Integrating ORNL's HPC and Neutron Facilities with a Performance-Portable CPU/GPU Ecosystem

Steven E. Hahn, Philip W. Fackler, William F. Godoy, Ketan Maheshwari, Zachary Morgan, Andrei T. Savici, Christina M. Hoffmann, Pedro Valero-Lara, Jeffrey S. Vetter, and Rafael Ferreira da Silva
Oak Ridge National Laboratory, Oak Ridge, TN, USA
{hahnse,facklerpw,godoywf,km0,morganzj,saviciat,choffmann,valerolarap,vetter,silvarf}@ornl.gov

***Abstract*—We explore the development of a performance-portable CPU/GPU ecosystem to integrate two of the US Department of Energy's (DOE's) largest scientific instruments, the Oak Ridge Leadership Computing facility and the Spallation Neutron Source (SNS), both of which are housed at Oak Ridge National Laboratory. We select a relevant data reduction workflow use-case to obtain the differential scattering cross-section from data collected by SNS's CORELLI and TOPAZ instruments. We compare the current CPU-only production implementation using the Garnet Python multiprocess package based on the Mantid C++ framework against our proposed CPU/GPU implementation that uses the LLVM-based, just-in-time Julia scientific language and the JACC.jl performance-portable package. Two proxy apps were developed: (i) an app for extracting relevant Mantid kernels (MDNorm) in C++ and (ii) the Julia MiniVATES.jl miniapp. We present performance results for NVIDIA A100 and AMD MI100 GPUs and AMD EPYC 7513 and 7662 CPUs. The results provide insights for future generations of data reduction software that can embrace performance portability for an integrated research infrastructure across DOE's experimental and computational facilities.**

***Index Terms*—Performance portability, experimental facilities, High-Performance Computing, Julia, LLVM**

I. Introduction

Oak Ridge National Laboratory (ORNL) hosts two of the US Department of Energy's (DOE's) largest scientific user facilities: the Oak Ridge Leadership Computing Facility (OLCF) and the Spallation Neutron Source (SNS). The OLCF houses two of the most powerful supercomputers in the world—the GPU-accelerated Frontier and Summit.[1] ORNL is also home to experimental neutron science facilities, including the SNS, the High-Flux Isotope Reactor (HFIR), and the next-generation Second Target Station (STS). The OLCF has been pushing the limits of leadership computing for scientific discovery over the last two decades, and, with the deployment of Frontier, recently ushered in the exascale era [1]. SNS, HFIR, and the upcoming STS provide cutting-edge experimental neutron scattering facilities to study the structure and properties of materials used across several industries and domains. ORNL's world-class expertise and rich history in high-performance computing (HPC) [2] and neutron science [3] have had significant impact on DOE's scientific mission.

To continue this tradition of excellence, computation and data management must scale up with every generation of instrument and data acquisition technique [4]. As such, the recent DOE Integrated Research Infrastructure (IRI) [5] program aims to provide a seamless integration of computational, experimental, and observational capabilities to advance DOE's science mission. IRI's requirements for performance, interoperability, and extensibility require rethinking existing algorithms and implementations for future applications.

In this paper, we describe incorporating HPC capabilities, specifically performance-portable CPU/GPU computing, into a science use-case that connects the SNS and OLCF facilities. We focus on a computationally expensive algorithm for calculating the neutron scattering cross-section of single-crystal materials [6]. The current CPU-only implementation is widely used in the data reduction of time-of-flight (TOF) measurements on the SNS's CORELLI [7], [8] and TOPAZ [9] instruments. Given the large volume of data, increasingly complex data processing techniques, and the relative stagnation of single-core CPU performance, portable HPC paradigms offer a significant opportunity to advance science through accelerated and efficient computation (e.g., near–real time data processing, raw data processing). To this end, we developed proxy applications [10] to capture relevant computational aspects and enable performance portability by using the Julia language [11] and the JACC.jl [12] package. In the narrative that follows, we will

- provide an understanding of the unique challenges to connecting ORNL experimental and computational facilities (Section II),
- outline a methodology for applying our proposed performance-portable ecosystem using proxy applications for existing SNS neutron science cases (Section III),
- quantify the potential performance improvements from the proposed ecosystem on CPU/GPUs from multiple vendors (Section IV), and
- provide future directions that can impact the science missions of ORNL facilities (Section VI).

This manuscript has been authored in part by UT-Battelle LLC under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The publisher, by accepting the article for publication, acknowledges that the US Government retains a non-exclusive, paid up, irrevocable, world-wide license to publish or reproduce the published form of the manuscript, or allow others to do so, for US Government purposes. The DOE will provide public access to these results per the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

[1]https://www.top500.org/lists/top500/2024/06/

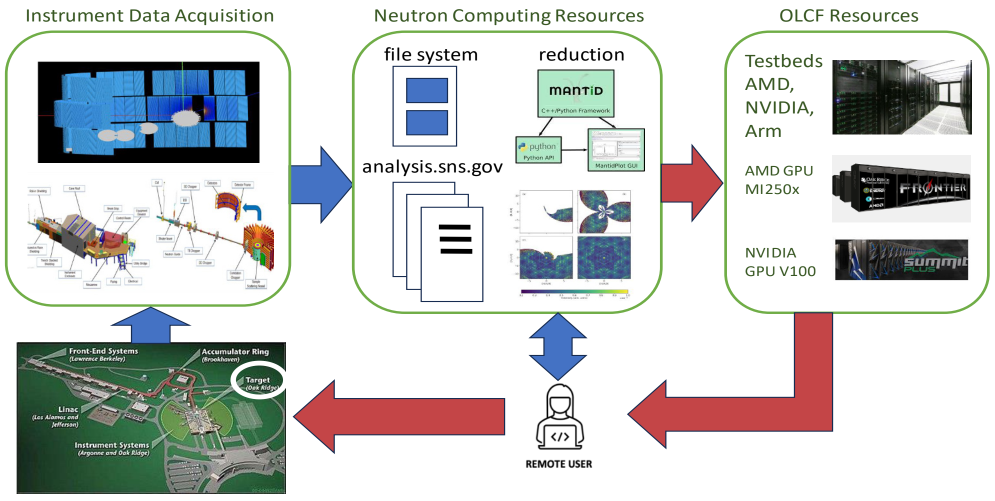
Instrument Data Acquisition
Neutron Computing Resources
OLCF Resources
file system
reduction
analysis.sns.gov
Testbeds AMD, NVIDIA, Arm
AMD GPU MI250x
NVIDIA GPU V100
Front-End Systems (Lawrence Berkeley)
Accumulator Ring (Brookhaven)
Target (Oak Ridge)
Linac (Los Alamos and Jefferson)
Instrument Systems (Argonne and Oak Ridge)
REMOTE USER

Fig. 1. Representation of current (blue) and potential (red) integrated facility workflows for performance-portable codes.

