a set of tests of MPI capabilities and performance on parallel platforms. The test suite was run on Summitdev to thoroughly check functionality and evaluate target performance measurements of MPI over the EDR InfiniBand network.

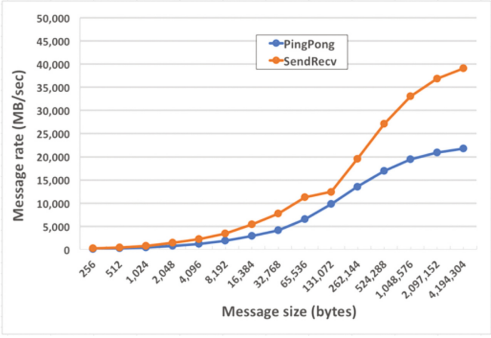


Fig. 3. MPI point-to-point bandwidth

Figure 3 shows point-to-point communication results. Maximum bandwidths attained were 21.793 unidirectional (resp. 39.062 bidirectional) GB/sec, comparing favorably to theoretical peak values of 25 (resp. 50) GB/sec; measured latencies were 1.29 (resp. 1.43) μs. This test was performed using 2 MPI ranks on 2 nodes without any special action to specify node placement.

5 Compiler Tests

The following test suites were selected to evaluate the different compiler implementations available on Summitdev. The test suites were chosen both to evaluate the implementation against the corresponding programming model specification and also to evaluate the functionality and performance of each compiler.

OpenMP Verification and Validation Suite. The OpenMP 3.1 Validation Testsuite [29] is a portable and robust validation test suite execution environment to validate the OpenMP implementation in several compilers. This test suite targets version 3.1 of the OpenMP specification, which does not support offloading to accelerators. OpenMP offloading was tested using the pre-release SPEC ACCEL suite for OpenMP. We used the version of the test suite that is included in the ongoing work to incorporate OpenMP offloading support into Clang and LLVM [18].

Table 3. OpenMP validation suite results

NTHREADS	C				Fortran		
	GNU	PGI	XL	CLANG	GNU	PGI	XL
2	94.3%	94.3%	94.3%	93.5%	88.5%	85.4%	88.5%
8	95.9%	95.9%	95.9%	95.1%	88.5%	85.4%	88.5%
16	95.9%	94.3%	95.9%	93.5%	88.5%	85.4%	87.5%
64	95.9%	94.3%	95.1%	93.5%	88.5%	85.4%	86.5%

The test suite included a total of 219 tests (123 tests for C and 96 for Fortran) that cover 115 OpenMP constructs (62 C constructs and 53 using Fortran). The test suite framework includes four varieties of tests for a target OpenMP directive; *normal tests* check that the directive (or clause) is implemented correctly, *cross tests* checks the impact of removing the target construct from the code. More details can be found in [29].

The test suite was exercised using different numbers of OpenMP threads for each of the available compilers. No special binding and mapping controls were used. Table 3 shows the percentage of successful tests for the combination of compilers, languages, and number of threads. Only GA versions of the compilers as listed in Table 1 were used. For GCC, the *gomp-4_0-branch* branch of the GCC 6.3 compiler suite was used.

The test suite results show better support for OpenMP C bindings among all tested compilers than for Fortran binding. The testing matrix for all OpenMP constructs across all compilers and using different number of threads helps identify issues with the compiler implementation (or in some cases, with the testing suite itself). For this evaluation, a total of 15 tests (2 C tests and 13 Fortran tests) failed for all combinations of compilers and number of threads, possibly indicating a problem in the tests themselves. Some tests show a different failure behavior as the number of threads change. In such cases, comparing the failure pattern with results for the same tests from other compilers may help identify if the failure is due to compiler implementation bug or an issue with the test itself that makes it invalid for certain thread counts.

SPEC ACCEL and SPEC OMP2012. The Standard Performance Evaluation Corporation (SPEC) releases a variety of realistic and standardized benchmarks to evaluate the performance of computer systems. For acceptance, two SPEC benchmark suites were used to evaluate the different compilers available for the EA systems. The benchmarks used were SPEC OMP2012 and SPEC ACCEL. The SPEC OMP2012 benchmark measures the performance of OpenMP-based applications. It includes 14 applications and is focused on OpenMP 3.1. SPEC ACCEL is a benchmark suite of computationally intensive applications and measures the performance of accelerator based systems. SPEC ACCEL supports OpenCL and OpenACC. Support for OpenMP 4.5 is currently in development and is expected to be released this year. In this work, a pre-release version of SPEC ACCEL with OpenMP 4.5 support was used to evaluate offloading capabilities of the available compilers.

