LLNL-TR-946569

Lawrence Livermore National Laboratory

Level-2 Milestone 6459: Sierra Applications Development Environment

Prepared by W. Scott Futral, Dong Ahn, Blaise Barney, Chris Chambreau, Elsa Gonsiorowski, John Gyllenhaal, Ramesh Pankajakshan (Livermore Computing), Max Katz (Nvidia), Roy Musselman (IBM)
August 27, 2018

Disclaimer

Table of Contents

Section 1

Introduction

This report documents the deployment of the applications development environment for the CORAL Sierra Initial Delivery system, which consisted of the first one-quarter of the full Sierra hardware. The milestone wording is as follows:

> *To coincide with the siting and installation of the Sierra Initial Delivery system (one-quarter size), the Scalable Applications Preparation (SAP) effort within the Applications Development Environment and Performance Team (ADEPT) will deploy an application development environment that enables initial porting, tuning, and characterization of ASC application codes. We will extend the CORAL EA system development environment to support new features provided with the Power9 Central Processing Unit (CPU), NVIDIA Volta GPU hardware, and new system software. The team will evaluate MPI-direct-from-GPU messaging, unified memory malloc, and burst-buffer related features. Third party tools and software for debugging and performance analysis will also be deployed and evaluated for expected capabilities.*

We document handoff to a Weapons Program customer using the Sierra Initial Delivery system. This report represents the completion of milestone 6459, Sierra Applications Development Environment.

Section 2

Overview

The system architecture for the Sierra system is described in some detail in the **Level-2 Milestone 6461: Sierra Initial Delivery System Acceptance**. The development environment strategy for the Early Delivery Systems was described in **Level-2 Milestone 6005: Applications Development Environment for Sierra Early Arrival Systems** and is extended for the current generation Sierra Initial Delivery system with the new features of the IBM POWER9 processor and Nvidia Volta GPU accelerator, updated system software, and additional scale taken into consideration. Evaluations of specified new features are presented in the 'Select Feature Evaluations' section of the report. Other sections provide a description of some significant advances to the code-development tools as the result of concerted efforts by the SAP team in concert with vendors and third-parties. Another significant advancement to the software environment for Sierra is the enhanced ability to handle complex workflows, which is enabled by the Flux resource manager effort at Livermore Computing and is described in the relevant section here.

Section 3

Documentation, Training, and Support

Elements of the user training and documentation of the development environment are included in the appendix of this document. These are the external-internet available documents **'Using LC's Sierra Systems Tutorial'**, and the presentation slides for topics covered in the **'Sierra Systems Getting Started Workshop'.** Not included in this report, but due to the dynamic and evolving state of the development environment for Sierra, additional documentation for users is provided in a wiki-based format that is accessible under password control for valid LC user accounts. This provides for access to users of preliminary or controlled LC, user generated, and vendor provided documentation that may not yet be approved for general release. This information is often the basis for updates of the external web-based documentation once it is vetted and approved for release.

Section 4

Code-Development Tools

Providing our code-development tool set on Sierra required significant effort above and beyond that on early arrival systems. The capabilities within our code-correctness, debugging, performance-profiling and MPI-tracing tools on Sierra is a superset of those on the earlier systems. Further, adapting our tool set to Sierra's unique environment in terms of its system software, hardware and sheer scale, as well as addressing the user feedback gathered from the existing systems not only required new feature development but also coordination effort among hardware vendors, third-party tool developers and LC staff members. The following highlights some of those efforts.

