Non-Recurring Engineering (NRE) Best Practices: A Case Study with the NERSC/NVIDIA OpenMP Contract

Christopher S. Daley
csdaley@lbl.gov
Lawrence Berkeley National
Laboratory
Berkeley, California, USA

Annemarie Southwell
asouthwell@nvidia.com
NVIDIA Corporation
Hillsboro, Oregon, USA

Rahulkumar Gayatri
rgayatri@lbl.gov
Lawrence Berkeley National
Laboratory
Berkeley, California, USA

Scott Biersdorff
sbiersdorff@nvidia.com
NVIDIA Corporation
Hillsboro, Oregon, USA

Craig Toepfer
ctoepfer@nvidia.com
NVIDIA Corporation
Hillsboro, Oregon, USA

Güray Özen
gozen@nvidia.com
NVIDIA Corporation
Berlin, Germany

Nicholas J. Wright
njwright@lbl.gov
Lawrence Berkeley National
Laboratory
Berkeley, California, USA

ABSTRACT

The NERSC supercomputer, Perlmutter, consists of AMD CPUs and NVIDIA GPUs. NERSC users expect to be able to use OpenMP to take advantage of the highly capable GPUs. This paper describes how NERSC/NVIDIA constructed a Non-Recurring Engineering (NRE) contract to add OpenMP GPU-offload support to the NVIDIA HPC compilers. The paper describes how the contract incorporated the strengths of both parties and encouraged collaboration to improve the quality of the final deliverable. We include our best practices and how this particular contract took into account emerging OpenMP specifications, NERSC workload requirements, and how to use OpenMP most efficiently on GPU hardware. This paper includes OpenMP application performance results obtained with the NVIDIA compilers distributed in the NVIDIA HPC SDK.

CCS CONCEPTS

• **Computer systems organization** → **Heterogeneous (hybrid) systems**; • **General and reference** → *Performance*; • **Theory of computation** → *Parallel computing models*; • **Hardware** → *Emerging languages and compilers*.

KEYWORDS

NRE, OpenMP target offload, NVIDIA GPU, Supercomputer procurement, Compiler development

ACM Reference Format:

Christopher S. Daley, Annemarie Southwell, Rahulkumar Gayatri, Scott Biersdorff, Craig Toepfer, Güray Özen, and Nicholas J. Wright. 2021. Non-Recurring Engineering (NRE) Best Practices: A Case Study with the NERSC/NVIDIA OpenMP Contract. In *The International Conference for High Performance Computing, Networking, Storage and Analysis (SC '21)*, November 14–19, 2021, St. Louis, MO, USA. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3458817.3476213>

1 INTRODUCTION

HPC Centers are increasingly deploying heterogeneous supercomputers in order to obtain necessary performance increases. These supercomputers provide tremendous capabilities but, potentially, are susceptible to shortcomings in software technologies that limit the achievable performance for real applications. Non-Recurring Engineering (NRE) contracts are one-time focused investments designed to refine such technologies in ways beneficial to HPC Centers and their users, and at the same time produce commercially viable solutions, which will be sustainable in the long run. The NRE investment is needed because market forces alone are insufficient to prompt the vendor to make the desired investments, and to enable the HPC center to influence the solution produced, to be sure it meets its, and its users, needs. There are numerous examples of Department of Energy (DOE) NRE investments: Pathforward [46], parts of CORAL-1 [40, 56] and CORAL-2, Cray DataWarp [2], and Codeplay SYCL compiler development [27].

The National Energy Research Scientific Computing (NERSC) HPC Center is the production compute facility for the DOE Office of Science. The supercomputers at NERSC are used by over 8000 users, representing approximately 700 codes. NERSC recently invested \$146 million into the Perlmutter supercomputer to support NERSC user research until approximately 2025 [55]. This supercomputer consists of CPU-only nodes and NVIDIA GPU-accelerated nodes. NERSC staff recognized early on in the project that OpenMP could simplify the porting of some applications to the GPUs of

This work is licensed under a Creative Commons Attribution International 4.0 License.
SC '21, November 14–19, 2021, St. Louis, MO, USA
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8442-1/21/11.
<https://doi.org/10.1145/3458817.3476213>

Perlmutter. There were non-portable methods, e.g. CUDA [41] and Thrust [43], and portable methods, e.g. OpenACC [45], Kokkos [20], and RAJA [3]; there were no plans, however, for any vendor compiler to support OpenMP offload [6] to the NVIDIA GPUs planned for Perlmutter. This was especially problematic for NERSC, as many users had modified their applications using OpenMP in order to achieve good performance on NERSC's Intel Knights Landing (KNL) based machine, Cori, which was delivered in 2016. With this in mind, NERSC and NVIDIA entered into an NRE contract, with the goal of adding OpenMP GPU-offload capabilities to the NVIDIA compiler suite.

This paper describes the OpenMP NRE contract between NERSC and NVIDIA, with a mind to contributing this narrative as a best practice to the HPC community at large. We describe the considerations that went into the formation of the contract, the contract itself, the milestones and their format, and our collaboration on applications and the influence this had on the compiler suite. We also describe our experiences implementing the new OpenMP standards for NVIDIA GPUs and discuss some of the issues encountered and how the partnership addressed them. Overall, the effort is part of a larger DOE wide performance portability effort, which is designed to ensure that there are vendor-supported OpenMP compilers for DOE users on all current and next generation supercomputers.

Our principle recommendations for such a HPC-center vendor engagement are

- Seek to construct a strong partnership, where both parties understand each other's needs and constraints, and which utilizes each other's strengths. In this particular example, NERSC has strong links to the user community, and can advise on the relative importance of various features and functionality. NVIDIA has a deep knowledge of the GPU architecture, and the ease or difficulty of implementing and maintaining a particular compiler feature.
- Start discussions and contract negotiations as early as possible to give sufficient time for joint research and development. NERSC/NVIDIA contract negotiations began in April 2018 which is approximately 3 years in advance of Perlmutter delivery.
- Write the contract milestones to allow for flexibility, and the ability to adapt to changing circumstances. In this particular case, the contract was written before the OpenMP 5.0 & 5.1 standards were complete, and NVIDIA was understandably reluctant to promise to support an unknown standard. Yet an OpenMP 5.1 compiler was delivered. The contract deliverables were written such that a mutually agreed upon, to be determined during the course of the contract execution, subset of the standard would be supported.
- Ensure that problems representative of your user community are addressed. The contract specified that five NERSC user applications would be ported to OpenMP as part of the effort. This resulted in support for parts of the OpenMP standard not originally anticipated, as well as the production of some exemplar applications and examples for training purposes, which will be released, as much as possible, to the HPC community [4, 13, 52, 53].

- Ensure that the delivered product is both functional and performant. Ensuring that the delivered compiler achieved its performance targets was very important to both NERSC and NVIDIA. If the compiler did not meet functional or performance targets then early adopters would simply give up on it, with potentially irredeemable damage to the product's reputation and the strategy of programming NVIDIA GPUs with OpenMP. This consideration also led to the support of only a subset of the OpenMP standard; parts unlikely to give good performance on GPUs are not supported. NERSC/NVIDIA carefully tracked functionality and performance throughout the duration of the contract: the first production compiler (May 2021) successfully compiled and executed 98% of our test cases and met performance requirements.

The rest of the paper is organized as follows. Section 2 gives background about the Perlmutter supercomputer, OpenMP, and the motivations of NERSC/NVIDIA to provide a GPU-enabled OpenMP compiler. Section 3 explains the Statement Of Work (SOW) that defined the partnership between NERSC and NVIDIA to deliver a GPU-enabled compiler for Perlmutter. The section explains the individual milestones, our thought process behind adding these milestones, and how we measured success for each milestone. Section 4 explains the project management activities to keep the project on track and enable useful collaboration between NERSC and NVIDIA. Section 5 lists the applications used in a test suite and how the choice of applications gave coverage of OpenMP features expected to be most important for NERSC users. Section 6 explains our continual engagement about GPU data management, performance portability to CPUs and GPUs, features to support in offloaded code sections, and interoperability. Section 7 highlights the compiler capabilities and performance achieved on microbenchmarks and NESAP-2 applications. Finally, Section 8 concludes the work by providing a critical assessment of the NRE project.

2 BACKGROUND

2.1 Perlmutter Supercomputer

The Perlmutter supercomputer was deployed at NERSC in 2021. Perlmutter is based on the HPE Cray "Shasta" platform and was deployed in two phases. The Phase 1 system consists of over 1500 GPU-accelerated nodes, each with 1 AMD Milan CPU and 4 NVIDIA A100 GPUs and 256 GB of memory per node. The Phase 2 system consists of over 3000 CPU-only nodes, each with 2 AMD Milan CPUs and 512 GB of memory per node. Heterogeneity at the system level enables NERSC to support a broad workload with different levels of GPU-readiness. In preparation for Perlmutter, NERSC deployed two development systems with GPUs. The first, named Cori-GPU, is an 18 node Cray CS Storm consisting of nodes with 2 Intel Skylake CPUs and 8 NVIDIA V100 GPUs. The second, named Cori-DGX in this paper, is 2 nodes of NVIDIA DGX with nodes consisting of 2 AMD Rome CPUs and 8 NVIDIA A100 GPUs.

2.2 OpenMP

OpenMP is a set of directives and APIs that may be used to create parallel C, C++ and Fortran applications. It is a portable standard that has a long history of success over twenty years [15]. Features have been added to the specification to keep up with evolving

computer architectures. Recent additions include SIMD and device directives to execute on highly data parallel many-core and GPU devices.