II. Background

As depicted in Fig. 1, the targeted workflow components consist of five stages: (i) experiment conducted at target instrument; (ii) instrument data acquisition; (iii) data processing/reduction on SNS compute systems; (iv) remote user access (portal, notebook, or ssh console); and (v) connection with OLCF's heterogeneous test beds.

Below, we provide descriptions of several components involved in connecting ORNL facilities as we work toward a performance-portable ecosystem.

Data generation at neutron facilities. In the past decade, ORNL neutron facilities have adopted event-based data collection techniques [13], which represent a great opportunity to obtain higher-fidelity recordings of experimental data, including detector pixel id, neutron TOF from source to detector, and proton pulse wall-clock time [14]. Raw event data at ORNL neutron instruments is collected at a rate of nearly 1.4 TB/day for a grand total of roughly 1.8 PB since its construction is 2006, with plans to expand as next-generation instruments become available. The raw, event-based data for each experiment run is stored by using the metadata-rich NeXus schema [15], which is built on top of the widely adopted HDF5 [16] file format. Each instrument stores a subset of the NeXus schema according to their characteristics. Data is hosted at ORNL computing facilities and available to users via remote access [17].

Single crystal CORELLI and TOPAZ instruments. SNS facilities at ORNL offer a suite of neutron single-crystal diffraction instruments [18]. TOPAZ is a high-resolution, single-crystal diffractometer, whereas CORELLI has energy discrimination to improve diffuse scattering measurements of single-crystal samples. These instruments enable the study of a wide range of materials with different molecular sizes. The selected science cases in our study represent the workloads used to process raw data on these beamlines.

Data reduction workflows. To transform the raw data into a meaningful, interpretable, and *reduced* form of histograms and plots that reveal material characteristics, an ecosystem of specialized software is required. The Mantid [19] framework began in 2007 as an international collaboration between multiple user facilities (ISIS, ILL, ORNL, CSNS, and more) to provide a common C++ software package for data reduction, visualization, and analysis of neutron and muon scattering experiments. As shown in Fig. 1, users remotely access and analyze their data on computing resources hosted at the SNS. Mantid's performance is determined by a combination of CPU-only compute, available memory, algorithm improvements, and file I/O [20]–[22]. Currently, Mantid primarily uses OpenMP [23] for multithreaded CPU parallelism and lacks multinode HPC capabilities. Moreover, next-generation data reduction software (e.g., Scipp [24], ScippNeutron [25]) do not target HPC applications. Recently, emphasis has shifted toward using Mantid as a back end driven by Python front end frameworks. The target for this work, Garnet, is one such interface that focuses on single-crystal diffraction instruments.

OLCF ACE Test Beds and ExCL. In preparation for IRI, the OLCF has deployed the Advanced Computing Ecosystem (ACE) test bed systems. ACE encompasses a diverse array of cutting-edge computing environments designed to propel scientific discovery and innovation. ACE offers a sandboxed environment designed to deploy and evaluate heterogeneous computing and data resources. The ACE test bed is structured around several key areas of interest that are pivotal to the evolution and enhancement of HPC systems as we target novel drivers in science (e.g., AI): (i) IRI workflow patterns; (ii) emerging architectures, networking, and storage; and (iii) HPC resources moving to the cloud. Future efforts will focus on further enhancing the test bed's capabilities, thereby expanding its range of applications. The Experimental Computing Laboratory (ExCL),[2] also hosted at ORNL, was used as part of our initial code validation for single-GPU architectures. ExCL provides a set of accelerator-ready systems[3] that enable us to experiment with vendor hardware and software stacks prior to launching jobs on OLCF systems. It also serves as point of reference for correctness and performance across similar GPU models (e.g., MI100 on OLCF's Defiant and ExCL's cousteau). A summary of the hardware used in our experiments is listed in Table I. We are particularly interested in running on NVIDIA and AMD GPUs, at least initially, with plans to extend this to future architectures.

TABLE I
SYSTEMS OVERVIEW

System	Characteristics
Defiant (OLCF)	Nodes: 36 (3 cabinets of 12) CPU: 64-core AMD EPYC 7662 Rome, 4 NUMA GPU: 4 AMD MI100 32 GB HBM2 Memory: 256 GB, DDR4
Milan0 (ExCL)	Nodes: 1 CPU: 2 × 32-Core AMD EPYC 7513, 2 NUMA GPU: 2 NVIDIA A100 80 GB Memory: 1 TB, DDR4-3200
bl12-analysis2 (SNS)	Nodes: 1 CPU: 16-core AMD EPYC 7343, 1 NUMA GPU: 1 NVIDIA T600 4 GB Memory: 512 GB, DDR4

Julia, LLVM, and JACC.jl. Powered by just-in-time (JIT) compilation via the widely adopted LLVM compiler toolchain [26], Julia offers a high-performance, dynamic, and science-friendly syntax and ecosystem for HPC [27]–[29]. JACC.jl is developed at ORNL as part of our portfolio of performance-portability capabilities for DOE. As shown in Fig. 2, the package unifies the existing Julia ecosystem that targets CPU and GPU vendors. It provides a simple memory `Array` and kernel `parallel_for` and `parallel_reduce` API for application development. The primary difference between JACC.jl and existing solutions, specifically KernelAbstractions.jl [30], is that JACC.jl targets more than just GPU performance and can adapt between coarse and fine granularity. This enables scientists to focus on their kernels rather than on lower-level computational aspects to achieve performance-portable code.

III. Methodology

Figure 3 shows an overview of our proposed methodology. To capture computational characteristics for exploring performance-portable solutions, the existing Garnet/Mantid workflow was used as a reference for developing proxy applications in C++ and Julia.

[2]https://docs.excl.ornl.gov/
[3]https://docs.excl.ornl.gov/system-overview

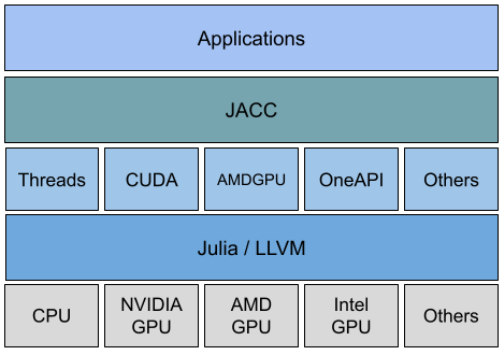

Fig. 2. JACC.jl performance portability architecture showing available LLVM and Julia CPU/GPU back ends.