For the evaluation, "base" runs were produced following SPEC rules. All benchmarks were built using common optimization flags, and were run with the test and train problem sizes, and 3 iterations of the benchmarks with the reference problem sizes. All of the metrics presented in this section are *measured estimates*. To build the benchmarks the "Production" compilers listed in Table 1 were used with the exception of the GCC compilers. For GCC, the development version of GCC 6.3.1 built from the *gomp-4_0-branch* branch was used because it provides partial support for OpenACC.

All SPEC OMP2012 runs are executed on a dedicated node for each benchmark run. For these tests 160 OpenMP threads were used to fully utilize the hardware threads available on the Power architecture. SPEC ACCEL benchmarks are also executed on a dedicated node for each benchmark run. While 4 P100 GPUs are available on the compute node, only one is used for the execution of the benchmarks.

PGI is the only production compiler that delivers successful results for SPEC OMP2012 and SPEC ACCEL for OpenACC. The measured estimates for the SPEC OMP2012 and SPEC ACCEL compute performance metrics can be seen

Table 4. Overview of execution of the SPEC OMP2012 suite.

	Benchmark													
	350.md	351.bwaves	352.nab	357.bt331	358.botsalgn	359.botsspar	360.ilbdc	362.fma3d	363.swim	367.imagick	370.mgrid331	371.applu331	372.smithwa	376.kdtree
XL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PGI	✓	✓	✓	✓	✓*	✓	✓	✓	✓	✓	✓	✓	✓	✓
GNU	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LLVM	✗[a]	✓	✓	✗[a]	✓*	✓	✓	✗[a]	✗[a]	✓	✗[a]	✗[a]	✓	✓

[a] Compile Error.

Table 5. Overview of execution of the SPEC ACCEL OpenACC suite.

	Benchmark														
	303.ostencil	304.olbm	314.omriq	350.md	351.palm	352.ep	353.clvrleaf	354.cg	355.seismic	356.sp	357.csp	359.miniGhost	360.ilbdc	363.swim	370.bt
PGI	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
GNU	✓	✓											✓		

Table 6. Overview of execution the SPEC ACCEL OpenMP suite.

	Benchmark														
	503.postencil	504.polbm	514.pomriq	550.pmd	551.ppalm	552.pep	553.pclvrleaf	554.pcg	555.pseismic	556.psp	557.pcsp	559.pminiGhost	560.pilbdc	563.pswim	570.pbt
XL	✓	✓	✗[b]	✗[b]	✗[a]	✗[b]	✗[b]	✗[b]	✗[b]	✗[b]	✗[a]	✗[b]	✓	✗[c]	✗[a]
LLVM	✓	✓	✗[b]	✗[a]	✗[a]	✓	✗[a]	✗[b]	✓	_[d]	✗[a]	✗[a]	✗[c]	✗[a]	✓

[a] Compile Error.
[b] Runtime Error.
[c] Verification Error.
[d] *556.psp* is a mixed C and Fortran code, and cannot be compiled. clang and xlflang cannot be used together.

in Tables 7 and 8. The compute performance metric (labeled as "Overall") is the geometric mean of the normalized ratios of all the benchmarks in a particular SPEC benchmark suite.

The measured estimates for OMP2012 with the IBM XL compiler can be found in Table 7. ACCEL estimates for OpenMP with the XL compiler are not presented here because the compiler does not provide full support for OpenMP 4.5 yet. The current status of ACCEL for OpenMP with the XL compiler is summarized in Table 6. While most of the benchmarks successfully compile, only 3 pass the verification. All other benchmarks currently either do not compile or link correctly, experience runtime errors, or do not pass the verification.

Measured estimates for OMP2012 using the GCC compilers can be found in Table 7. The GCC compiler provides partial support for OpenACC offloading, and because it does not yet support *acc kernels* only three benchmarks in ACCEL OpenACC, as shown in Table 5, are parallelized. GCC does not currently provide OpenMP 4.0 offloading for GPU targets, therefore ACCEL OpenMP was not included. For GCC, measured estimates obtained with the reference problem size are reported.