The OpenMP-4.5 specification provides significant support for device offload and was released in Nov 2015. In the specification, the OpenMP target directive specifies that a region of code should be executed on a device. Multiple compilers support execution of OpenMP code regions on a GPU device, including the open source LLVM/Clang compiler which supports offload to NVIDIA GPUs [32]. It is common for OpenMP applications utilizing device offload to be labeled as OpenMP target offload applications. The OpenMP-5.0 and 5.1 specifications include many clarifications and feature enhancements to OpenMP device offload and were released in Nov 2018 and Nov 2020, respectively.

2.3 NERSC and NVIDIA priorities

NERSC and NVIDIA had multiple meetings in 2018 to understand how an OpenMP engagement could work in a way that would benefit both parties. These initial discussions were crucial to put together a SOW that would create a production-ready compiler aligned with the priorities of NERSC and NVIDIA.

NERSC is a strong proponent of OpenMP. The staff working at NERSC are involved in the OpenMP standard committee and have been educating NERSC users about how to use OpenMP for a long time. Staff interactions with users have revealed increased use of OpenMP by NERSC users over the years. The NERSC Energy Research Computing Allocations Process (ERCAP) 2017 user survey asked "Do your codes use any of the following (anywhere, not just at NERSC)?". Of the 328 question respondents, over 80% selected OpenMP. In addition, NERSC users involved in the NERSC Exascale Science Applications Program (NESAP-1) had significant success using MPI+OpenMP on the Cori supercomputer [30]. Here, NESAP-1 was focused on achieving high performance on Intel Knights Landing (KNL) many-core CPUs. This demonstrated that significant parallelism could be exploited by many NERSC applications which in aggregate make up a large fraction of the total NERSC workload.

The NERSC strategy is to enable users to continue using MPI+OpenMP on the CPU-only nodes and CPU+GPU nodes of the Perlmutter supercomputer. This strategy is motivated by the desire to use a portable programming approach with as little disruption to users as possible. A major challenge back in 2018 was the lack of NERSC staff experience with OpenMP on GPUs. This was coupled with the fact that the handful of staff who had used OpenMP on GPUs experienced a lack of compiler maturity, particularly for Fortran OpenMP applications. The NERSC staff found that the compilers often generated incorrect code and/or poorly performing code. This led to a strong desire within NERSC for a robust compiler which would support the common uses of OpenMP in a range of scientific applications. The compiler should be able to compile and correctly execute non-trivial C, C++ and Fortran applications and make it possible to obtain acceptable performance without significant low-level tuning.

Many HPC applications have achieved high performance on NVIDIA GPUs using one or more of the CUDA, Thrust, and OpenACC programming models. Concern about working with these approaches effectively kept some NERSC users from considering

introducing GPU acceleration to their applications. In the initial discussions, NVIDIA made it clear that not all OpenMP applications could simply be recompiled and be expected to achieve high performance on NVIDIA GPUs. Attaining highest performance on GPUs requires $O(10^5)$ -way fine-grained parallelism, sufficient work per kernel execution, minimal synchronization, and application data structures enabling contiguous memory access. It is not possible for a compiler to create these characteristics if they are missing from an application. Application programmers must make appropriate source modifications and use parallel programming abstractions well-matched to the architecture of a GPU to fully benefit from the massive parallelism GPUs provide. This need for developer participation led to one of the priorities of the NRE contract, to focus upon the subset of OpenMP features that could be expected to achieve high performance on NVIDIA GPUs. Features expected to limit application performance should be excluded, e.g. OpenMP tasks generated on a GPU, OpenMP locks, and OpenMP critical sections. As well as reducing the implementation burden on the compiler developers, these constraints also keep application programmers from using non-scalable features that inhibit good performance on a GPU.

3 CREATING A STATEMENT OF WORK

It was understood at the start of the engagement that some compromises would be needed to realize the vision of having many OpenMP scientific applications run successfully with high performance on Perlmutter. Some of the challenges include

- Only a subset of the NERSC workload could successfully use GPUs. A 2018 NERSC workload study showed that the applications consuming approximately 50 percent of NERSC compute time either partially or fully supported NVIDIA GPUs [1]. This covered tens of applications but there were still hundreds of applications, especially those in the "long-tail", with zero or unknown GPU-readiness. This indicated that there was still an ongoing challenge to use heterogeneous computing whatever the programming approach.
- The use of OpenMP on GPUs had not yet demonstrated broad success. There had been individual case studies that demonstrated some level of success [12, 57], however, the HPC community often reported a lack of OpenMP compiler maturity [48].
- The OpenMP specification was still rapidly evolving to support accelerators. At this time the OpenMP-5.0 specification was still a work in progress; the preview OpenMP-5.0 technical reports had been released and indicated some significant new features and improvements for accelerators.
- There was only a 3 year time-frame before Perlmutter was made available to users.

These challenges motivated a SOW that emphasized a research and development partnership. There were uncertainties in how to use OpenMP successfully on GPUs, a lack of OpenMP GPU-offload applications, and no experience with new OpenMP-5.0 features. It was important to address these challenges together and not structure the SOW to mandate that all OpenMP features would be implemented for the GPU. We agreed to a contract SOW consisting of seven milestones over a period of three years. The contract was

structured as a fixed-price contract with a dollar amount associated with every milestone in the SOW. Milestone payments only happen when the NERSC technical representative is satisfied that a deliverable meets the requirements of a given milestone. Milestones cannot be skipped because later milestones build upon earlier milestones. It is undesirable, but acceptable, for NVIDIA to submit a deliverable later than the target milestone date. If required, contract milestones could be modified upon agreement by the technical and contractual representatives of NERSC and NVIDIA. In the remainder of this section, we describe the contract SOW milestones and the thinking behind them.

Milestone 1 (Apr 2019): Requirements Document and Test Suite. Deliver a requirements document consisting of the subset of OpenMP-5.0 features to be implemented as well as a rationale for those features that considered application requirements and the priorities of NERSC staff. Here, NERSC staff priorities also considered the opinions of colleagues from other DOE laboratories who had significant experience using OpenMP target offload with various compilers. This gave reassurance that the most critical features were included and that new OpenMP-5.0 features were appropriately prioritized. The milestone also required that NERSC/NVIDIA work together to create an OpenMP target offload application test suite to demonstrate compiler correctness and performance. The test suite was used as an information exchange vehicle between the user community and NVIDIA, allowing determination of the OpenMP features used in today's applications and influencing the proposed list of supported OpenMP features.

Milestone 2 (Aug 2019): Application Selection. Select five NESAP-2 teams to partner with to add OpenMP target offload to their applications. The task involved NVIDIA proposing an initial OpenMP GPU porting plan for each application, which was subject to approval by NERSC and the respective NESAP-2 team. The selected applications were required to be diverse, i.e. cover a range of scientific domains, use different programming languages, and have different levels of GPU-readiness. This milestone was intended to make NVIDIA familiar with the challenges facing DOE applications and focus compiler development effort on the most pressing challenges of real applications. It also enabled NVIDIA engineers to influence how NESAP-2 teams used OpenMP in applications in the same way as the activities occurring in a Center of Excellence (COE) [39]. The milestone was important because there were no NESAP-2 OpenMP target offload applications at the beginning of the project. Many NESAP-2 teams were interested in the possibility of using OpenMP target offload but were reluctant to invest time until there was a robust OpenMP compiler that delivered high performance. This milestone ensured that there would be concurrent compiler and application development. It was not part of the first milestone because the participants of the NESAP-2 program were not known until after the first milestone.

Milestone 3 (Oct 2019): Alpha Compiler and Feature Timeline. Deliver an initial compiler which successfully compiles and executes C, C++ and Fortran OpenMP target offload versions of the Stream microbenchmark and provide a timeline when all agreed upon OpenMP features would be implemented. We put the feature timeline into this milestone so that the timeline could take into account the requirements of the five NESAP-2 teams and any updated priorities from NERSC. At NVIDIA's request, this compiler would

only be shared with NERSC staff and the five NESAP-2 teams under a Non-Disclosure Agreement (NDA).

Milestone 4 (Apr 2020): Closed Beta Compiler and Updated Test Suite. Deliver an improved NDA compiler, document the improvements over the Alpha compiler, and work with NERSC to add new applications to the test suite. It was a strategic decision to split test suite creation into two milestones because application development efforts within NERSC and across the DOE were happening at the same time as compiler development. It was important to take advantage of these new applications, some of which were using OpenMP-5.0 features.

Milestone 5 (Nov 2020): Open Beta Compiler and Training Event. Deliver an improved compiler that is accessible to all NERSC users and provide a training event. The NERSC training event was used to share key information about what features were implemented, how to use the compiler, and best practices. It gave NVIDIA the opportunity to advise NERSC users about the best way to use their compilers.

Milestone 6 (May 2021): First Production Compiler. Deliver the first production compiler and a publicly available document about the progress and lessons learned from the five NESAP-2 applications as well as the state of all OpenMP features in the compiler. The compiler must successfully compile and execute every single application in the test suite as well as the five NESAP-2 applications, subject to those OpenMP features being implemented in the compiler. The OpenMP target offload test suite applications must achieve at least 90% of the performance of a corresponding OpenACC application on NERSC's Cori-GPU system. This milestone was important because it gave NERSC evidence that the compiler is production-ready.

Milestone 7 (Nov 2021): Final Production Compiler. Deliver the final production compiler which implements all features specified in the requirements document. The compiler must achieve the same correctness and performance targets as the Milestone 6 compiler but this time on Perlmutter GPU nodes. Deliver a publicly available document similar to the Milestone 6 document but applicable to the Milestone 7 compiler.