A. Garnet/Mantid Workflow

The current workflow starts with each experiment's run stored in a NeXus/HDF5 file and tagged with a particular identifier. For simplicity, we use 1-to-N. Each raw dataset is reduced by using a generic `LoadEventNexus` that creates a memory structure called `MDEventWorkspace`. The Mantid MDNorm algorithm is then applied to each component of the dataset and can be broken down into two parts: (i) histogramming (BinMD) an array of individual events and (ii) calculating the associated normalization (MDNorm) on a grid. The differential scattering cross-section is therefore the sum of all histograms over the sum of all normalizations (Fig. 3). The overall process is shown in Fig. 4 using the Bixbyite case as an example [31]. First, we see a single run (one per file) in which a symmetry is applied. The same symmetry operation process is then applied to all runs in the experiment, resulting in the cross-section for the entire measurement. The overall cross-section calculation is represented in Algorithm 1.

Algorithm 1 Cross-Section Calculation using MDNorm and BinMD

```
$start, end \leftarrow range(MPI\_Rank, MPI\_Size)$
$0 \leftarrow mdnorm, binmd$
**for** $i = start$ to $end$ **do**
    $event\_data \leftarrow LOAD\, events, rotations, charge, ...$
    $mdnorm \mathrel{+}= MDNorm(events) \leftarrow CPU/GPU$
    $binmd \mathrel{+}= BinMD(events) \leftarrow CPU/GPU$
**end for**
$cross\_section \leftarrow \frac{f\_MPI\_Reduce(binmd)}{f\_MPI\_Reduce(mdnorm)}$
```

B. Proxy Applications

We developed proxy applications that calculate the cross-section with Algorithm 1 and used C++ to capture current Mantid CPU workloads and the Julia-based MiniVATES.jl package for adding performance portability with JACC.jl. We also made improvements over the monolithic closed-box nature of the current Garnet/Mantid workflow. Breaking the algorithm into smaller pieces can potentially future-proof the process for adaptable performance. These algorithmic

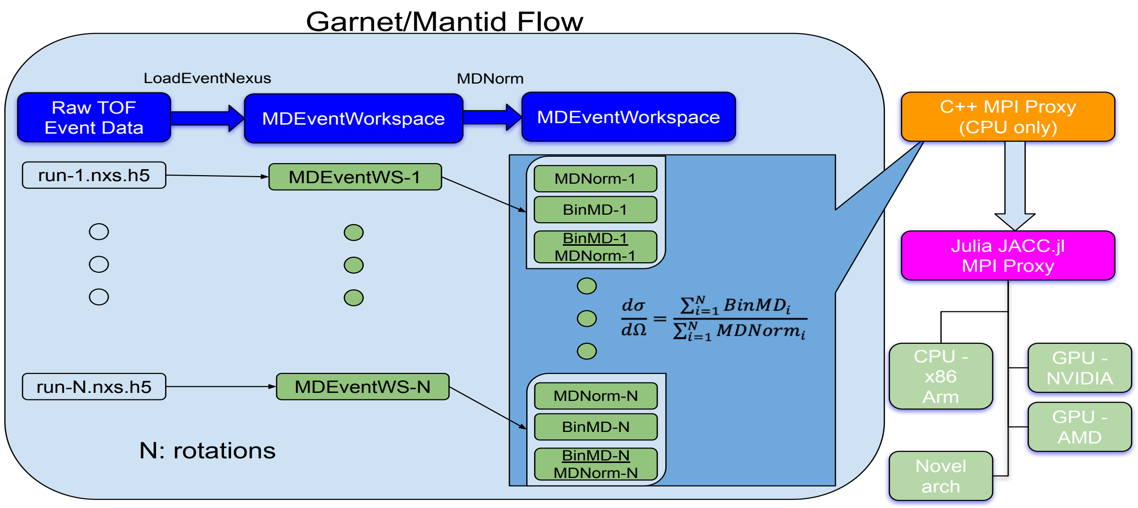

Fig. 3. Proposed proxy applications in C++ and Julia and JACC.jl used to evaluate CPU/GPU performance-portable implementations for the current Garnet reduction workflow that uses Mantid.

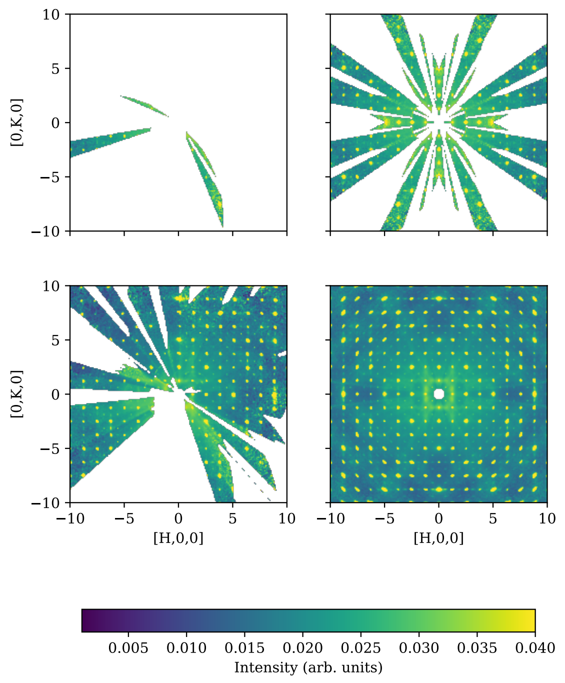

Fig. 4. Cross-section scattering data reduction ensemble measurement steps: single run (upper left), single run + symmetry (upper right), multiple (22) runs (lower left), and multiple runs + symmetry (lower right) [31].

enhancements include improving the complexity of linear searches with a more adaptable region-of-interest strategy. Also, we experimented with more HPC-oriented data structures: instead of sorting an array of structs, we sort an array of indices using primitive types. Benefits of this approach should increase with the number of dimensions. The Message Passing Interface (MPI) was used for the outermost loop over files. Each process allocates its own histogram, and the MPI reduce function is used to add histograms that contain the results of MDNorm and BinMD on a single process. These two histograms are then divided to calculate the scattering cross-section.

```
#pragma omp parallel for collapse(2) private(idx,
  momentum, intersections, xValues, yValues)
    // Symmetry transformations ~ 6 and 24
    for (const auto &op : transforms)
      // CORELLI 372K and TOPAZ 1.6M
      for (size_t i = 0; i < ndetectors; ++i)
        // calculate intersections ~(600x600x1)
          for (size_t i = 0; i < hBins; i++) ...
          for (size_t i = 0; i < kBins; i++) ...
          for (size_t i = 0; i < lBins; i++) ...
          // size for each operation
          // < hBins + kBins + lBins + 2
          sort(...)
          linear_interpolation(...)
          append_to_histogram(...)
```

Listing 1. **Parallelization of the MDNorm operation.**

The resulting C++ proxy application is outlined in Listings 1 and 2. The MDNorm code contains a loop over each symmetry operation and number of detectors, which get parallelized by using OpenMP's `collapse` clause. Each parallel worker runs 3 for-loops to calculate intersections in reciprocal space, a periodic lattice in (H, K, L) coordinates. An additional 3 linear operations sort, interpolate, and append the resulting normalization histogram needed for the denominator in the cross-section calculation. Hence, the complexity of MDNorm depends on the instrument and material characteristics.