Providing a fully capable debugging environment continued to present a major challenge due in large part to the introduction of new GPUs (i.e., NVIDIA Volta) and a resource manager (i.e., jsrun), and we mitigated the challenge with a few initiatives: 1) Responding to ASC users' feedback, we worked with Rogue Wave Software to modify the TotalView debugger's GPU breakpoint handling, unifying ways that CPU and GPU breakpoints are handled. The new capability referred to as CUDA Unified Breakpoint has already proven to be effective on ASC codes as it allows for a seamless CPU/GPU debugging work flow. 2) We also worked closely with hardware vendors to analyze and address some of the critical system deficiencies that prevented us from providing the full functionality of our debuggers. This effort included the deployment of a fix for a long-standing bug within a system driver software (libcuda.so) that prevented TotalView from attaching to GPU code. 3) We newly co-designed the post-mortem debugging subsystem on Sierra to allow not only full binary core dumps but also lightweight dumps for both CPU and GPU, striking a balance between capability and scalability. The co-design also enabled our users to induce GPU core dumps, a capability that proved critical for the recent debugging of a race condition in one of the system libraries (i.e., NCCL). To fully realize the potential of this underlying capability, the Stack Trace Analysis Tool's post-mortem debugging has been ported to the new scheme as well.

With the architectural trends towards more complex memory hierarchy, being able to analyze data motion within the compute node has been identified as critical. To respond to this need, one of our thrust areas was to provide a software infrastructure that allows for monitoring data motion within the compute node. For this purpose, we co-designed with IBM a performance hardware counter subsystem, referred to as NEST events. NEST counters allow users to monitor detailed data motion at various memory transfer interfaces (e.g., measuring number of bytes flowing through NVLINK and memory controller ports). Unfortunately, utilizing NEST required

a root privilege, a major usability concern. Thus, our co-designed solution was to enable users to start up Performance Co-Pilot (PCP) daemons that can read NEST counters as root and provide the measurements to user-level tools. To avoid higher noise levels that can arise if these daemons are persistently active, we enabled this only as an on-demand capability so that users can turn it on only for their own jobs.

We continued to work closely with both open source performance tool developers and vendor personnel to adapt their tools to new environment as well as the new capabilities such as NEST and PCP above. The end result is that nearly all of the tools we had planned have been deployed, which includes PAPI, HPCToolkit, Open|SpeedShop, TAU, mpiP, NVIDIA's NVPROF and NVPP, and IBM's PPT. The main development efforts of these tools have been to support the performance analysis of Sierra's unique features including GPU and OpenMP4 support while leveraging new capabilities such as NEST and PCP. As the results of long partnership with vendors, CASC researchers and academia, we have also successfully deployed a set of state-of-the-art memory and thread correctness checkers. This includes the AddressSantizer memory checker, the Archer OpenMP data race detector and the ReMPI record and replay tool.

While our efforts have led to a highly capable tools ecosystem, our evaluation suggests that there are system software issues that we still need to resolve in order to make our tools ecosystem more robust and scalable. For instance, applying our debuggers to ASC applications revealed that there are still performance issues for certain primitive operations within the GPU debug and performance APIs (e.g., a performance regression on NVIDIA Volta in kernel launch under CUDA debug API control). We will continue to work closely with hardware vendors to remediate these issues. Additionally, IBM has included the LLNL-based MPI profiling tool mpiP in their SMPI distribution. We will continue to work with IBM to address configuration issues to provide full mpiP functionality as well as new functionality implemented by mpiP developers.

Section 5

Enhancements for Complex Workflows

A majority of early science applications being run on Sierra features non-traditional co-scheduling and execution patterns that the existing system scheduler could not easily cope with. The Cancer Moonshot Pilot2 workflow features the co-scheduling of coupled simulations at different scales (i.e., continuum models-based simulations with several thousand MD simulations, coordination between CPU and GPU runs), the use of a machine learning module to schedule, de-schedule, and execute simulations dynamically at a high rate, and the use of a data store to coordinate the data flow between different tasks. Similarly, the Merlin workflow, a component of the Machine Learning Strategic Initiative (MLSI), strives to schedule 1 billion tasks with the similar co-scheduling needs.

To help these science applications overcome the scheduling challenges, we ported and provided Flux on Sierra, our next-generation resource management framework which is under active development at LLNL. At the heart of Flux lies its ability to be nested seamlessly within batch allocations created by other schedulers as well as itself. Once a hierarchy of Flux instances is created within each allocation, Flux's consistent and rich set of well-defined APIs portably and efficiently support those workflows that can often feature non-traditional execution patterns.