4 PROJECT MANAGEMENT

In this section we explain some of the organizational aspects of the NRE project. We had two different weekly half-hour meetings for the duration of the project. The first meeting was between the contract technical representatives of NERSC and NVIDIA. The meeting involved discussing SOW requirements and expectations, NVIDIA progress towards milestones, and refined prioritization of work based on compiler issues encountered and new application requirements. The second meeting was between NERSC staff involved in the project and NVIDIA Developer Technology Engineers (DevTechs). It occasionally included people associated with the five selected NESAP-2 projects. The purpose of this meeting was to coordinate on the NERSC/NVIDIA test suite, share NESAP-2 application progress, and understand compiler correctness and performance issues affecting applications. The DevTechs filed internal issues and discussed application priorities with the compiler team.

The project was organized to be collaborative in nature. For example, NERSC reported compiler issues directly to the compiler

team by email rather than through a bug tracking system. This required trust on both sides that direct communication would not be harmful to project progress. It also included ad-hoc meetings between NERSC and the NVIDIA compiler team to discuss implementation issues and feature tweaks based on updates in the OpenMP-5.1 specification. Throughout the project, NERSC/NVIDIA engaged in activities to raise the profile of the compiler and give confidence to users that OpenMP is a viable way to use the Perlmutter GPUs. This included joint presentations at GPU Technology Conference (GTC) [10, 11], an OpenMP training event at NERSC in December 2020 [44], and a strong NVIDIA presence at the ECP SOLLVE OpenMP hackathon at NERSC in January 2021 [49].

We ensured that multiple NVIDIA staff had access to the NERSC Cori-GPU system used to evaluate the NVIDIA compilers. This allowed easy sharing of applications and enabled NVIDIA to reproduce issues that only happened on the Cori-GPU system. It was also the platform used to assess the compiler correctness and performance in multiple milestones. This removed any ambiguity and enabled NVIDIA staff to test private compiler snapshots prior to compiler deliverable milestones in their own home directories. NERSC made the early access compilers as widely available as possible while still respecting confidentiality agreements. This was achieved on Cori-GPU by using an Access Control List (ACL) to manage individual user access on a system available to many NERSC users. Information about the state of the current compiler was shared in a restricted access Google Drive folder.

5 THE OPENMP TEST SUITE

The OpenMP test suite was a requirement of the first contract milestone. It allowed NERSC and NVIDIA to track compiler correctness and performance over time. The effort involved significant work to collect available OpenMP target offload applications and then incorporate the applications into an automated test suite environment. It was a challenge to merely collect the applications because of a lack of publicly available OpenMP target offload benchmarks at the time. NERSC and NVIDIA therefore worked together to flesh out and increase the value of the test suite by incorporating early NERSC and other DOE application porting efforts. This was a key step because the people involved in each NERSC porting effort always encountered compiler maturity issues whatever compiler they used. Therefore, the inclusion of early NERSC porting efforts gave confidence that the compiler would better support future OpenMP applications expected from NERSC users. The applications had various levels of tuning for GPUs. Many of these applications also had equivalent OpenMP target offload and OpenACC ports. Our performance criteria of an OpenMP application achieving at least 90% of the corresponding OpenACC application performance gave evidence that the compiler is competitive whether or not an application has been extensively tuned for GPUs. The initial tests are shown in Table 1 and explained below. In the table, the performance criteria column indicates whether an OpenMP test has a corresponding OpenACC test.

The SOLLVE V&V [17] tests are a collection of correctness tests in C, C++ and Fortran which evaluate whether OpenMP features are implemented in a standard conforming way [18]. At the time of test suite creation, the SPEC ACCEL [50] benchmark suite was the other

obvious source of OpenMP target offload benchmarks in both C and Fortran languages [7]. Each OpenMP target offload benchmark in SPEC ACCEL has a corresponding OpenACC implementation, allowing them to be used in our performance evaluation.

We added 3 micro-benchmarks named Stream [37], Transpose [33] and Laplace [8] to measure streaming memory bandwidth, strided memory bandwidth, and OpenMP reduction performance, respectively. We created C and Fortran OpenMP offload versions of Stream and added a C++ version named BabelStream [16]. This gave coverage of the three base languages and a demonstration that the same peak GPU memory bandwidth could be obtained in C, C++ and Fortran. Transpose is a matrix transpose benchmark providing multiple implementations of naïve and tiled matrix transposes. The tiled matrix transposes require parallelism on different loops and provide a simple way to test OpenMP directives split across loops and any associated performance overheads. Laplace is a simple benchmark implementing Jacobi's method to solve the matrix equation. It is useful because each iteration of the solver requires an OpenMP reduction over grid points. This can become a bottleneck in OpenMP compilers [14]. We test multiple grid sizes including a very small grid size that is sensitive to OpenMP target region latency. Finally, we added a version of each of these micro-benchmarks where data is allocated in CUDA Managed Memory via the CUDA API function `cudaMallocManaged` and OpenMP map clauses are omitted. These provide very simple tests of interoperability between OpenMP and CUDA data management.

The BerkeleyGW General Plasman Pole (GPP) [21] and Full Frequency (FF) [21] mini-apps are written in C++ and use a custom array class of complex numbers on the GPU. They provide a good test of non-trivial mapping of data between CPU and GPU. The Flow [34], Neutral [36], and Hot [35] mini-apps are created by the University of Bristol High Performance Computing group. They provide OpenMP target offload and OpenACC implementations and are familiar to NVIDIA staff involved in this project. The ToyPush [28] and ElectromagneticPIC [38] mini-apps are both Particle In Cell (PIC) mini-apps that use Fortran OpenMP target offload. These two applications are non-trivial and are important to demonstrate the capabilities of the NVIDIA Fortran compiler. The ElectromagneticPIC mini-app has additional requirements, including the ability to use CUDA allocated data in OpenMP target regions and to mix CUDA kernels and OpenMP target regions in the same application. Finally, miniQMC [25] is a Quantum Monte Carlo code written in C++. It uses C++ complex numbers and tests more advanced OpenMP features, including multiple CPU OpenMP threads launching OpenMP target regions and the use of the OpenMP `nowait` clause to launch OpenMP target regions asynchronously.

We extended the test suite in milestone 4 to include the tests shown in Table 2. These tests use OpenMP target offload in ways not covered by the initial test suite and add significant value to the test suite. The GAMESS-rimp2 [24] suite of mini-apps were added to test the use of CUDA math libraries through `cuBLAS`, `cuBLASxt` and `NVBLAS` APIs alongside application use of OpenMP target offload. We also added applications with non-trivial data management based on NERSC's difficulties creating OpenMP target offload versions of HPGMG [51] and TestSNAP [52]. For example, HPGMG exposed data management weaknesses in all compilers except LLVM/Clang.

Table 1: The initial NERSC/NVIDIA test suite

Test Name	Languages	Details	Perf. Criteria
SOLLVE V&V Suite [17]	C, C++, Fortran	Correctness suite consisting of more than 100 tests	N
SPEC ACCEL V1.2 [50]	C, Fortran	Benchmark suite consisting of 15 mini-apps	Y
Stream [16, 37]	C, C++, Fortran	Benchmark to measure peak memory bandwidth	Y
Transpose [33]	C	Benchmark to measure peak memory bandwidth	Y
Laplace [8]	C	Benchmark to measure reduction performance	Y
BerkeleyGW-GPP [21]	C++	Material science. Proxy app for BerkeleyGW	Y
BerkeleyGW-FF [21]	C++	Material science. Proxy app for BerkeleyGW	Y
Flow [34]	C	2D hydrodynamics	Y
Neutral [36]	C	Monte Carlo Neutron Transport	Y
Hot [35]	C	Heat diffusion	Y
Toypush [28]	Fortran & C	Particle In Cell. Proxy app for push phase of XGC	Y
ElectromagneticPIC [38]	Fortran & C++	Particle In Cell. Proxy app for WarpX	Y
miniQMC [25]	C++	Material science. Proxy app for QMCPACK	N

We found that mapping data in nested C structs containing pointers and double pointers from the host to the device led to either compile-time or run-time failures [12]. Similarly, TestSNAP exposed failures related to mapping complex C++ objects that contained custom array objects with dynamically allocated data. We added the SU3_Bench [19] microbenchmark because `std::complex` calculations in OpenMP target regions as well as data management with `std::complex` variables caused issues in many OpenMP compilers. Finally, the RAJA OpenMP target offload Performance Suite [47] provides multiple tests of the RAJA framework which is an important DOE framework. These performance tests ensure that user lambda functions can be executed on the device in the same way as needed in the Kokkos OpenMP target offload backend [20].

The final test suite in milestone 4 consisted of 312 tests: 206 SOLLVE V&V tests, 15 SPEC ACCEL tests, 42 RAJA Performance Suite tests, and 49 benchmark and proxy application tests. We sometimes compiled or executed the same benchmark / proxy application in different ways to create multiple tests. This is an extensive test suite that has been curated to exercise a wide variety of OpenMP features and interoperability use cases needed in production applications. As of August 2021, the NVIDIA compilers are on track to correctly execute the full test suite and meet the performance requirements on Perlmutter. The first production compilers, which correspond to the compilers distributed in the NVIDIA HPC SDK 21.5, obtain a 98% pass rate on Cori-GPU.

6 AREAS OF CONTINUAL ENGAGEMENT

This section explains the continual engagement between NERSC and NVIDIA on areas of GPU data management, performance portability to CPU and GPU, features to support in OpenMP target regions, and interoperability. These were ongoing activities that evolved according to the requirements from the test suite and NESAP-2 applications, refinements in the OpenMP-5.1 specification, and experiences from other OpenMP target offload activities occurring within the HPC community.

6.1 GPU Data Management

One of the more challenging aspects of GPU-programming is managing the data in host and device memory. This is supported in OpenMP via the `map` clause, the `declare target` directive and various API functions. In particular, NERSC users encountered problems with compiler support for data management in their early attempts to target GPUs. Two problematic examples stand out: mapping structures containing pointers to dynamically allocated data and mapping pointers to pointers to dynamically allocated data (i.e. double pointers). It was not until the OpenMP-5.0 specification that language existed that stated the pointer attachment rules that would make these use cases work in the way users expected. This was therefore an important capability that would enable many non-trivial OpenMP target offload applications.