Table 7. Measured estimates of the SPEC OMP2012 suite (higher is better).

	XL	PGI	GNU
350.md	6.38	6.88	1.41
351.bwaves	0.898	10.90	2.14
352.nab	3.98	4.49	5.93
357.bt331	10.20	7.50	9.95
358.botsalgn	3.78	3.30	3.74
359.botsspar	2.97	3.39	3.42
360.ilbdc	8.93	8.84	0.132
362.fma3d	4.30	4.19	6.17
363.swim	10.8	9.85	11.20
367.imagick	8.63	9.15	7.71
370.mgrid331	8.52	7.67	8.75
371.applu331	11.10	8.99	12.20
372.smithwa	7.40	8.26	12.60
376.kdtree	3.78	4.46	13.50
Overall	5.53	6.50	4.74

Table 8. Measured estimates of the SPEC ACCEL for OpeACC (higher is better).

	PGI	GNU
303.ostencil	7.66	3.26
304.olbm	11.10	8.86
314.omriq	8.86	
350.md	13.70	
351.palm	2.98	
352.ep	8.47	
353.clvrleaf	8.43	
354.cg	7.18	
355.seismic	8.16	
356.sp	8.21	
357.csp	8.94	
359.miniGhost	6.67	
360.ilbdc	9.33	4.39
363.swim	5.61	
370.bt	18.70	
Overall	8.31	

The *358.botsalgn* benchmark in the OMP2012 suite does not successfully run for the test and train problem sizes when compiled with the PGI and the GCC compilers. Running the benchmarks with the "ref" problem size, however, produces valid runs. The benchmarks are marked with a * in Table 4 since they will currently not produce a reportable run according to the SPEC reporting rules [10].

6 CORAL Benchmarks

The CORAL Benchmark codes are a suite of benchmarks and mini-applications designed to represent the workloads of the laboratories involved in the CORAL collaboration and will be used to evaluate the systems when deployed. More information about the CORAL Benchmark codes and unmodified versions of the applications can be found in [3].

Nekbone. Nekbone is a CORAL benchmark used to capture the basic structure and user interface of the Nek5000 software. Nek5000 is a high order, incompressible Navier-Stokes solver based on the spectral element method. Nekbone solves a standard Poisson equation using a conjugate gradient iteration with a simple preconditioner on a block or linear geometry. The benchmark is highly scalable and can accommodate a wide range of problem sizes. The benchmark is intrinsically well load balanced, with each process having the same number of spectral elements and point-to-point communication with up to 26 surrounding neighbors.

For the purposes of acceptance, a modified version of Nekbone to utilize the GPUs based on CUDA using the XL compiler was used. A single node was used to run a small problem and verify the correct output using the accelerated version of the application.

HACCmk. The Hardware Accelerated Cosmology Code (HACC) framework uses N-body techniques to simulate the formation of structure in collisionless fluids under the influence of gravity in an expanding universe. The HACC framework was designed with great flexibility making it easily portable between different platforms. HACC uses MPI and OpenMP and depends on an external FFT library.

For acceptance, a modified version of the HACC microkernel to utilize the GPUs based on CUDA using the XL compiler. A small problem utilizing 2 nodes and 8 processes (1 process/GPU) was used to verify the correct output using the accelerated version of the application. Detailed performance data was not collected, and a comparison of runtime across different node counts, problem sizes, and CPU vs. GPU was not performed.

QBOX. QBOX is a scalable first-principles molecular dynamics (FPMD) application used to compute properties of materials. QBOX is written in C++ and uses MPI [8,26].

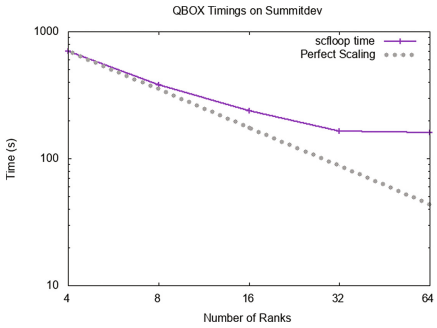


Fig. 4. QBOX results for a 640-atom bcc magnesium oxide system.