To capture the simple computational complexities of Mantid's BinMD, we use a straightforward application of the algorithm applied to a single array of neutron events. Mantid's BinMD uses a more adaptive strategy by having a hierarchy of boxes with equal numbers of events. Hence, the single box algorithm in Listing 2 will capture the relevant 2 for-loops that apply the corresponding symmetry transformations to the existing number of events.

```
#pragma omp parallel for collapse(2)
    // Symmetry transformations ~ 6 and 24
    for (const auto &op : transforms)
      // CORELLI 40M, TOPAZ 280M
      for (auto &val : events)
        apply_transform(...)
```

Listing 2. **Parallelization of the BinMD operation.**

The resulting Julia proxy application, MiniVATES.jl, follows the structure of the C++ proxy app to perform the same set of computations. Listing 3 shows the `JACC.parallel_for` implementation for the BinMD algorithm that works on CPUs and GPUs from multiple vendors. Hybrid parallelism is achieved by using MPI.jl [32] and JACC.jl. JACC.jl maps parallel kernels and allocated memory to the appropriate back end—Threads, CUDA.jl [33], or AMDGPU.jl [34]—for different CPU/GPU systems.

For the MDNorm calculation in the Julia proxy app, best practice is to pre-allocate memory on the GPU. To avoid excessive allocation, an additional kernel (one for each file) is called before the main MDNorm kernel to ensure an accurate estimate of intersections can be computed. An elegant solution might use `JACC.parallel_reduce` with a MAX operator, but this function does not currently support custom reduction operators (it uses + internally). A workaround in MiniVATES.jl adds communication between device and host, and we hope this work will motivate future efforts in JACC and the Julia HPC stack.

```
function binEvents!(h::Hist3, events::AbstractArray,
    transforms::Array1{SquareMatrix3c})
  JACC.parallel_for(
    (length(transforms), size(events, 2)),
    (n, i, t) -> begin
      @inbounds begin
        op = t.transforms[n]
        v = op * C3[t.events[6, i], t.events[7, i],
            t.events[8, i]]
        atomic_push!(t.h, v[1], v[2], v[3],
        t.events[1, i])
      end
    end,
    (h = h, events, transforms),
  )
end
```

Listing 3. **MiniVATES.jl BinMD CPU/GPU implementation using JACC.jl.**

Another challenge in developing MiniVATES.jl is the need to sort the intersections for each detector within the body of the kernel. CUDA.jl sorting functions must be called from the CPU execution context because they launch their own kernel(s) internally. On the other hand, sorting algorithms in the Julia standard library (and other packages) all perform dynamic allocation internally for scratch space and are undesirable within a repeatedly called GPU kernel. A local implementation was used in this work. In pursuing an algorithm that does not allocate scratch space, we settled on *comb sort* after a bit of experimentation. Finally, MiniVATES.jl uses its own implementation of a 3D histogram based on Mantid's MDHistoworkspace. The bin values are thread-safe and incremented with atomic operations.

IV. Experimental Results

We establish a baseline by using two single-crystal materials that offer different computational characteristics: (i) Benzil [6], [35] measured on CORELLI and (ii) Bixbyte [31] measured on TOPAZ. Table II shows the relevant data and instrument characteristics that drive the computational costs in the MDNorm and BinMD algorithms.

TABLE II
Selected Use-Case Characteristics and WCTs on BL12-ANALYSIS2

Experiment	CORELLI Benzil	TOPAZ Bixbyte
Files	36	22
Symmetries	6	24
Events/file	40M	280M
Detectors	372K	1.6M
Bin counts	(603,603,1)	(601,601,1)
Bin bases	([H,H],[H,-H],[L])	([H],[K],[L])
Garnet/Mantid		
WCT **MDNorm + BinMD**	55 s	102 s
WCT Total	271 s	904 s

Current Garnet/Mantid wall-clock times (WCT) for the overall workflow execution and the MDNorm + BinMD calculation are also provided to illustrate user expectations on the SNS's shared production system. Bin counts are chosen for 2D slicing (`lBins=1`) to provide a balance between current memory, computation, and data movement costs. Speeding up these calculations enables broader modeling and simulation options (e.g., 3D volumes, real-time) and dynamically modifying histogram binning parameters while minimizing the need for data movement.

A. Benzil – CORELLI

Table III shows the execution WCTs for the proposed C++ and MiniVATES.jl proxy implementations on the Defiant system. We include JIT times due to the nature of the Julia language and because it is an amortized cost. As seen, WCTs for the MDNorm and BinMD proxies largely outperform the existing Garnet/Mantid implementation on SNS systems by ∼74× on CPU and ∼299× on GPU. Nevertheless, MiniVATES.jl total times show that precompilation offsets some of the benefits of running on Defiant's M100 GPUs. The UpdateEvents row measures the time spent loading an HDF5 array with 8 columns and a row for each neutron event.

On Milan0, the NVIDIA A100 provides a significant performance boost: MDNorm is over 3× faster and BinMD is over 172× faster than on the Defiant's AMD MI100. These results suggest that the NVIDIA A100 is atomically updating the histogram more efficiently than the AMD MI100. UpdateEvents is slightly faster with MiniVates.jl but the C++ proxy is 16× slower on Defiant. Both proxies use wrappers

TABLE III
WCT in Seconds for the Benzil (CORELLI) Proxies on Defiant's AMD EPYC 7662 64-core CPU and MI100 GPU

WCT (s)	C++ Proxy (CPU)	MiniVATES.jl (GPU) JIT	no JIT
UpdateEvents	0.092	0.136	0.064
MDNorm	0.688	4.669	0.174
BinMD	0.057	0.488	0.010
MDNorm + BinMD	0.746	5.157	0.184
Total	7.746	48.932	

TABLE IV
WCT in Seconds for the Benzil (CORELLI) Proxies on Milan0's AMD EPYC 7513 2 × 32-core CPU and NVIDIA A100 GPU

WCT (s)	C++ Proxy (CPU)	MiniVATES.jl (GPU) JIT	no JIT
UpdateEvents	1.250	0.090	0.0504
MDNorm	0.456	2.367	0.0532
BinMD	0.034	0.517	0.0000
MDNorm + BinMD	0.490	2.894	0.0532
Total	15.985	30.135	

over the C HDF5 API and transpose a 2D array from row-major to column-major. Substantial optimization opportunities might exist on certain network file systems.