We made this capability one of the highest priorities. NERSC shared standalone tests <100 lines of code that illustrated expected data management. These test cases were extracted from real applications including HPGMG and TestSNAP. This helped prioritize the most important cases for the compiler team and enabled them to validate their implementation. It also let NERSC and NVIDIA DevTechs know when the capability was in place that could enable the real applications. NVIDIA added the problematic HPGMG and TestSNAP applications to the test suite in milestone 4 to demonstrate correct OpenMP-5.0 data management in real applications. Finally, we also ensured that early compilers provided the capability to allocate data in Managed Memory. This was both a backup option while the OpenMP-5.0 data management capability was being developed and an option in itself to simplify data management for new users or for experienced users with complicated data structures.

6.2 Performance Portability to CPUs and GPUs

In CPU-only OpenMP programs, the program creates a single team of threads using the `parallel` construct and uses the `for/do` constructs to workshare the iterations across those threads. To take advantage of the higher degree of parallelism in GPUs, OpenMP added a `teams` construct to create multiple teams of threads. When teams are created, only the primary thread is active until it encounters a `parallel` construct, which activates the other threads in the

Table 2: The additions to the NERSC/NVIDIA test suite

Test Name	Languages	Details	Perf. Criteria
GAMESS-rimp2 [24]	Fortran	Material science. Proxy app for GAMESS	Y
HPGMG [51]	C	Geometric Multigrid solver benchmark	Y
RAJA Perf. Suite [47]	C++	Benchmark suite consisting of more than 40 tests	N
SU3_Bench [19]	C++	QCD. Proxy app for MILC	Y
TestSNAP [52]	C++	Material science. Proxy app for LAMMPS	N

team. There are therefore three different directives that can be used to create and control parallelism: `teams`, `parallel`, and `simd`. The NVIDIA compiler maps these directives to CPU and GPU as shown in Table 3.

Table 3: OpenMP-4.5 parallelism constructs and mapping to GPUs and CPUs

Directive	GPU	CPU
<code>teams</code>	thread blocks	sequential: <code>num_teams(1)</code>
<code>parallel</code>	threads	threads
<code>simd</code>	sequential: <code>simdlen(1)</code>	hint for vector inst.

It is important to understand the mapping because it can have an impact on performance portability to CPUs and GPUs. This is because highest performance on a CPU generally involves exploiting coarse-grained parallelism on the outermost loop to improve data cache reuse. In contrast, highest performance on a GPU generally requires exploiting additional fine-grained parallelism on the innermost loops to effectively use the large number of GPU threads which have limited cache memory per thread. To exploit the massive parallelism of the GPU with this mapping, a programmer should use `teams` and `parallel` as shown Listing 1. However, this is non-optimal on the CPU because there is only parallelism on the inner loop. This highlights a performance portability concern for a single compiler. Unfortunately, things become even more complicated when considering other compilers that may use a different mapping than Table 3.

Listing 1: An example of using OpenMP-4.5 parallelism constructs on multiple loops

```
#pragma omp target teams distribute
for (int j=1; j<n-1; j++) {
  #pragma omp parallel for simd
  for (int i=1; i<n-1; i++) {
```

NERSC/NVIDIA pursued two ways of improving performance portability in the NRE project. In Section 6.2.1, we discuss the OpenMP-5.0 loop construct and in Section 6.2.2 we discuss code specialization with the metadirective and declare variant directives.

6.2.1 Performance Portability with the loop construct. NVIDIA was a key contributor to the loop construct in the OpenMP-5.0 specification. The loop construct asserts that loop iterations are independent and enables the compiler to create additional parallelism

to execute the body of the loop. The NVIDIA compiler team views the loop construct as the best way to achieve performance portability to CPUs and GPUs. The ability to create additional parallelism enables the construct to map to both thread blocks and threads on the GPU without the need for an explicit `parallel` construct as shown in Table 4. This means that the multiple loop code fragment can be written as shown in Listing 2.

Table 4: OpenMP-5.0 loop construct and mapping to GPUs and CPUs

Directive	GPU	CPU
<code>loop bind(teams)</code>	thread blocks & threads	threads
<code>loop bind(parallel)</code>	threads	threads
<code>loop bind(thread)</code>	sequential: <code>simdlen(1)</code>	hint for vec. inst.

Listing 2: An example of using the OpenMP loop construct on multiple loops

```
// "loop bind(teams)" can also be used
#pragma omp target teams loop
for (int j=1; j<n-1; j++) {
  #pragma omp loop
  for (int i=1; i<n-1; i++) {
```

The loop feature is attractive for high performance because the compiler can run this program as Single Program Multiple Data (SPMD) which is suitable for the GPU. Programs written with the `parallel` construct cannot always be run as SPMD for a couple of reasons. First, there are OpenMP APIs and other constructs such as `single`, `critical`, `master` that require the compiler to generate more complex code and use a more complex runtime to support them, leading to overhead. Second, if the `parallel` construct is used in a function for which the compiler lacks visibility, it can be a challenge for the compiler to generate an efficient program as SPMD.

NERSC did not have experience with the loop construct prior to the NRE contract because there was no high quality implementation in any compiler. As such, NERSC did not initially view it as a high priority feature. However, a key aspect of the partnership was to enable NVIDIA to advise NERSC about the best way to use OpenMP on GPUs. NVIDIA described the performance benefits and also enhanced the loop construct compared to the OpenMP specification in two ways. Firstly, NVIDIA added support for atomic operations within a loop region. This is not supported according

to the OpenMP specification but is needed in some NERSC applications performing sparse atomic updates. Secondly, NVIDIA added an auto-parallelization feature that gives the compiler the freedom to automatically select parallelism to help new users migrate to massively parallel GPUs. NERSC/NVIDIA have since worked together to introduce the loop construct to NERSC users at an OpenMP training event at NERSC in December 2020 [44] and an ECP SOLLVE OpenMP hackathon at NERSC in January 2021 [49].

6.2.2 Performance Portability with `declare variant` and meta-directive `directives`. The `declare variant` and `metadirective` features were designed to address challenges with performance portability. The features enable code specialization based on compiler, architecture or user conditions. At the start of the project, NVIDIA proposed a partial implementation of these features based on an assessment of implementation difficulty and usefulness. During the NRE project, it was found that the OpenMP-5.0 definition of these features had an ambiguity that was corrected in the OpenMP-5.1 specification. This warranted revisiting the support to be implemented in the NVIDIA compiler. A strength of the partnership has been open conversations around code examples that are motivated from NERSC applications. NERSC/NVIDIA agreed to a revised OpenMP-5.1 subset for `declare variant` and `metadirective` features. This will enable the `metadirective` feature to be used to select different directives for CPU and GPU as shown in Listing 3.

Listing 3: An example of using the OpenMP `metadirective` directive on multiple loops

```
#pragma omp metadirective \
  when( target_device={kind(gpu)}: \
        target teams distribute ) \
  default( parallel for )
for (int j=1; j<n-1; j++) {

#pragma omp metadirective \
  when( device={kind(gpu)}: parallel for ) \
  default( simd )
  for (int i=1; i<n-1; i++) {
```

6.3 Features to Support in OpenMP Target Regions

The OpenMP specification provides a Normative References section that lists the supported base language standards, e.g. C99. This section also lists any features provided by the base language that have unspecified behavior when used in an OpenMP application. The OpenMP standards committee has done a very thorough job of making the list of features that should be avoided in compliant applications as small as possible. However, it presents a challenge for compilers, especially compilers offloading code sections to GPUs, because the list of all possible features that could be used in the base language is huge. NERSC and NVIDIA prioritized the supported features according to those used in existing applications and planned to be used in new applications. We created an extensive test suite and closely collaborated on the five NESAP-2 applications to capture the features used in real applications' offloaded code sections.

The early NERSC experience with OpenMP target offload often used the LLVM/Clang compiler [32] on Cori-GPU. Progress was impeded by the inability to use math functions declared in `cmath` and `math.h` header files in OpenMP target regions. This was not fixed until LLVM/Clang 11.0.0 [31]. This was often an impediment for new users and an inconvenience for more experienced users which could only sometimes be worked around with reasonable effort. NVIDIA made this a high priority and provided the support in the Alpha compiler.

In contrast with math function support that was emphasized at the outset, the importance of early `std::complex` support was only understood part way through the contract. NERSC knew that many applications used complex numbers, including the BerkeleyGW-GPP and miniQMC C++ mini-apps that were in the initial test suite. However, issues with `std::complex` were initially missed because BerkeleyGW-GPP used a custom complex number class and miniQMC was a challenging mini-app which did not compile for other reasons until NVIDIA HPC SDK 21.3. The focus on applications in the NRE contract enabled NERSC and NVIDIA to prioritize `std::complex` support as soon as the deficiency was found. It also helped us find other related deficiencies. For examples, BerkeleyGW-GPP performed complex number OpenMP data reductions as separate scalar reductions for real and imaginary complex number components instead of using a User Defined Reduction (UDR) with the `declare reduction` capability. This was a developer workaround for immature compilers. The `declare reduction` feature was not in the agreed subset of features, however, we knew that miniQMC and other applications required `std::complex` OpenMP data reductions. NVIDIA implemented native support for `std::complex` OpenMP data reductions midway through the contract and it is currently being used successfully by the miniQMC application.