For the Summitdev acceptance, a single test case was run using 4 MPI ranks with 4 OpenMP threads per rank on a single node. To better understand the scalability of the code, four additional cases were created with 8, 16, 32, and 64 MPI ranks. Figure 4 shows the results obtained when running the standard GPU-enabled QBOX benchmark which simulates a large bcc magnesium oxide system with 640 atoms. The results show that the case does not scale well beyond 2 nodes. Further investigation is needed to better understand the scalability of QBOX.

7 Mini-applications

ScaLAPACK Tests. ScaLAPACK [21] is a Fortran library for performing dense linear algebra operations on distributed CPU-based systems using MPI. It is required by some OLCF applications including one of the thirteen Summit early readiness applications targeted by the ORNL CAAR program. For this test case, the `xsgsep` code is executed, a symmetric eigensolver test from the ScaLAPACK test suite. The test executes with four MPI ranks on one Summitdev node.

ScaLAPACK must be built against a version of the LAPACK [17] library. For this we tested two options: use of the standard LAPACK distribution, or use of the optimized LAPACK functionality found in ESSL [15]. For the latter, since ESSL is missing some required routines of the standard LAPACK version needed by ScaLAPACK, the code build's linker step was set up to use standard LAPACK as a backing library to satisfy any unsatisfied references, and repeated references to LAPACK routines were ignored. Also, the ScaLAPACK version included in the standard release of the PGI compiler was tested.

Cases run are shown in Table 9. All cases were successful except for those involving LLVM. The LLVM xlflang compiler used is an early beta version still in development. It is expected that robustness will improve as the compiler becomes more mature.

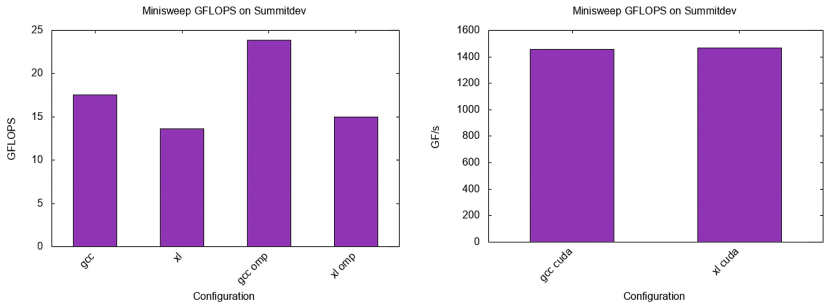
Table 9. ScaLAPACK cases tested

Compiler	LAPACK version	ScaLAPACK version	Status
GCC	Standard	Standard	Passed
GCC	ESSL	Standard	Passed
PGI	Standard	Standard	Passed
PGI	ESSL	Standard	Passed
PGI	Standard	PGI	Passed
LLVM	Standard	Standard	Failed
LLVM	ESSL	Standard	Failed
XL	Standard	Standard	Passed
XL	ESSL	Standard	Passed

Though not attempted here, ScaLAPACK use cases employing ESSL can in principle be modified to use the CPU-threaded or the CUDA-enabled version of ESSL to accelerate the performance of ScaLAPACK on Summitdev.

Minisweep. Minisweep is a miniapp designed to mimic the behavior of the sweep operation of the Denovo S_n radiation transport code [16]. It can be built with OpenMP 3.1 or CUDA support under multiple compilers in single processor or MPI mode.

Figure 5 shows the results obtained from running Minisweep using 8 MPI ranks on 2 compute nodes under the three different configurations: MPI-only, MPI with 2 OpenMP threads per rank, and MPI with CUDA. Minisweep built with the XL compiler shows a smaller performance improvement when OpenMP threads are enabled. This will require further investigation to understand how thread pinning and placement will impact performance.



(a) CPU only. (b) CPU and GPU.

Fig. 5. Minisweep results using the XL and GCC production compilers.