Flux has proven to be highly effective on these emerging, non-traditional workflows. The Pilot2 project integrated Flux into their workflow manager to handle the volume of jobs and the required co-scheduling of resources. The MSLI's Merlin workflow also integrated Flux to solve both jsrun's co-scheduling issue and nesting issue (i.e., Sierra does not handle nested launches where there was one jsrun call for the allocation and a subsequent jsrun call for the simulation). As ASC workflows are also getting increasingly more complex -- e.g., a strong push towards rigorous verification and validation (V&V) and uncertainty quantification (UQ) – our effort lights the exascale path for our ASC applications.

Section 6

Select Feature Evaluations

In addition to the software development, deployment and documentation completed for this milestone, the SAP team has conducted evaluations of technical performance for many aspects of the Sierra system features and software. Evaluations of several new features of Sierra, including those mentioned in the milestone description, are summarized here.

- ## IO Libraries and Burst Buffer

  The Sierra Systems include node-local SSDs which take advantage of NVMe over Fabrics technologies. These devices will be managed via the LSF resource scheduler and integrated with new IBM provided software. As a whole, the Burst Buffer presents users with a new level in the storage hierarchy which must be utilized and managed. For this milestone we have worked with users to educate them about the burst buffer concept and how to best utilize the storage hierarchy on Sierra. This work has continued through attendance at code team meetings and Joint Sierra Readiness sessions.

  In addition to user education, we have developed libraries and tools to enable application portability. These tools provide an abstraction layer above the complex storage hierarchy and vendor-specific APIs.

  First, we have developed AXL, the Asynchronous Transfer Library. AXL includes support for a number of transfer protocols, including IBM's BBAPI technologies and Cray's DataWarp technologies. In addition, we have included a similar portability layer in the SCR (Scalable Checkpoint Restart) tool. Particular to the checkpoint/restart use case, SCR manages file sets across node-local resources. Finally, we have developed mpiFileUtils. These tools allow users to efficiently manage large file sets through a familiar set of file utilities. The mpiFileUtils use MPI to implement efficient unix file tools (such as "cp" and "rm"). In addition to providing these tools, we have created documentation and performed user training.

  Finally, we have continued to work closely with IBM's Burst Buffer development team as the software stack and LSF integration has been developed. Throughout the development process, we have continued to communicate the needs of LLNL's applications and necessary features to IBM's team. Technical and performance evaluations of the Burst Buffer software will continue as features become available on the Sierra systems at LC.

- **MPI from GPU Memory**

Spectrum MPI(10.2.0.06rtm0) supports the use of certain point-to-point and collective operations with data objects that reside on the GPU in either device or managed memory. This capability simplifies programming by making MPI calls agnostic to data location while enabling optimizations from reduced data copying operations. This feature of Spectrum MPI was evaluated using point-to-point messages to measure the bandwidth between 2 Sierra nodes with 1 process per node while also checking the messages for correctness. The figure below compares the effective bandwidth for messages ranging from 8 bytes to 64 MB. The computed bandwidth was the average of 100 repeats. The achieved bandwidths for both pageable and pinned memory buffers on the host were also plotted for reference.