The benefit of focusing on real applications is that they sometimes use features in different ways than standalone kernels. For example, many of our Fortran applications failed initially with early NVIDIA compilers because the compilers were unable to use Fortran automatic arrays in OpenMP target regions. This issue was not found by SOLLVE V&V tests because the V&V suite consists of the simplest possible tests to assess the correctness of an OpenMP feature; it would not make sense to create a subroutine containing a Fortran automatic array because this would add unnecessary complexity to the SOLLVE V&V test. The use of actual applications also identified issues with using data and functions defined in different compilation units.

6.4 Interoperability

Many DOE applications use multiple programming models or make use of scientific math libraries. Ensuring that such applications work correctly requires significant consideration of the interoperability requirements. We added several applications to the test suite to ensure expected behavior when mixing OpenMP target offload and CUDA. For example, ElectromagneticPIC uses CUDA and also passes CUDA-allocated data to OpenMP target regions by specifying the variables in the OpenMP `is_device_ptr` clause. As another example, GAMESS-rimp2 uses the cuBLAS math library by specifying the OpenMP-allocated variables in the OpenMP `use_device_ptr` clause. It is likely that many application teams

will use these capabilities for one of two reasons. Firstly, it allows incremental porting of CUDA applications to portable OpenMP target offload. Secondly, it allows OpenMP target offload applications to be optimized for NVIDIA GPUs by using specialized CUDA implementations or CUDA math library calls for a small number of performance critical functions.

NERSC and NVIDIA have also collaborated on more advanced interoperability use cases. The NVIDIA compiler includes a CUDA stream get/set API extension to easily launch an OpenMP target region on a particular stream to meet a BerkeleyGW use case [5]. NVIDIA has also enabled interoperability of OpenMP with OpenACC, C++ standard parallelism, and Fortran do concurrent to support future use cases likely to emerge from NERSC and other HPC user communities. Finally, one interesting use case emerged during the project which required appropriate handling. NERSC found that two NESAP-2 application teams wanted to mix Python with OpenMP target offload. This included the ImSim Batoid [26] application which is written in Python and uses functions in a shared C++ library. We found that the NVIDIA compiler did not initially support using OpenMP target offload in a shared library. The partnership enabled NERSC to prioritize fixing this issue to enable the NESAP-2 teams to continue to make progress in time for the Perlmutter supercomputer.

7 CASE STUDIES

In this section, we provide various case studies demonstrating the success of the OpenMP target offload capability in the NVIDIA compilers. The case studies include the BabelStream micro-benchmark and some applications selected for application engagement [13]. Unless otherwise stated, we used the NVIDIA HPC SDK 21.7 compiler on Cori-DGX. We highlight compiler capabilities used by the individual applications and performance achieved.

7.1 BabelStream

The BabelStream micro-benchmark contains several kernels to measure achievable memory bandwidth. The benchmark is important to ensure that applications can obtain close to the peak memory bandwidth on the target hardware. It includes OpenMP-4.5 and CUDA versions. We added an OpenMP loop version to BabelStream by replacing `target teams distribute parallel for simd` with `target teams loop`. Figure 1 shows the fraction of peak memory bandwidth obtained in Triad and Dot Product kernels.

The results show that all implementations of both kernels achieve between 80% and 90% of the published NVIDIA A100 GPU memory bandwidth. All 3 implementations of the Triad kernel achieve approximately the same performance. The Dot Product kernel is a more complicated kernel that includes a data reduction. The results show that both OpenMP implementations achieve higher performance than the CUDA implementation. This is an impressive result considering the BabelStream CUDA reduction kernel uses an efficient implementation of a tree-based parallel reduction algorithm with intermediate results stored in GPU shared memory. It is also well known that many OpenMP compilers deliver poor OpenMP reduction performance [14].

Figure 1: Percentage of peak memory bandwidth for BabelStream Triad and Dot Product kernels on Cori-DGX. The percentage of peak is calculated using a peak memory bandwidth of 1,555 GB/s [42]. Results are shown for CUDA, OpenMP-4.5 and OpenMP loop versions.

7.2 BerkeleyGW-GPP

BerkeleyGW is a Fortran material science application which has recently been GPU-accelerated with CUDA [5]. Over the years, the BerkeleyGW team and staff at NERSC have experimented with GPU-offload ports of BerkeleyGW. In fact, the BerkeleyGW OpenMP offload mini-apps in the test suite were ported entirely from Fortran to C++ to enable experimentation with more mature C/C++ OpenMP offload compilers [23]. The BerkeleyGW team are interested in migrating their Fortran + C interface + CUDA code to Fortran OpenMP target offload to improve maintainability and portability. We created a Fortran OpenMP target offload port of the GPP mini-app [4] based on an earlier OpenACC port [58]. The benchmark includes OpenMP-4.5 and OpenMP loop versions. The key characteristics of this code are structured data management using a `target data` directive, Fortran complex numbers, OpenMP reductions over complex numbers, and collapsed loops parallelized over teams and threads.

The OpenMP-4.5 and OpenMP loop versions of the code achieve 4.31 TF/s on a single A100 GPU. This is 44% of the double precision floating point peak (without tensor cores) on an A100 GPU [42]. This is a significant accomplishment for two reasons. Firstly, it demonstrates that the NVIDIA compiler provides significant support for Fortran applications. Secondly, it demonstrates that the compiler can generate high quality code even for highly compute-bound kernels.

7.3 Kokkos OpenMP target offload backend

Kokkos [20] is a C++ based programming framework that supports multiple backends for CPUs and different vendor GPUs. It includes GPU backends implemented in the native architecture specific programming frameworks such as CUDA or SYCL. It also includes a backend under development named OpenMPTarget in which the Kokkos API functions are implemented using OpenMP target offload directives and API functions. NERSC/NVIDIA worked closely on the concurrent development of the compiler and the Kokkos OpenMPTarget backend.

In Kokkos, there is a programming approach where the user application defines functions that operate on Kokkos-managed data.

This requires the mapping and execution of user-defined functions in OpenMP target regions. We ensured that this was supported in the compiler, and ready to use in the OpenMPTarget backend, by adding the OpenMP target offload version of the RAJA Performance Suite to the test suite. Both Kokkos and RAJA frameworks depend on this capability. NERSC and NVIDIA also worked closely on other eventual requirements, including explicitly managing data on the CPU and GPU using `omp_target_alloc`, `omp_target_memcpy` and `omp_target_free` API functions, enabling manual worksharing of loop iterations across OpenMP teams using `omp_get_num_teams` and `omp_get_team_num` API functions, and supporting the use of OpenMP `parallel` and `for` directives in different functions. Finally, we worked together to understand the requirements of the Kokkos `team_size` abstraction. This abstraction enables the programmer to specify the number of threads to use in an OpenMP team. It is implemented in the OpenMPTarget backend using the OpenMP `thread_limit` clause. The OpenMP specification allows the compiler to use a lower thread limit than requested, however, for Kokkos it is beneficial to get the requested value unless it would cause resource exhaustion. NVIDIA provided a refined implementation during the project that better met the thread control requirements in Kokkos.

Kokkos provides a testing infrastructure for backend developers. The NVIDIA HPC SDK 21.7 compiler successfully executes 10 out of 17 levels of Kokkos incremental tests when using the OpenMPTarget backend. The 10 levels consist of 17 individual unit tests. This is sufficient functionality to run some applications, including the TestSNAP application described in Section 7.4 and the XGC-Collision application described in Section 7.5.

7.4 LAMMPS - TestSNAP

LAMMPS [54] is a molecular dynamics application written in C++ and Kokkos. The NERSC/NVIDIA intent is to be able to execute the full LAMMPS application using Kokkos with the OpenMPTarget backend. This would demonstrate a large suite of OpenMP offload capabilities as well as the ability to compile a large C++ application. TestSNAP [22] is a proxy application that captures the computational requirements of LAMMPS. The NVIDIA compiler correctly executes the OpenMP target offload version of TestSNAP and the Kokkos version of TestSNAP with the OpenMPTarget backend. The full LAMMPS application using Kokkos with the OpenMPTarget backend is still a work in progress.

The OpenMP target offload version of TestSNAP [52] has non-trivial data management because of the use of a C++ object containing multiple dynamically allocated arrays. In TestSNAP, this C++ object is deep copied from CPU to GPU using multiple OpenMP `target enter` data directives: first the C++ object is mapped to the GPU, and then each dynamically allocated array is mapped to the GPU. This requires OpenMP-5.0 pointer attachment rules to automatically attach the GPU array data to the GPU object pointers. NERSC and NVIDIA prioritized this compiler capability to enable progress on TestSNAP. We also created a Kokkos version of TestSNAP [53] to help evaluate the Kokkos OpenMPTarget backend. The performance of a TestSNAP problem with 205 bispectrum

coefficients for the OpenMP target offload version, the Kokkos version with the CUDA backend, and the Kokkos version with the OpenMPTarget backend is shown in Figure 2.

Figure 2: TestSNAP grind time when executing on Cori-DGX. There are three comparable implementations of TestSNAP: OpenMP target offload, Kokkos using the CUDA backend, and Kokkos using the OpenMPTarget backend. The Kokkos OpenMPTarget backend results were obtained using NVIDIA HPC SDK 21.3 because of a performance regression in NVIDIA HPC SDK 21.7.

The results show that the OpenMP target offload version is only marginally slower than the Kokkos version using the CUDA backend. There is more of a performance difference between the two Kokkos backends. This indicates an abstraction penalty in the OpenMPTarget backend that is not present in the CUDA backend. NERSC/NVIDIA are currently trying to understand the performance difference between Kokkos backends. NERSC/NVIDIA found that TestSNAP executed 1.3x faster when using HPC SDK 21.3 compared to HPC SDK 21.7 on Cori-DGX. Analysis showed that the compilers make different trade-offs between the compiler optimizations to use on heavily templated C++ applications and shorter compilation time. NVIDIA are currently investigating whether the heuristics in upcoming compilers can be improved.