	GNU, F	GNU, F, OMP3	GNU, C	GNU, C, OMP3	LLVM, F	LLVM, F, OMP3	LLVM, C	LLVM, C, OMP3	PGI, F	PGI, F, OMP3	PGI, C	PGI, C, OMP3	XL, F	XL, F, OMP3	XL, C	XL, C, OMP3
ESSL																
ESSL/SMP		N/A		N/A												
ESSL/SMPCUDA		N/A		N/A												
LAPACK																
PGI LAPACK	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A					N/A	N/A	N/A	N/A
OpenBLAS																
MAGMA																
OpenMP 4	N/A	N/A	N/A	N/A					N/A	N/A	N/A	N/A				
OpenACC+cuBLAS	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A					N/A	N/A	N/A	N/A

Fig. 6. NUCCOR kernels: configuration combinations tested

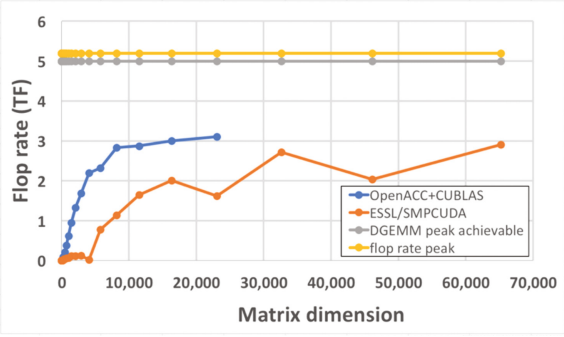


Fig. 7. NUCCOR kernels: timings for matrix products including transfers

During acceptance, the distributed version of Minisweep that uses CUDA and is compiled with GCC resulted in the highest performance. Building Minisweep with CUDA enabled with the PGI compiler resulted in build errors, and so did the OpenMP and CUDA versions of the miniapp when built with the LLVM compiler. The Minisweep test ran during acceptance was a small case to test functionality only.

NUCCOR Kernels. The NUCCOR kernels code is designed to model the performance of a significant computation of the NUCCOR nuclear physics application [23]. NUCCOR kernels computes the matrix product $C = A_1^T A_2 A_3$ for a series of dense matrix triples (A_1, A_2, A_3) of sizes representative of cases from NUCCOR workloads.

NUCCOR kernels tests multiple combinations of compiler family, source language, library and threading model. In practice, not all combinations of options are allowed, and not all allowed combinations are tested. The purpose of the test is to verify correctness for many supported combinations expected to be required by users of Summitdev rather than test all combinations.

Figure 6 shows the combinations tested. The designator "OMP3" denotes that the host code included OpenMP 3.1 constructs as a test for compatibility. In each LAPACK case, the library was entirely built by the respective host compiler. PGI LAPACK is a custom CPU-only build of LAPACK provided by PGI. OpenMP 4 cases use offload constructs and hand-coded DGEMM loops. The OpenACC/cuBLAS case uses OpenACC directives for offloading and cuBLAS for the DGEMM. The designated LLVM tests used an early OpenMP 4 implementation [13] Every case tested ran successfully and gave correct results, even though many of these combinations are very new, including XL/OpenMP 4, PGI/POWER, LLVM/Fortran and LLVM/OpenMP 4. The capability of these components to perform efficiently and interoperate correctly will be important to our users going forward.

Figure 7 shows timings for selected cases using square matrices for a range of sizes on a single GPU. The PGI Fortran compiler is used in both cases. Timings include data transfers to and from the GPU. The OpenACC+cuBLAS case benefits from less transfer due to the ability to keep an intermediate matrix on the GPU. The reasons for performance irregularities for the ESSL/SMPCUDA case, particularly for the $n = 4,081$ case, are unknown. For both cases, DGEMM performance is a significant fraction of peak attainable. We expect performance to improve as the software stack matures.

XRayTrace. The XRayTrace miniapp represents the primary computational component for a 3D coupled atomic-physics/ray-propagation code used to simulate ASE (Amplified Spontaneous Emission) and seeded X-ray lasers [16,19]. XRayTrace consists in solving many independent rays in parallel, aggregating the results to form an image that is used to couple the atomic physics in the full application.

Most C++ standard compilers are supported, and multiple programming models are tested including C++11 threads, OpenMP, OpenACC, and CUDA. No external libraries are required and all programming models are optional.