B. Bixbyite – TOPAZ

Despite having fewer files, more detectors and neutron events make Bixbyite a much slower and more memory-intensive calculation. WCTs on a single node take minutes instead of seconds. To stay within memory limits, the C++ proxy was run with four MPI processes and 16 OpenMP threads per process. Between algorithmic improvements to MDNorm and larger file sizes, most time is spent loading events from disk. In MiniVATES.jl, the 6× slower loading of subsequent files is repeatable but not yet understood. Compared to the C++ proxy, MDNorm is over 6× faster and BinMD is almost 2× faster on the AMD MI100 GPUs.

TABLE V
WCT in Seconds for the Bixbyite (TOPAZ) Proxies on Defiant's AMD EPYC 7662 64-core CPU and MI100 GPU

WCT (s)	C++ Proxy (CPU)	MiniVATES.jl (GPU) JIT	no JIT
UpdateEvents	23.70	3.12	18.12
MDNorm	2.81	4.51	0.45
BinMD	5.40	3.70	2.95
MDNorm + BinMD	8.21	8.21	3.40
Total	215.98	553.89	

As shown in Table VI, MiniVATES.jl BinMD excels on Milan0's NVIDIA A100, with iterations after JIT compilation running over 50,000× faster than the C++ proxy on CPU and on the AMD MI100. MiniVATES.jl's MDNorm is 3× faster than the C++ proxy and approximately equal on the AMD MI100. MiniVATES.jl's UpdateEvents is 6× faster than on Defiant. The first file that includes JIT compilation is 0.7 seconds slower, as expected. In contrast, with the C++ proxy, UpdateEvents is almost 2× slower than on Defiant.

TABLE VI
WCT in Seconds for the Bixbyite (TOPAZ) Proxies on Milan0's AMD EPYC 7513 2 × 32-core CPU and NVIDIA A100 GPU

WCT(s)	C++ Proxy (CPU)	Julia Proxy (GPU) JIT	no JIT
UpdateEvents	42.59	3.784	3.037
MDNorm	1.53	3.133	0.518
BinMD	3.08	0.766	5.31E-5
MDNorm + BinMD	4.61	3.899	0.518
Total	306.46	67.02	

V. Related Work

Recent relevant work has focused on different aspects of connecting experimental and computational facilities. Veseli et al. [36], Prince et al. [37], and Parraga et al. [38] demonstrated improved streaming, computing, and workflow capabilities across the Advanced Photon Source and the Polaris supercomputer, both hosted at Argonne National Laboratory. Kommera et al. [39] focused on offloading x-ray diffraction workflows for image reconstruction to NVIDIA GPUs on the Summit and Perlmutter systems. Their research highlighted that performance portability is a desired future capability given the heterogeneous nature of DOE's HPC systems.

There are also relevant efforts in connecting ORNL neutron science and HPC facilities. Shipman et al. [40] introduced the Accelerating Data Acquisition, Reduction, and Analysis system, which provides live-streaming capabilities inside the Mantid framework for seamless interactions with ORNL's Compute and Data Environment for Science. This approach allows scientists with little or no HPC expertise to access HPC capabilities for real-time experiment analysis and steering if necessary. Watson et al. presented CALVERA [41], a platform for integrated services for a more effective neutron scattering data reduction and analysis. Their work manages resources at the workflow level, and one of their use cases involves integration with HPC and leveraging advances in the DCA++ code [42], [43] on the Summit supercomputer for studying strongly correlated quantum materials. In the INTERSECT initiative, Engelmann et al. [44], [45] provide an open federated architecture and microservices to enable connecting experimental and computational resources, including HPC, to target mainly automated and autonomous instrument science and data analysis.

VI. Conclusions and Future Work

We described performance-portable capabilities to connect and improve ongoing science at two of DOE's largest user facilities: experimental neutron science at the SNS and computational science at the OLCF. The selected science use-cases, the SNS's CORELLI Benzil and TOPAZ's Bixbyte, illustrate relevant computationally expensive workloads that can be adapted to existing HPC facilities for better performance capability.

We argue that the current practice of targeting multithreaded CPUs and hard-coded frameworks for a particular data hosting platform is not sustainable with the advent of next-generation data producers in a heterogeneous, AI-driven, post-Moore era.

In a connected facility setting, we demonstrate that proxy applications can be used to effectively investigate the factors (e.g., algorithms, memory movement, CPU/GPU parallelization) that drive performance beyond data movement. This approach helps scale future experimental workloads while acknowledging the differences from traditional HPC simulations. We also demonstrate the promise of Julia as a single performance-portable language powered by LLVM to improve developer productivity and user experience over the traditional decision to split code into a user-friendly scripting language and higher-performing compiled language. Addressing performance-portable aspects early on allows for an incremental co-design effort to balance computation and I/O improvements. Thus, we expect to contribute toward a more integrated research infrastructure as the scientific requirements (e.g., energy efficient, large data volume, AI workloads) evolve for future HPC and experimental facilities.

Acknowledgements

This research used resources of the OLCF and the ExCL at ORNL, which is supported by DOE's Office of Science under Contract No. DE-AC05-00OR22725. This research also used resources at the SNS, a DOE Office of Science User Facility operated by ORNL. This work is funded, in part, by Bluestone, an X-Stack project in the DOE Advanced Scientific Computing Office with program manager Hal Finkel.

References

[1] D. Kothe, S. Lee, and I. Qualters, "Exascale Computing in the United States," *Computing in Science & Engineering*, vol. 21, no. 1, pp. 17–29, 2019.