7.5 XGC-Collision

XGC is a PIC code used to simulate magnetically-confined plasma in a Tokamak fusion reactor [29]. The XGC-Collision proxy application is written in C++ and Kokkos. Performance profiles showed that the implementation only used a small fraction of the GPU hardware capability. This was related to a design decision where only a coarse-grained outer loop (with a loop trip count of 930) was parallelized with Kokkos. NERSC/NVIDIA worked with the XGC team to prototype fine-grained parallelism on inner loops using OpenMP target offload [49]. At the time, there were issues using XGC-Collision with the Kokkos OpenMPTarget backend and NVIDIA HPC SDK 20.11. Therefore, this activity depended on interoperability between the Kokkos CUDA backend and OpenMP target offload. OpenMP was key because the XGC developers did not know a way to perform data reductions over multiple scalar variables using Kokkos. The performance results of the original Kokkos implementation and the prototyped OpenMP-4.5 and OpenMP loop versions for the two dominant XGC kernels are shown in Figure 3.

Figure 3: Time spent in two XGC kernels on Cori-DGX. The initial Kokkos implementation used coarse-grained parallelism only and the two tuned OpenMP implementations exploited additional fine-grained parallelism.

The results show that both fine-grained implementations improve upon the initial performance¹. This emphasizes just how important it is to provide sufficient parallel work on GPUs. The implementations benefited from highly-tuned OpenMP reductions, as was also seen with the Dot Product results in Section 7.1. This ensured that frequent OpenMP reductions in the inner loops did not adversely impact performance. In contrast to Sections 7.1 and 7.2, where there was no performance advantage to using OpenMP loop, XGC-Collision performance improves when using OpenMP loop. This is because XGC-Collision is the only case-study application that uses the hard-to-optimize code pattern shown in Listing 1. The OpenMP loop version has a performance advantage because it avoids the parallel fork-join in the frequently encountered inner loops. Performance is also helped because the NVIDIA compiler can generate simpler code and use a lightweight OpenMP runtime because of the restrictions and parallelism guarantees of the loop construct. The XGC developers are in the process of trying to replicate the fine-grained parallelism strategy using Kokkos only. The focus on interoperability in the NVIDIA compiler made it possible for these prototyping activities that demonstrated that significant performance improvements in XGC are possible. NERSC/NVIDIA have also successfully executed XGC-Collision with the Kokkos OpenMPTarget backend and NVIDIA HPC SDK 21.7.

8 CONCLUSIONS AND LESSONS LEARNED

This paper described how NRE investments by HPC Centers can address gaps in supercomputer hardware or software technologies. This particular NRE investment funded NVIDIA to deliver a production OpenMP GPU-offload compiler for Perlmutter, the first GPU-accelerated supercomputer at NERSC. Test suite results and progress on NESAP-2 applications indicate that the compiler will meet functional and performance targets. The contract was organized to give the subcontractor the freedom to recommend solutions for NERSC but also the responsibility to understand and satisfy NERSC users' requirements. This approach served to prioritize

useful features, increasing the likelihood that the NRE investment would yield a long-term benefit to the OpenMP community. As of November 2020, NVIDIA publicly provides OpenMP GPU-offload compiler capability in the NVIDIA HPC SDK.

There was a strong emphasis on partnership and open dialog between NERSC and NVIDIA. NERSC staff know user application requirements, have experience using OpenMP target offload, and are aware of user pain points programming GPUs. NVIDIA staff have a sense of the broader market value of compiler capabilities, understand the engineering burden, and know what has worked well and poorly with OpenACC. It is clear that each party has complementary expertise. Our partnership worked but involved effort from both parties to agree on the best path forward. Similar NRE contracts may not have the same level of success unless both parties are engaged, willing to discuss issues in detail, see the perspective of the other party, and are open to compromise. Our most fruitful exchanges generally involved discussions focused around real code examples, often coming from test suite or NESAP-2 codes. This allowed us to motivate the importance of certain OpenMP features and have healthy conversations about whether code could or should be written in a different way.

A lesson learned is that sufficient time should be spent upfront to ensure that the SOW captures the most important requirements. Our approach of minimizing the project scope by focusing on a subset of OpenMP features bounded the work and gave both parties confidence that milestone requirements could be met. There has been no need to change project scope. Milestones 1-4 were successfully completed on time, milestone 5 was successfully completed one month later than targeted to align with NVIDIA commercial compiler releases, and milestone 6 was successfully completed two months later than targeted. The only change has been to prioritize compiler stability and performance over features in milestone 5. This mutually-agreed prioritization occurred months before the milestone target date. It was informed by weekly meetings as well as NERSC spending sufficient time to experiment with and understand capabilities and limitations of early compiler deliverables.

We recommend that HPC Centers consider their unique needs before pursuing similar NRE activities. The HPC Center should perform a cost-benefit analysis to understand whether NRE money would be better spent elsewhere, e.g. purchasing additional hardware. This analysis should also consider the long-term benefits that might extend beyond the lifetime of a single supercomputer. For NERSC, OpenMP software investment was important for the continuity of a successful programming approach as well as portability to Perlmutter, other DOE supercomputers, and almost certainly the next NERSC supercomputer. Similar NRE projects must also consider how much focus to place on training and user support. In this project, it would have been desirable to provide individual consultation and hands-on support to all NERSC users. However, this is not scalable to the approximately 8000 users and 700 codes at NERSC. Instead, we focused on the needs of five NESAP-2 teams and included milestones with training and documentation requirements. NERSC and NVIDIA were able to create training materials for all NERSC users based on lessons learned from test suite and NESAP-2 applications. Other HPC Centers with fewer users and codes may see more value in funding more extensive individual support.

¹The fine-grained performance results are approximately 2.5x faster than published at [9]. This is for two reasons: use of A100 versus V100 GPUs, and commenting out `cudaDeviceSetCacheConfig(cudaFuncCachePreferShared)` in Kokkos to prevent the Kokkos CUDA backend taking away L1 cache capacity from the XGC kernels.

ACKNOWLEDGMENTS

The NRE activity was funded by Subcontract 7437777. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231. Many people contributed to the effort to add OpenMP target offload support to the NVIDIA HPC Compilers. This paper's authors from NVIDIA give heartfelt thanks to the following colleagues: Simone Atzeni, Evgeny Baskakov, Mathew Colgrove, Sebastien Deldon, Morris Hafner, Zahra Khatami, Mark LeAir, Brent Leback, Jie Li, Razvan Lupusoru, Pallavi Mathew, Scott McMillan, Doug Miles, Dave Parks, Adam Smith, Mahdi Soltan Mohammadi, Zhen Wang, Michael Wolfe, and Yu You. The authors from NERSC would like to thank Charlene Yang, Mauro Del Ben, Dhruva Kulkarni, Brandon Cook and Douglas Doerfler for contributing NESAP-2 application test cases and Brian Friesen for installing NVIDIA compilers on NERSC systems.