For acceptance, XRayTrace was used to test the C++ compiler support for the different programming models and to compare the relative performance of the available compilers. All GPU tests used a single GPU only, while CPU tests used all CPU cores. The timings listed in seconds only include the cost of the kernel or work performed for one iteration within the main application. Table 10 shows the results of the ASE/seeded problems for two common problem sizes. For all cases, all compilers/parallel models produced the correct output. In general, all compilers had similar timing results, CUDA showing the largest speedup. OpenACC with PGI had similar performance to CUDA. An unknown issue occurs when running OpenMP with PGI that causes a significant slowdown that was not seen with other compilers, which will require more investigation. In all cases, optimized flags for each compiler were chosen.

Table 10. XRayTrace timing results shown in seconds.

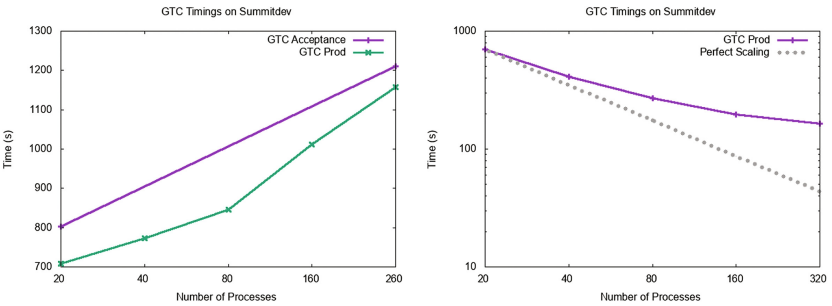
Problem	Compiler	Serial	Threads	OpenMP	OpenAcc	CUDA
ASE (small)	GCC	3.902	0.178	0.278	—	0.035
	PGI	3.296	0.197	12.082	0.038	—
	XL	2.766	0.150	0.314	—	0.056
	LLVM	3.898	0.197	0.409	—	0.059
ASE (medium)	GCC	9.377	0.376	0.611	—	0.073
	PGI	7.923	0.396	12.711	0.085	—
	XL	7.177	0.308	0.637	—	0.071
	LLVM	9.501	0.351	0.790	—	0.073
Seeded (small)	GCC	49.759	1.734	1.950	—	0.472
	PGI	49.996	1.969	619.476	0.453	—
	XL	32.519	1.481	4.609	—	0.463
	LLVM	51.546	1.776	3.991	—	0.456
Seeded (medium)	GCC	103.923	3.281	6.266	—	0.678
	PGI	117.806	4.326	371.055	0.734	—
	XL	78.791	3.017	7.189	—	0.670
	LLVM	111.715	3.558	6.529	—	0.666

8 OLCF Applications

To ensure that the system can support realistic workloads, a set of applications commonly used at OLCF were selected for acceptance of the EA systems including GTC [27], NAMD [25], and LSMS [22]. For this work, the GTC test cases were extended and its results are presented in this section.

GTC. GTC [27] is a 3D particle-in-cell code developed by the Princeton Plasma Physics Laboratory and the University of California at Irvine to study microturbulence in magnetically confined fusion plasmas. [27]. It is scalable to hundreds of thousands of processor cores and has been used previously for acceptance testing of OLCF systems [28]. The version of GTC used here is an older mature version based on MPI and OpenMP 3.1. Two cases are run, at 2 and 26 nodes with 10 MPI ranks per node and 2 OpenMP threads per rank. The cases represent 10 simulation steps with 769 radial and 3,072 poloidal gridcells with two electrons and two ions per gridcell.

After acceptance, additional cases using 4, 8, 16, and 32 nodes were added in order to better understand GTC's scaling on the Power architecture. Results obtained from running GTC with OpenMP enabled are shown in Fig. 8(a). In addition, Fig. 8(b) shows a strong scaling plot of GTC when running on Summitdev. GTC was compiled using GCC.



(a) Constant work per process. (b) Strong scaling.

Fig. 8. GTC timing results. (a) Problem size is doubled with the number of processes to keep the amount of work per process approximately constant. (b) Problem size is kept constant and the number of MPI ranks is doubled with each experiment.

9 Lessons Learned

Several considerations are necessary when porting codes to the IBM Power architecture. First, on Power, *char*s are by default *unsigned* whereas on x86 the default is *signed*. This is an important consideration as it can result in incorrect results. This was observed when running the SPEC OMP2012 benchmarks, which required the addition of *-qchars=signed*, and *-fsigned-char* for the XL and GCC compilers, respectively. Another difference to be aware of is that *long double*s on Power are by default 128-bits. It is also important to understand the different optimization levels provided by each compiler. The XL compilers, unlike GCC and PGI, provide up to optimization level *-O5* so careful consideration must be given to selection of optimization flags that match the compiler. For example, the Minisweep OpenMP tests built the XL compiler required *-O4 -qsmp=omp* in order to achieve performance improvements when compared to the MPI-only version of the code.