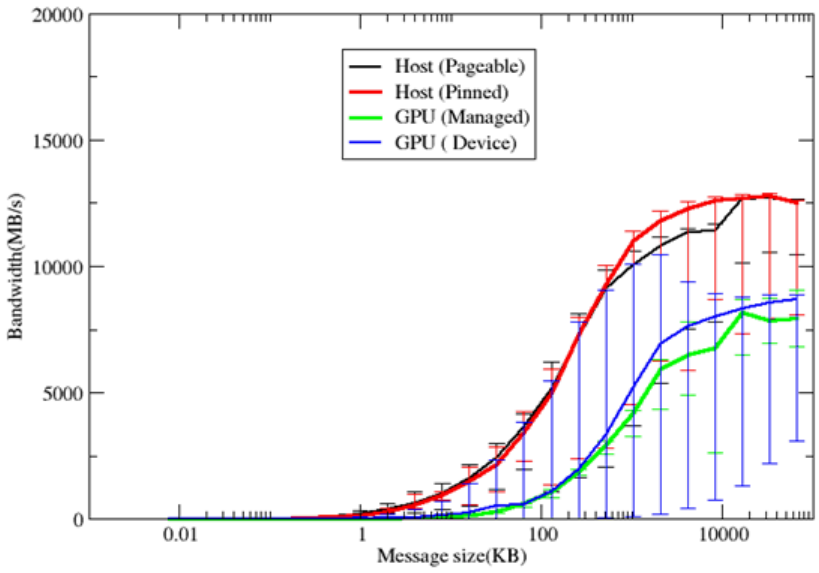

Figure 1: Effective bandwidth between two Sierra nodes based on data location

The figure shows that the maximum bandwidth achievable for buffers on the GPU is less than the maximum achievable on the host by 33%. The capability of Sierra to perform direct MPI messaging from GPU memory is functional but not fully performant.

- **MPI Collectives using SHARP**

IBM's Spectrum MPI includes support for Mellanox's Fabric Collective Accelerator (FCA) and Scalable Hierarchical Aggregation and Reduction Protocol (SHARP) capabilities which use network-based support for optimizing some collective operations. When setting the appropriate flags to use SHARP, benchmarking of some MPI collectives at 1024 nodes on

sierra has resulted in improvements of 2-3x lower latency relative to the FCA measurements.

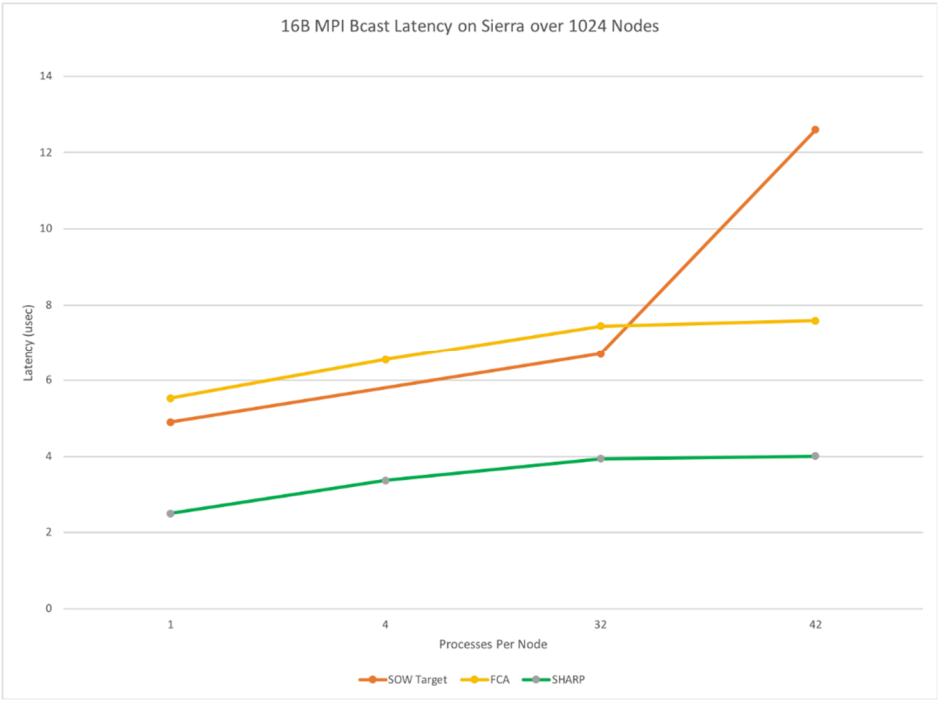

Figure 2: Bcast Latency Improvements with SHARP

The figure shows the improvement in Bcast latency over FCA when using SHARP.

We have provided example settings for users to activate FCA and SHARP functionality with the jsrun command and how to verify use of SHARP from run time messages. We have also provided guidance for tuning parameters and identifying additional settings.