REFERENCES

- [1] Brian Austin et al. 2020. *NERSC-10 Workload Analysis (Data from 2018)*. NERSC. Retrieved August 20, 2021 from https://portal.nersc.gov/project/m888/nersc10/workload/N10_Workload_Analysis.latest.pdf
- [2] Jon Bashor. 2015. *NERSC, Cray Move Forward With Next-Generation Scientific Computing*. NERSC. Retrieved August 20, 2021 from <https://www.nersc.gov/news-publications/nersc-news/nersc-center-news/2015/nersc-cray-move-forward-with-next-generation-scientific-computing/>
- [3] David A Beckingsale, Jason Burmark, Rich Hornung, Holger Jones, William Kilian, Adam J Kunen, Olga Pearce, Peter Robinson, Brian S Ryujin, and Thomas RW Scogland. 2019. RAJA: Portable performance for large-scale scientific applications. In *2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)* (Denver, CO, USA). IEEE Computer Society, Los Alamitos, CA, USA, 71–81. <https://doi.org/10.1109/P3HPC49587.2019.00012>
- [4] Mauro Del Ben, Charlene Yang, and Christopher Daley. 2021. *BerkeleyGW-Kernels-Fortran*. LBNL and NERSC. Retrieved August 20, 2021 from <https://gitlab.com/NERSC/nersc-proxies/berkeleygw-kernels-fortran>
- [5] Mauro Del Ben, Charlene Yang, Zhenglu Li, Felipe H. da Jornada, Steven G. Louie, and Jack Deslippe. 2020. *Accelerating Large-Scale Excited-State GW Calculations on Leadership HPC Systems*. IEEE Computer Society, Los Alamitos, CA, USA, 1–11.
- [6] OpenMP Architecture Review Board. 2020. *OpenMP Application Programming Interface. Version 5.1 November 2020*. OpenMP Architecture Review Board. Retrieved August 20, 2021 from <https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-1.pdf>
- [7] Swen Boehm, Swaroop Pophale, Verónica G. Vergara Larrea, and Oscar Hernandez. 2018. Evaluating Performance Portability of Accelerator Programming Models using SPEC ACCEL 1.2 Benchmarks. In *High Performance Computing, Rio Yokota, Michèle Weiland, John Shalf, and Sadaf Alam* (Eds.). Springer International Publishing, Cham, 711–723.
- [8] Sunita Chandrasekaran and Guido Juckeland. 2017. *OpenACC for Programmers: Concepts and Strategies*. Addison-Wesley Professional, Boston, MA, USA. (Note: NERSC created an OpenMP offload version of Laplace based on the OpenACC implementation in the book: Retrieved August 20, 2021 from https://github.com/OpenACCUserGroup/openacc_concept_strategies_book/blob/master/Code_Examples/Chapter_04/laplace_final_acc.c).
- [9] Barbara Chapman, Buu Pham, Charlene Yang, Christopher Daley, Colleen Bertoni, Dhruva Kulkarni, Dossay Oryspayev, Ed D'Azevedo, Johannes Doerfert, Keren Zhou, Kiran Ravikumar, Mark Gordon, Mauro Del Ben, Meifeng Lin, Melisa Alkan, Michael Kruse, Oscar Hernandez, P.K. Yeung, Paul Lin, Peng Xu, Swaroop Pophale, Tosaporn Sattasathuchana, Vivek Kale, William Huhn, and Yun He. 2021. Outcomes of OpenMP Hackathon: OpenMP Application Experiences with the Offloading Model. In *OpenMP: Enabling Massive Node-Level Parallelism, IWOMP 2021*, Simon McIntosh-Smith, Bronis R. de Supinski, and Jannis Klinkenberg (Eds.). Springer International Publishing, Cham. Forthcoming. Accepted but not yet published.
- [10] Christopher Daley and Annemarie Southwell. 2020. Accelerating Applications for the NERSC Perlmutter Supercomputer Using OpenMP. GPU Technology Conference (GTC), March 25 2020. Retrieved August 22, 2021 from <https://developer.nvidia.com/gtc/2020/video/s21387-vid>
- [11] Christopher Daley and Güray Özen. 2021. Accelerating Applications for NERSC's Perlmutter Supercomputer using OpenMP and NVIDIA HPC SDK. GPU Technology Conference (GTC), April 13 2021. Retrieved August 22, 2021 from <https://www.nvidia.com/en-us/on-demand/session/gtcspring21-s31657>
- [12] Christopher Daley, Hadia Ahmed, Samuel Williams, and Nicholas Wright. 2020. A Case Study of Porting HPGMG from CUDA to OpenMP Target Offload. In *OpenMP: Portable Multi-Level Parallelism on Modern Systems*, Kent Milfeld, Bronis R. de Supinski, Lars Koesterke, and Jannis Klinkenberg (Eds.). Springer International Publishing, Cham, 37–51.
- [13] Christopher Daley and Rahul Gayatri. 2021. *NERSC/SC21-NERSC-NRE-paper: Artifacts for SC21 NERSC-NRE-paper titled "Non-Recurring Engineering (NRE) Best Practices: A Case Study with the NERSC/NVIDIA OpenMP Contract"*. NERSC. <https://doi.org/10.5281/zenodo.5168093>
- [14] Joshua Hoke Davis, Christopher Daley, Swaroop Pophale, Thomas Huber, Sunita Chandrasekaran, and Nicholas J. Wright. 2021. Performance Assessment of OpenMP Compilers Targeting NVIDIA V100 GPUs. In *Accelerator Programming Using Directives*, Sridutt Bhalachandra, Sandra Wienke, Sunita Chandrasekaran, and Guido Juckeland (Eds.). Springer International Publishing, Cham, 25–44.
- [15] Bronis R. de Supinski, Thomas R. W. Scogland, Alejandro Duran, Michael Klemm, Sergi Mateo Bellido, Stephen L. Olivier, Christian Terboven, and Timothy G. Mattson. 2018. The Ongoing Evolution of OpenMP. *Proc. IEEE* 106, 11 (2018), 2004–2019. <https://doi.org/10.1109/JPROC.2018.2853600>
- [16] Tom Deakin, James Price, Matt Martineau, and Simon McIntosh-Smith. 2021. *BabelStream*. University of Bristol High Performance Computing group. Retrieved August 20, 2021 from <https://github.com/UoB-HPC/BabelStream>
- [17] Jose Monsalve Diaz, Josh Davis, Thomas Huber, Sunita Chandrasekaran, Swaroop Pophale, Oscar Hernandez, and David Berndt. 2021. *SOLLVE V&V*. Department of Energy's Exascale Computing Project. Retrieved August 20, 2021 from https://github.com/SOLLVE/sollve_vv
- [18] Jose Monsalve Diaz, Swaroop Pophale, Oscar Hernandez, David E. Bernholdt, and Sunita Chandrasekaran. 2018. OpenMP 4.5 Validation and Verification Suite for Device Offload. In *Evolving OpenMP for Evolving Architectures*, Bronis R. de Supinski, Pedro Valero-Lara, Xavier Martorell, Sergi Mateo Bellido, and Jesus Labarta (Eds.). Springer International Publishing, Cham, 82–95.
- [19] Doug Doerfler. 2021. *su3_bench: Lattice QCD SU(3) matrix-matrix multiply microbenchmark*. NERSC. Retrieved August 20, 2021 from https://gitlab.com/NERSC/nersc-proxies/su3_bench
- [20] H. Carter Edwards, Christian R. Trott, and Daniel Sunderland. 2014. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *J. Parallel and Distrib. Comput.* 74, 12 (2014), 3202 – 3216. <https://doi.org/10.1016/j.jpdc.2014.07.003>
- [21] Rahul Kumar Gayatri. 2020. *BerkeleyGW-Kernels-CPP*. NERSC. Retrieved August 20, 2021 from <https://gitlab.com/NERSC/nersc-proxies/BerkeleyGW-Kernels-CPP> (Note: Includes GPP and FF BerkeleyGW mini-apps).
- [22] Rahul Kumar Gayatri, Stan Moore, Evan Weinberg, Nicholas Lubbers, Sarah Anderson, Jack Deslippe, Danny Perez, and Aidan P. Thompson. 2020. Rapid Exploration of Optimization Strategies on Advanced Architectures using Test-SNAP and LAMMPS. *CoRR* abs/2011.12875 (2020). [arXiv:2011.12875](https://arxiv.org/abs/2011.12875) [cs.DC]
- [23] Rahul Kumar Gayatri, Charlene Yang, Thorsten Kurth, and Jack Deslippe. 2019. A Case Study for Performance Portability Using OpenMP 4.5. In *Accelerator Programming Using Directives*, Sunita Chandrasekaran, Guido Juckeland, and Sandra Wienke (Eds.). Springer International Publishing, Cham, 75–95.
- [24] JaeHyuk Kwack and Buu Pham. 2021. *GAMMESS RI-MP2 mini-app*. Argonne National Laboratory. Retrieved August 20, 2021 from https://github.com/jkwack/GAMMESS_RI-MP2_MiniApp
- [25] Jeongnim Kim. 2021. *miniQMC - QMCPACK Miniapp*. Condensed Matter Physics, National Center for Supercomputing Applications, University of Illinois materials computation Center, University of Illinois. Retrieved August 20, 2021 from <https://github.com/QMCPACK/miniqmc> (Note: OpenMP offload version in the OMP_offload branch).
- [26] Josh Meyers. 2021. *batoid: A c++ backed python optical raytracer*. LSST Dark Energy Science Collaboration. Retrieved August 20, 2021 from <https://github.com/jmeyers314/batoid>
- [27] Kathy Kincade. 2021. NERSC, ALCF, Codeplay Partner on SYCL for Next-generation Supercomputers. Retrieved August 20, 2021 from <https://www.nersc.gov/news-publications/nersc-news/nersc-center-news/2021/nersc-alcf-codeplay-partner-on-sycl-for-next-generation-supercomputers/>
- [28] Tuomas Koskela and Charlene Yang. 2019. *ToyPush*. NERSC. Retrieved August 20, 2021 from <https://gitlab.com/NERSC/toypush> (Note: OpenMP offload version in the oacc-omp branch).
- [29] Seung-Hoe Ku, Choong-Seock Chang, and Patrick H. Diamond. 2009. Full-f gyrokinetic particle simulation of centrally heated global ITG turbulence from magnetic axis to edge pedestal top in a realistic tokamak geometry. *Nuclear Fusion* 49, 11 (sep 2009), 115021. <https://doi.org/10.1088/0029-5515/49/11/115021>
- [30] Thorsten Kurth, William Arndt, Taylor Barnes, Brandon Cook, Jack Deslippe, Doug Doerfler, Brian Friesen, Yun (Helen) He, Tuomas Koskela, Mathieu Lobet,