Looking at support for the different programming models, results showed varied levels of support. OpenMP 3.1 is well supported by the four compilers tested, however, some of the tests executed showed little or no performance improvements as with Minisweep, while others showed lower performance as was the case with Nekbone when built with PGI. This can be partially attributed to the fact that by default in the Power environment, threads are not pinned. This will require further investigation. As far as OpenMP 4.5 is concerned, two compilers are scheduled to provide support: XL and LLVM. While XL provides partial support OpenMP 4.5 offloading to the GPU, the implementation is still maturing as shown by the SPEC ACCEL OMP results. Similarly, OpenMP 4.5 support in LLVM is currently in active development. OpenACC support on Power is currently provided by the PGI compiler, which, as results of SPEC ACCEL ACC show, is a mature implementation. Partial support for OpenACC in GCC is currently provided in GCC 6.3 and is expected to be fully supported in the GCC 7 release.

10 Conclusions

At roughly 1 PF of peak performance and only 54 nodes, Summitdev is a very powerful system. The step-up in performance of the Summitdev nodes compared to Titan nodes is immediately felt by applications.

As expected with a new system, some replacements of problematic hardware were required shortly after delivery. Careful testing with multiple diagnostic benchmark codes was valuable for uncovering these problems.

Programming the node is a more complex process, ostensibly due to the presence of multiple GPUs per node, but also from other factors such as simultaneous multithreading (SMT) modes and the need to coordinate use of GPUs and CPU threads across multiple NUMA domains. The interplay of LSF and mpirun with respect to node execution configuration and the interaction of these with OpenMP and CUDA environment variables, MPS, host code threading and device selection must also be managed. This will most likely become more tractable through time and experience.

Relationships between different compiler versions, supported features and libraries have become complex and will require careful build configuration management by users. Newly developed compiler features, in some cases still in beta, are expected to mature and harden over time. Vendors are aggressively working to improve these products and respond to reported bugs.

We anticipate Summitdev to be an effective resource for preparing applications for Summit, and it has already begun to bear fruit in this regard.

Acknowledgements. The authors would like to thank the entire Summitdev Acceptance Test team for the development of tests for each phase. In addition to the authors, the team also includes: Adam Simpson, Mike Brim, Dustin Leverman, Oscar Hernandez, Chris Zimmer, Sarp Oral, Scott Atchley, and Matt Ezell.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

1. Aurora. http://aurora.alcf.anl.gov/
2. Center for Accelerated Application Readiness (CAAR). https://www.olcf.ornl.gov/caar
3. CORAL Benchmark Codes. https://asc.llnl.gov/CORAL-benchmarks
4. CUDA Samples. http://docs.nvidia.com/cuda/cuda-samples/index.html#samples-reference
5. Exascale Computing Project. https://exascaleproject.org/
6. Intel MPI Benchmarks User Guide. https://software.intel.com/en-us/imb-user-guide
7. NVIDIA TESLA P100 GPU Accelerator. https://images.nvidia.com/content/tesla/pdf/nvidia-tesla-p100-datasheet.pdf
8. Qbox: First-Principles Molecular Dynamics. http://qboxcode.org/
9. Sierra. https://asc.llnl.gov/coral-info