- **Unified Memory Management**

The CORAL acceptance software stack includes an NVIDIA driver and IBM system firmware that together enable CPU-GPU cache coherence through the Address Translation Service (ATS) protocol. Through the ATS mechanism, the CPU and GPU can coherently read and write to each other's memory. Prior to the enablement of ATS, this type of functionality could only be achieved by explicitly opting into CUDA Unified Memory. Now, the CPU can allocate memory with malloc() and this memory is coherently accessible on the GPU.

The ATS functionality was originally enabled in March 2018 but was quickly discovered to lead to very large performance regressions (30%) in High Performance Linpack runs on Sierra. As a result, it was discontinued for production use on Sierra; however, further

analysis continued, with IBM and NVIDIA engaged to determine the root cause of the issue. The eventual fix that was determined was implemented in the Linux kernel and a kernel with the patch was delivered to LLNL in August 2018. Initial review of the ATS-enabled platform indicates that for several CORAL benchmarks, including LULESH, AMG, and SNAP, performance -- as measured by the Figure of Merit -- with ATS enabled is within 2% of the value for the non-ATS platform, indicating that the ATS functionality adds minimal overhead cost for these cases. However, ATS functionality does add unavoidable overhead for certain operations involving memory mapping, leading to some operations being slower when ATS is enabled. Current efforts are focused on understanding the extent of this effect, and the impact that this may have on production science applications.

In terms of the new functionality offered by ATS, initial investigation indicates that ATS may offer improvements in cases where an explicit memory allocation call can be avoided, since in the CUDA paradigm these memory allocations are more expensive than (pageable) CPU memory allocations. The cache coherency mechanism can also be used to enable easier porting of code to GPUs that is hard to explicitly manage memory allocation for on the GPU. However, there are still significant caveats. At this time, functionality to automatically migrate memory from the CPU to the GPU on demand (similar to what is available through CUDA Unified Memory) has not been delivered, so the functionality does not yet offer competitive performance for most program codes at LLNL, which relies on the most commonly accessed data residing on the GPU for low-latency, high-bandwidth access. There does exist functionality to manually migrate memory between devices, however this is also not yet at performance competitive with existing CUDA unified memory.

Overall, the ATS functionality has been demonstrated to work correctly and its enablement is no longer a significant disruption to existing workloads. At LLNL, work is ongoing to determine where it may be best used, and at the vendors, work continues to enable higher performance for codes that rely on ATS exclusively.

Appendix A

Certification Letter and Documentation

TO: LLNL ASC Office
FROM: Peter B. Robinson
SUBJECT: Completion of LLNL ASC Level 2 Milestone 6459

As a Weapons Program application code developer and user of the CORAL EA and initial delivery systems (rzmanta, sierra, rzansel, and shark), I certify that the application development environment provided the essential capabilities that enabled porting, tuning, debugging, characterization and running of ALE3D successfully on the aforementioned platforms. In the last year, efforts to deliver consistent upgrades to the environment on those platforms as well as efforts to improve the debugging capability for GPU programs running at scale have been particularly impactful. This satisfies the requirements of the LLNL ASC Level 2 Milestone 6459, "Sierra Applications Development Environment."

During the siting and installation of the Sierra Initial Delivery system and continuing on to installation of the full-scale system, ALE3D has been ported and tuned to run on Power8+/P100 GPU platforms on both the OCF and SCF, as well as Power9/V100 GPU platforms on the OCF. Ale3d used multiple compilers (nvcc, xlf, clang) and spectrum MPI, provided by Livermore Computing (LC) to run these calculations. As these software stacks were updated during acceptance efforts, we were able to leverage those updates consistently across those platforms thanks to the well documented installations provided by LC. Shortfalls, especially in the totalview debugger, were identified early in the process and have been addressed. We have been largely pleased with the level of support we have been receiving and expect it to continue as the development environment reaches a production level of maturation.

Date signed,
08/27/2018

Peter B. Robinson
ALE3D Computer Science Lead

[Using LC's Sierra Systems Tutorial]

[Sierra Systems Getting Started Workshop]