[2] S. Atchley, C. Zimmer, J. R. Lange, D. E. Bernholdt, V. G. Melesse Vergara, T. Beck, M. J. Brim, R. Budiardja, S. Chandrasekaran, M. Eisenbach, T. Evans, M. Ezell, N. Frontiere, A. Georgiadou, J. Glenski, P. Grete, S. Hamilton, J. Holmen, A. Huebl, D. Jacobson, W. Joubert, K. McMahon, E. Merzari, S. G. Moore, A. Myers, S. Nichols, S. Oral, T. Papatheodore, D. Perez, D. M. Rogers, E. Schneider, J.-L. Vay, and P. Yeung, "Frontier: Exploring exascale the system architecture of the first exascale supercomputer," in *SC23: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2023, pp. 1–16.

[3] J. Rumsey, "A history of neutron scattering at ORNL," *Neutron News*, vol. 29, no. 1, pp. 10–16, 2018. [Online]. Available: https://doi.org/10.1080/10448632.2018.1446588

[4] S. Campbell, M. Doucet, S. Hartman, J. Hetrick, J. Lin, Y. Liu, T. Naughton III, T. Proffen, S. Qian, and J. Taylor, "Second target station computer science and math workshop report," 9 2022. [Online]. Available: https://www.osti.gov/biblio/1909115

[5] W. L. Miller, D. Bard, A. Boehnlein, K. Fagnan, C. Guok, E. Lançon, S. Ramprakash, M. Shankar, N. Schwarz, and B. L. Brown, "Integrated Research Infrastructure Architecture Blueprint Activity (Final Report 2023)," 7 2023. [Online]. Available: https://www.osti.gov/biblio/1984466

[6] A. T. Savici, M. A. Gigg, O. Arnold, R. Tolchenov, R. E. Whitfield, S. E. Hahn, W. Zhou, and I. A. Zaliznyak, "Efficient data reduction for time-of-flight neutron scattering experiments on single crystals," *Journal of Applied Crystallography*, vol. 55, no. 6, pp. 1514–1527, Dec 2022. [Online]. Available: https://doi.org/10.1107/S1600576722009645

[7] S. Rosenkranz and R. Osborn, "Corelli: Efficient single crystal diffraction with elastic discrimination," *Pramana*, vol. 71, pp. 705–711, 2008.

[8] F. Ye, Y. Liu, R. Whitfield, R. Osborn, and S. Rosenkranz, "Implementation of cross correlation for energy discrimination on the time-of-flight spectrometer CORELLI," *Journal of Applied Crystallography*, vol. 51, no. 2, pp. 315–322, Apr 2018. [Online]. Available: https://doi.org/10.1107/S160057671800403X

[9] M. Frost, C. Hoffmann, J. Thomison, M. Overbay, M. Austin, P. Carman, R. Viola, E. Miller, and L. Mosier, "Initial testing of a Compact Crystal Positioning System for the TOPAZ Single-Crystal Diffractometer at the Spallation Neutron Source," *Journal of Physics: Conference Series*, vol. 251, no. 1, p. 012084, nov 2010. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/251/1/012084

[10] S. Matsuoka, J. Domke, M. Wahib, A. Drozd, A. A. Chien, R. Bair, J. S. Vetter, and J. Shalf, "Preparing for the Future—Rethinking Proxy Applications," *Computing in Science & Engineering*, vol. 24, no. 2, pp. 85–90, 2022.

[11] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, Jan. 2017.

[12] P. Valero-Lara, W. F. Godoy, H. Mankad, K. Teranishi, J. S. Vetter, J. Blaschke, and M. Schanen, "JACC: Leveraging HPC Meta-Programming and Performance Portability with the Just-in-Time and LLVM-based Julia Language," in *Proceedings of the SC '24 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, 2024.

[13] G. E. Granroth, K. An, H. L. Smith, P. Whitfield, J. C. Neuefeind, J. Lee, W. Zhou, V. N. Sedov, P. F. Peterson, A. Parizzi, H. Skorpenske, S. M. Hartman, A. Huq, and D. L. Abernathy, "Event-based processing of neutron scattering data at the Spallation Neutron Source," *Journal of Applied Crystallography*, vol. 51, no. 3, pp. 616–629, Jun 2018. [Online]. Available: https://doi.org/10.1107/S1600576718004727

[14] P. F. Peterson, S. I. Campbell, M. A. Reuter, R. J. Taylor, and J. Zikovsky, "Event-based processing of neutron scattering data," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 803, pp. 24 – 28, 2015.

[15] M. Könnecke, F. A. Akeroyd, H. J. Bernstein, A. S. Brewster, S. I. Campbell, B. Clausen, S. Cottrell, J. U. Hoffmann, P. R. Jemian, D. Männicke, R. Osborn, P. F. Peterson, T. Richter, J. Suzuki, B. Watts, E. Wintersberger, and J. Wuttke, "The NeXus data format," *Journal of Applied Crystallography*, vol. 48, no. 1, pp. 301–305, Feb 2015.

[16] The HDF Group. (1997-NNNN) Hierarchical Data Format, version 5. [Online]. Available: http://www.hdfgroup.org/HDF5/

[17] S. I. Campbell, S. D. Miller, J.-C. Bilheux, M. A. Reuter, P. F. Peterson, J. A. Kohl, J. R. Trater, S. S. Vazhkudai, V. E. Lynch, and M. L. Green, "The SNS/HFIR web portal system for SANS," *Journal of Physics: Conference Series*, vol. 247, p. 012013, oct 2010.

[18] L. Coates, H. Cao, B. C. Chakoumakos, M. D. Frontzek, C. Hoffmann, A. Y. Kovalevsky, Y. Liu, F. Meilleur, A. M. dos Santos, D. A. Myles *et al.*, "A suite-level review of the neutron single-crystal diffraction instruments at Oak Ridge National Laboratory," *Review of Scientific Instruments*, vol. 89, no. 9, 2018.

[19] O. Arnold, J. Bilheux, J. Borreguero, A. Buts, S. Campbell, L. Chapon, M. Doucet, N. Draper, R. Ferraz Leal, M. Gigg, V. Lynch, A. Markvardsen, D. Mikkelson, R. Mikkelson, R. Miller, K. Palmen, P. Parker, G. Passos, T. Perring, P. Peterson, S. Ren, M. Reuter, A. Savici, J. Taylor, R. Taylor, R. Tolchenov, W. Zhou, and J. Zikovsky, "Mantid—Data analysis and visualization package for neutron scattering and μ SR experiments," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 764, pp. 156–166, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900214008729

[20] W. F. Godoy, P. F. Peterson, S. E. Hahn, J. Hetrick, M. Doucet, and J. J. Billings, "Performance Improvements on SNS and HFIR Instrument Data Reduction Workflows Using Mantid," in *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI*, J. Nichols, B. Verastegui, A. B. Maccabe, O. Hernandez,

S. Parete-Koon, and T. Ahearn, Eds. Cham: Springer International Publishing, 2020, pp. 175–186.

[21] W. F. Godoy, P. F. Peterson, S. E. Hahn, and J. J. Billings, "Efficient Data Management in Neutron Scattering Data Reduction Workflows at ORNL," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 2674–2680.