10. SPEC ACCEL Run and Reporting Rules. https://www.spec.org/accel/docs/runrules.html
11. Summit: Scale New Heights Discover New Solutions. https://www.olcf.ornl.gov/summit/
12. Summitdev Quickstart. https://www.olcf.ornl.gov/kb_articles/summitdev-quickstart
13. Using OpenMP 4.5 in the CLANG/LLVM compiler toolchain. https://www.olcf.ornl.gov/wp-content/uploads/2017/01/SummitDev_Using-OpenMP-4.5-in-the-CLANGLLVM-compiler-toolchain.pdf. Accessed 12 Apr 2017
14. Fact Sheet: Collaboration of Oak Ridge, Argonne, and Livermore (CORAL). Technical report, Department of Energy (2014). https://energy.gov/sites/prod/files/2014/12/f19/CORAL%20Fact%20Sheet__FINAL%20AS%20ISSUED_UPDATED.pdf
15. ESSL Guide and Reference (2016). https://publib.boulder.ibm.com/epubs/pdf/a2322688.pdf
16. Miniapps derived from production HPC applications using multiple programing models. Int. J. High Perform. Comput. Appl. 1094342016668241 (2016). doi:10.1177/1094342016668241
17. Anderson, E., Bai, Z., Dongarra, J., Greenbaum, A., McKenney, A., Du Croz, J., Hammerling, S., Demmel, J., Bischof, C., Sorensen, D.: LAPACK: a portable linear algebra library for high-performance computers. In: Proceedings of the 1990 ACM/IEEE Conference on Supercomputing, Supercomputing 1990, CA, USA, pp. 2–11 (1990). http://dl.acm.org/citation.cfm?id=110382.110385
18. Antao, S.F., Bataev, A., Jacob, A.C., Bercea, G.T., Eichenberger, A.E., Rokos, G., Martineau, M., Jin, T., Ozen, G., Sura, Z., Chen, T., Sung, H., Bertolli, C., O'Brien, K.: Offloading Support for OpenMP in Clang and LLVM. In: Proceedings of the Third Workshop on LLVM Compiler Infrastructure in HPC, LLVM-HPC 2016, pp. 1–11. IEEE Press, Piscataway (2016). doi:10.1109/LLVM-HPC.2016.6
19. Berrill, M.: Modeling of laser-created plasmas and soft x-ray lasers. Ph.D. thesis, Colorado State University (2010)
20. Caldeira, A., Haug, V., Vetter, S.: IBM Power Systems S822LC for High Performance Computing: Technical Overview and Introduction. Technical report, IBM, September 2016
21. Choi, J., Dongarra, J.J., Pozo, R., Walker, D.W.: ScaLAPACK: a scalable linear algebra library for distributed memory concurrent computers. In: 1992, Fourth Symposium on the Frontiers of Massively Parallel Computation, pp. 120–127. IEEE (1992). http://ieeexplore.ieee.org/document/234898/. Accessed 11 Oct 2016
22. Eisenbach, M., Zhou, C., Nicholson, D.M., Brown, G., Larkin, J., Schulthess, T.C.: Thermodynamics of magnetic systems from first principles: WL-LSMS
23. Hagen, G., Jansen, G.R., Papenbrock, T.: Structure of ^{78}Ni from first-principles computations. Phys. Rev. Lett. **117**, 172501 (2016). https://link.aps.org/doi/10.1103/PhysRevLett.117.172501
24. Oral, S., Dillow, D.A., Fuller, D., Hill, J., Leverman, D., Vazhkudai, S.S., Wang, F., Kim, Y., Rogers, J., Simmons, J., et al.: OLCFs 1 TB/s. Next-Generation Lustre File System
25. Phillips, J.C., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., Chipot, C., Skeel, R.D., Kalé, L., Schulten, K.: Scalable molecular dynamics with NAMD. J. Comput. Chem. **26**(16), 1781–1802 (2005). doi:10.1002/jcc.20289
26. Schlipf, M., Gygi, F.: Optimization algorithm for the generation of ONCV pseudopotentials. Comput. Phys. Commun. **196**, 36–44 (2015). http://www.sciencedirect.com/science/article/pii/S0010465515001897

27. Tang, W., Wang, B., Ethier, S., Lin, Z.: Performance portability of HPC discovery science software: fusion energy turbulence simulations at extreme scale. Supercomputing Front. Innovations **4**(1), 83–97 (2017)
28. Tharrington, A., Hai Ah Nam, W.J., Brown, W.M., Anantharaj, V.G.: Early applications experience on the cray XK6 at the Oak Ridge leadership computing facility. In: Cray User Group Meeting CUG (2012)
29. Wang, C., Chandrasekaran, S., Chapman, B.: An OpenMP 3.1 validation testsuite. In: Chapman, B.M., Massaioli, F., Müller, M.S., Rorro, M. (eds.) IWOMP 2012. LNCS, vol. 7312, pp. 237–249. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30961-8_18