- Tareq Malas, Leonid Oliker, Andrey Ovsyannikov, Samuel Williams, Woo-Sun Yang, and Zhengji Zhao. 2017. Analyzing Performance of Selected NESAP Applications on the Cori HPC System. In *High Performance Computing*, Julian M. Kunkel, Rio Yokota, Michela Taufer, and John Shalf (Eds.). Springer International Publishing, Cham, 334–347.
- [31] LLVM Foundation. 2020. *Clang 11.0.0 Release Notes. OpenMP Support in Clang*. LLVM Foundation. Retrieved August 20, 2021 from <https://releases.llvm.org/11.0.0/tools/clang/docs/ReleaseNotes.html#openmp-support-in-clang>
- [32] LLVM Foundation. 2021. *The LLVM Compiler Infrastructure*. LLVM Foundation. Retrieved August 20, 2021 from <https://github.com/llvm/llvm-project>
- [33] Mark Harris. 2013. An Efficient Matrix Transpose in CUDA C/C++. Retrieved August 20, 2021 from <https://developer.nvidia.com/blog/efficient-matrix-transpose-cuda-cc> (Note: NERSC's OpenMP offload transpose application executed multiple matrix transpose kernels, including the tuned methods outlined in the NVIDIA blog post).
- [34] Matt Martineau and Simon McIntosh-Smith. 2021. *flow: A 2D hydrodynamics mini-app*. University of Bristol High Performance Computing group. Retrieved August 20, 2021 from <https://github.com/UoB-HPC/flow>
- [35] Matt Martineau and Simon McIntosh-Smith. 2021. *hot: A heat diffusion mini-app that uses a CG solver*. University of Bristol High Performance Computing group. Retrieved August 20, 2021 from <https://github.com/UoB-HPC/hot>
- [36] Matt Martineau and Simon McIntosh-Smith. 2021. *neutral: A Monte Carlo Neutron Transport Mini-App*. University of Bristol High Performance Computing group. Retrieved August 20, 2021 from <https://github.com/UoB-HPC/neutral>
- [37] John D. McCalpin. 2013. *Stream*. Department of Computer Science School of Engineering and Applied Science, University of Virginia, Charlottesville, Virginia. Retrieved August 20, 2021 from <http://www.cs.virginia.edu/stream/FTP/Code>
- [38] Andrew Myers, Axel Huebl, Marc Day, Weiqun Zhang, and Michael Zingale. 2021. *Tutorials for the AMReX framework*. AMReX code team. Retrieved August 20, 2021 from <https://github.com/AMReX-Codes/amrex-tutorials> (Note: OpenMP offload version of ElectromagneticPIC in directory Particles/ElectromagneticPIC/Exec/OpenMP).
- [39] Hai Ah Nam, Gabriel Rockefeller, Mike Glass, Shawn Dawson, John Levesque, and Victor Lee. 2017. The Trinity Center of Excellence Co-Design Best Practices. *Computing in Science & Engineering* 19, 5 (2017), 19–26. <https://doi.org/10.1109/MCSE.2017.3421542>
- [40] J. Robert Neely and Bronis R. de Supinski. 2017. Application Modernization at LLNL and the Sierra Center of Excellence. *Computing in Science Engineering* 19, 5 (2017), 9–18. <https://doi.org/10.1109/MCSE.2017.3421556>
- [41] NVIDIA Corporation. 2021. *CUDA Toolkit*. NVIDIA Corporation. Retrieved August 20, 2021 from <https://developer.nvidia.com/cuda-toolkit>
- [42] NVIDIA Corporation. 2021. *NVIDIA A100 Tensor Core GPU. Unprecedented scale at every scale*. NVIDIA Corporation. Retrieved August 20, 2021 from <https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/a100-80gb-datasheet-update-nvidia-us-1521051-r2-web.pdf>
- [43] NVIDIA Corporation. 2021. *Thrust*. NVIDIA Corporation. Retrieved August 20, 2021 from <https://github.com/NVIDIA/thrust>
- [44] NVIDIA Corporation and NERSC. 2020. *NVIDIA HPC SDK - OpenMP Target Offload Training*, December 2020. Retrieved August 20, 2021 from <https://www.nersc.gov/users/training/events/nvidia-hpcsdk-openmp-target-offload-training-december-2020>
- [45] OpenACC Organization. 2020. *The OpenACC Application Programming Interface. Version 3.1. November, 2020*. OpenACC Organization. Retrieved August 20, 2021 from <https://www.openacc.org/sites/default/files/inline-images/Specification/OpenACC-3.1-final.pdf>
- [46] Exascale Computing Project. 2017. *On the Path to the Nation's First Exascale Supercomputers: PathForward*. Retrieved August 20, 2021 from <https://www.exascaleproject.org/path-nations-first-exascale-supercomputers-pathforward/>
- [47] Rich Hornung. 2021. *RAJA Performance Suite*. Lawrence Livermore National Lab. Retrieved August 20, 2021 from <https://github.com/LLNL/RAJAPerf>
- [48] Andrew Siegel, Erik Draeger, Jack Deslippe, Anshu Dubey, Tom Evans, Tim Germann, and William Hart. 2020. *State and Progress of ECP Applications: Application Development*. Technical Report WBS 2.2, Milestone PM-AD-1080. Lawrence Livermore National Lab. (LLNL), Livermore, CA, USA. Retrieved August 20, 2021 from https://www.exascaleproject.org/wp-content/uploads/2020/03/ECP_AD_Milestone-Early-Application-Results_v1.0_20200325_FINAL.pdf
- [49] SOLLVE and NERSC. 2021. January 2021 ECP OpenMP Hackathon by SOLLVE and NERSC. Retrieved August 20, 2021 from <https://sites.google.com/view/ecpomphackjan2021>
- [50] SPEC High Performance Group. 2017. *SPEC ACCEL*. Standard Performance Evaluation Corporation. Retrieved August 20, 2021 from <https://www.spec.org/accel>
- [51] SPEC High Performance Group. 2021. *SPEC HPC 2021 Benchmark Suites*. Standard Performance Evaluation Corporation. Retrieved August 20, 2021 from <https://www.spec.org/hpc2021> (Note: An OpenMP offload version of HPGMG is available in the SPEC HPC 2021 benchmark suite).
- [52] Aidan P Thompson, Rahul Gayatri, Stan Moore, Steve Plimpton, and Christian Trott. 2019. *TestSNAP*. Sandia Corporation. Retrieved August 20, 2021 from <https://github.com/FitSNAP/TestSNAP> (Note: OpenMP offload version in the OpenMP4.5 branch).
- [53] Aidan P Thompson, Rahul Gayatri, Stan Moore, Steve Plimpton, and Christian Trott. 2021. *TestSNAP (Kokkos)*. Sandia Corporation. Retrieved August 20, 2021 from <https://github.com/rgayatri23/TestSNAP> (Note: Kokkos version in the Kokkos-nvhpc branch).
- [54] Aidan P Thompson, Laura P Swiler, Christian R Trott, Stephen M Foiles, and Garritt J Tucker. 2015. Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials. *J. Comput. Phys.* 285 (2015), 316–330.
- [55] U.S. Department of Energy. 2018. *DOE to Build Next-Generation Supercomputer at Lawrence Berkeley National Laboratory*. U.S. Department of Energy. Retrieved August 20, 2021 from <https://www.energy.gov/articles/doe-build-next-generation-supercomputer-lawrence-berkeley-national-laboratory>
- [56] Sudharshan S. Vazhkudai, Bronis R. de Supinski, Arthur S. Bland, Al Geist, James Sexton, Jim Kahle, Christopher J. Zimmer, Scott Atchley, Sarp Oral, Don E. Maxwell, Veronica G. Vergara Larrea, Adam Bertsch, Robin Goldstone, Wayne Joubert, Chris Chembreau, David Appelhans, Robert Blackmore, Ben Casses, George Chochia, Gene Davison, Matthew A. Ezell, Tom Gooding, Elsa Gonsiorowski, Leopold Grinberg, Bill Hanson, Bill Hartner, Ian Karlin, Matthew L. Leininger, Dustin Leverman, Chris Marroquin, Adam Moody, Martin Ohmacht, Ramesh Pankajakshan, Fernando Pizzano, James H. Rogers, Bryan Rosenberg, Drew Schmidt, Mallikarjun Shankar, Feiyi Wang, Py Watson, Bob Walkup, Lance D. Weems, and Junqi Yin. 2018. The Design, Deployment, and Evaluation of the CORAL Pre-Exascale Systems. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society, Los Alamitos, CA, USA, 661–672. <https://doi.org/10.1109/SC.2018.00055>
- [57] Verónica G. Vergara Larrea, Reuben D. Budiardja, Rahul Kumar Gayatri, Christopher Daley, Oscar Hernandez, and Wayne Joubert. 2020. Experiences in porting mini-applications to OpenACC and OpenMP on heterogeneous systems. *Concurrency and Computation: Practice and Experience* 32, 20 (2020), e5780. <https://doi.org/10.1002/cpe.5780>
- [58] Charlene Yang. 2020. 8 Steps to 3.7 TFLOP/s on NVIDIA V100 GPU: Roofline Analysis and Other Tricks. *CoRR abs/2008.11326* (2020). arXiv:2008.11326 [cs.DC] <https://arxiv.org/abs/2008.11326>

Appendix: Artifact Description/Artifact Evaluation

SUMMARY OF THE EXPERIMENTS REPORTED

We evaluated the functionality and performance of NVIDIA HPC SDK compilers on a NERSC development system with NVIDIA A100 GPUs. We used NVIDIA HPC SDK 21.3 and 21.7 compilers as well as CUDA 11.0.2 and 11.2.1. The development system, named Cori-DGX in this paper, is 2 nodes of NVIDIA DGX with nodes consisting of 2 AMD Rome CPUs and 8 NVIDIA A100 GPUs. The evaluation included multiple HPC codes, including BabelStream, BerkeleyGW-GPP-Fortran, TestSNAP (kokkos and OpenMP-4.5 versions), and XGC-Collision. The publicly available benchmarks are at <https://github.com/NERSC/SC21-NERSC-NRE-paper>.

Author-Created or Modified Artifacts:

Persistent ID: <https://doi.org/10.5281/zenodo.5168093>
Artifact name: The authors' GitHub artifact repository

BASELINE EXPERIMENTAL SETUP, AND MODIFICATIONS MADE FOR THE PAPER

Relevant hardware details: Cori-DGX: NVIDIA DGX system: nodes have 2 AMD Rome CPUs and 8 NVIDIA A100 GPUs.

Operating systems and versions: Cori-DGX: openSUSE Leap 15.0 running Linux kernel 4.12.14-lp150.12.82-default

Compilers and versions: NVIDIA HPC SDK 21.3 and 21.7, CUDA 11.0.2 and 11.2.1

Applications and versions: BabelStream, BerkeleyGW-GPP-Fortran, TestSNAP (kokkos and OpenMP-4.5 versions), XGC-Collision

Libraries and versions: Kokkos (0b61c81: develop branch on Aug 2 2021)

Key algorithms: Algorithm depends on the application: BabelStream - Data streaming, BerkeleyGW-GPP-Fortran - Generalized Plasmon Pole model, TestSNAP - N body molecular dynamics, XGC-Collision - Particle In Cell Collision kernel

URL to output from scripts that gathers execution environment information.

https://github.com/NERSC/SC21-NERSC-NRE-paper/blob/main/aster/hardware-configuration/dgx_info.log