[22] W. F. Godoy, A. T. Savici, S. E. Hahn, and P. F. Peterson, "Efficient loading of reduced data ensembles produced at ORNL SNS/HFIR neutron time-of-flight facilities," in *2021 IEEE International Conference on Big Data (Big Data)*, 2021, pp. 2949–2955.

[23] OpenMP Architecture Review Board, "OpenMP application program interface version 5.2," November 2021. [Online]. Available: https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-2.pdf

[24] S. Heybrock, O. Arnold, I. Gudich, D. Nixon, and N. Vaytet, "Scipp: Scientific data handling with labeled multi-dimensional arrays for c++ and python," *Journal of Neutron Research*, vol. 22, no. 2-3, pp. 169–181, 2020.

[25] S. Heybrock, J.-L. Wynen, O. Arnold, N. Vaytet, T. Willemsen, M. D. Jones, S. Jones, jokasimr, and M. Seth, "scipp/scippneutron: v24.07.0," Jul. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.12683198

[26] C. Lattner and V. Adve, "LLVM: A compilation framework for lifelong program analysis & transformation," in *International Symposium on Code Generation and Optimization, 2004. CGO 2004.* IEEE, 2004, pp. 75–86.

[27] V. Churavy, W. F. Godoy, C. Bauer, H. Ranocha, M. Schlottke-Lakemper, L. Räss, J. Blaschke, M. Giordano, E. Schnetter, S. Omlin, J. S. Vetter, and A. Edelman, "Bridging hpc communities through the julia programming language," 2022.

[28] W. F. Godoy, P. Valero-Lara, T. E. Dettling, C. Trefftz, I. Jorquera, T. Sheehy, R. G. Miller, M. Gonzalez-Tallada, J. S. Vetter, and V. Churavy, "Evaluating performance and portability of high-level programming models: Julia, python/numba, and kokkos on exascale nodes," in *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2023, pp. 373–382.

[29] W. F. Godoy, P. Valero-Lara, C. Anderson, K. W. Lee, A. Gainaru, R. Ferreira Da Silva, and J. S. Vetter, "Julia as a unifying end-to-end workflow language on the frontier exascale system," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1989–1999. [Online]. Available: https://doi.org/10.1145/3624062.3624278

[30] V. Churavy, D. Aluthge, L. C. Wilcox, J. Schloss, S. Byrne, M. Waruszewski, J. Samaroo, A. Ramadhan, Meredith, S. Schaub, J. Bolewski, A. Smirnov, C. Kawczynski, C. Hill, J. Liu, O. Schulz, Oscar, P. Haraldsson, T. Arakaki, and T. Besard, "JuliaGPU/KernelAbstractions.jl: v0.8.3," Jun. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6742177

[31] N. Roth, F. Ye, A. F. May, B. C. Chakoumakos, and B. B. Iversen, "Magnetic correlations and structure in bixbyite across the spin-glass transition," *Phys. Rev. B*, vol. 100, p. 144404, Oct 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.100.144404

[32] S. Byrne, L. C. Wilcox, and V. Churavy, "MPI.jl: Julia bindings for the Message Passing Interface," *Proceedings of the JuliaCon Conferences*, vol. 1, no. 1, p. 68, 2021. [Online]. Available: https://doi.org/10.21105/jcon.00068

[33] T. Besard, C. Foket, and B. De Sutter, "Effective extensible programming: Unleashing Julia on GPUs," *IEEE Transactions on Parallel and Distributed Systems*, 2018.

[34] J. Samaroo, V. Churavy, W. Phillips, A. Ramadhan, J. Barmparesos, J. TagBot, L. Räss, M. Schanen, T. Besard, A. Smirnov, T. Arakaki, S. Antholzer, Alessandro, C. Elrod, M. Raayai, and T. Hu, "JuliaGPU/AMDGPU.jl: v0.4.1," Aug. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6949520

[35] R. Welberry and R. Whitfield, "Single crystal diffuse neutron scattering," *Quantum Beam Science*, vol. 2, no. 1, 2018. [Online]. Available: https://www.mdpi.com/2412-382X/2/1/2

[36] S. Veseli, J. Hammonds, S. Henke, H. Parraga, and N. Schwarz, "Streaming data from experimental facilities to supercomputers for real-time data processing," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 2110–2117. [Online]. Available: https://doi.org/10.1145/3624062.3624610

[37] M. Prince, D. Gürsoy, D. Sheyfer, R. Chard, B. Côté, H. Parraga, B. Frosik, J. Tischler, and N. Schwarz, "Demonstrating cross-facility data processing at scale with laue microdiffraction," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 2133–2139. [Online]. Available: https://doi.org/10.1145/3624062.3624613

[38] H. Parraga, J. Hammonds, S. Henke, S. Veseli, W. Allcock, B. Côté, R. Chard, S. Narayanan, and N. Schwarz, "Empowering scientific discovery through computing at the advanced photon source," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 2126–2132. [Online]. Available: https://doi.org/10.1145/3624062.3624612

[39] P. R. Kommera, V. B. Ramakrishnaiah, and C. M. Sweeney, "Accelerate M-TIP on GPUs and deploy to Summit and NERSC-9 (against simulated data) WBS 2.2.4.05 ExaFEL, Milestone ADSE13-199," 7 2020. [Online]. Available: https://www.osti.gov/biblio/1830563

[40] G. Shipman, S. Campbell, D. Dillow, M. Doucet, J. Kohl, G. Granroth, R. Miller, D. Stansberry, T. Proffen, and R. Taylor, "Accelerating data acquisition, reduction, and analysis at the spallation neutron source," in *2014 IEEE 10th International Conference on e-Science*, vol. 1, 2014, pp. 223–230.

[41] G. R. Watson, G. Cage, J. Fortney, G. E. Granroth, H. Hughes, T. Maier, M. McDonnell, A. Ramirez-Cuesta, R. Smith, S. Yakubov, and W. Zhou, "Calvera: A platform for the interpretation and analysis of neutron scattering data," in *Accelerating Science and Engineering Discoveries Through Integrated Research Infrastructure for Experiment, Big Data, Modeling and Simulation*, K. Doug, G. Al, S. Pophale, H. Liu, and S. Parete-Koon, Eds. Cham: Springer Nature Switzerland, 2022, pp. 137–154.

[42] U. R. Hähner, G. Alvarez, T. A. Maier, R. Solcà, P. Staar, M. S. Summers, and T. C. Schulthess, "DCA++: A software framework to solve correlated electron problems with modern quantum cluster methods," *Computer Physics Communications*, vol. 246, p. 106709, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465519300086

[43] G. Balduzzi, A. Chatterjee, Y. W. Li, P. W. Doak, U. Haehner, E. F. D'Azevedo, T. A. Maier, and T. Schulthess, "Accelerating dca++ (dynamical cluster approximation) scientific application on the summit supercomputer," in *2019 28th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2019, pp. 433–444.

[44] C. Engelmann, O. Kuchar, S. Boehm, M. J. Brim, T. Naughton, S. Somnath, S. Atchley, J. Lange, B. Mintz, and E. Arenholz, "The INTERSECT Open Federated Architecture for the Laboratory of the Future," in *Accelerating Science and Engineering Discoveries Through Integrated Research Infrastructure for Experiment, Big Data, Modeling and Simulation*, K. Doug, G. Al, S. Pophale, H. Liu, and S. Parete-Koon, Eds. Cham: Springer Nature Switzerland, 2022, pp. 173–190.

[45] C. Engelmann and S. Somnath, "Science use case design patterns for autonomous experiments," in *Proceedings of the 28th European Conference on Pattern Languages of Programs*, ser. EuroPLoP '23. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3628034.3628060

Appendix

Artifact Description

All the codes in this study are hosted on GitHub.

- Mantid: https://github.com/mantidproject/mantid
- Garnet: https://github.com/neutrons/garnet_reduction
- JACC.jl: https://github.com/JuliaORNL/JACC.jl

Proxy Applications produced for this study are also on GitHub.

- C++ Proxy: https://github.com/quantumsteve/extract_mdnorm
- Julia Proxy: https://github.com/JuliaORNL/MiniVATES.jl

Appendix: Artifact Description/Artifact Evaluation

Artifact Description (AD)

I. Overview of Contributions and Artifacts

A. Paper's Main Contributions

We explore the benefits of performance-portable CPU/GPU computing using an innovative neutron diffraction data processing workflow that calculates the differential scattering cross-section from data collected at the Spallation Neutron Source (SNS). We compare the current CPU-only production implementation against our proposed CPU/GPU implementation that uses the Julia scientific language and the JACC.jl performance-portable package. Performance results are presented for NVIDIA A100 and AMD MI100 GPUs and for AMD EPYC 7513 and 7662 CPUs.

The paper contributions are:

- C_1 Provide a baseline wall-clock time of the CPU-only production implementation utilizing data collected by SNS's CORELLI and TOPAZ instruments.
- C_2 Develop and evaluate proxy applications for calculating the differential neutron scattering cross-section on current HPC CPU/GPU hardware.

B. Computational Artifacts

- A_1 https://github.com/neutrons/garnet_reduction
- A_2 https://github.com/quantumsteve/extract_mdnorm
- A_3 https://github.com/JuliaORNL/MiniVATES.jl

Artifact ID	Contributions Supported	Related Paper Elements
A_1	C_1	Table II
A_2	C_2	Tables III-VI
A_3	C_2	Tables III-VI

II. Artifact Identification

A. Computational Artifact A_1

Relation To Contributions

Garnet reduction is the current state of the practice for processing raw neutron scattering data for single crystal diffraction. Measured wall-clock times were used as a baseline for the proxy applications. It was also used to prepare intermediate data files that became inputs for the proxy applications.

Expected Results

The HDF5 output file from Garnet is the reduced and normalized data scientists would use for further analysis. It can be loaded and viewed in Mantid. The Bixbyite output should match Fig. 4 (lower right) and the Benzil output should match Fig 8b of reference 7.

Expected Reproduction Time (in Minutes)

Execution times on the order of minutes are presented in Table II.

Artifact Setup (incl. Inputs)

Hardware: Garnet is typically run on shared memory workstations with at least 256 GB of memory and depends on network access to the ORNL Neutron Catalog (ONCAT) and the /SNS and /HFIR remote data mounts. It will utilize multiple CPU cores, but will not utilize GPUs or distributed computing.

Software: All the codes in this study are hosted on GitHub.
Garnet: https://github.com/neutrons/garnet_reduction
Git commit ec821be2 was used in this work.

Datasets / Inputs: Data files are stored on a remote filesystem mounted on analysis.sns.gov and instrument computers.

Installation and Deployment: The conda environment was created an activated with

```
conda env create -f environment.yml
conda activate garnet_reduction
```

Artifact Execution

The CORELLI and TOPAZ reduction files were modified to match the parameters used in the proxies and are available in the extract_mdnorm repository

The normalization workflow can be run with pytest

```
pytest -s tests/test_normalization.py \
    -k test_corelli
pytest -s tests/test_normalization.py \
    -k test_topaz
```

Artifact Analysis (incl. Outputs)

The console output contains the combined wall-clock times from MDNorm and BinMD for each neutron scattering measurement as well as total execution time for each workflow.

B. Computational Artifact A_2

Relation To Contributions

The C++ proxy extracts the minimal relevant code from Mantid to reproduce MDNorm and BinMD calculations for single crystal diffraction. This intermediate step was used to explore potential algorithmic improvements and translate into Julia.

Expected Results

This proxy application should reproduce the Garnet reduction output, but be easier to build and run on HPC hardware outside the SNS and HFIR facilities. It should be faster than Garnet on similar hardware.

Expected Reproduction Time (in Minutes)

Execution times on the order of minutes are presented in Tables III-VI.

Artifact Setup (incl. Inputs)

Hardware: The C++ proxy can be run on a single node or on distributed HPC systems. It will utilize multiple CPU cores and nodes, but will not utilize GPUs.

Software: The C++ Proxy (Git commit 143641e4 was used in this work) uses the CMake build system to find MPI, Boost, Eigen3 and HDF5. HighFive and Catch2 are built using CMake FetchContent. On milan0, the GCC 13.2.0 module was loaded and Ubuntu 22.04 packages are used for OpenMPI, Boost, Eigen3 and HDF5.

On Defiant, cce/15.0.0, cray-mpich/8.1.23, and cray-hdf5/1.12.2.1 modules were loaded and spack used to build CMake, Boost 1.74 and Eigen3.

Datasets / Inputs: The input data to Mantid's MDNorm algorithm is saved into two HDF5 files per run. Python code such as this example available in the extract_mdnorm repository is inserted before Garnet reduction or a similar workflow calls MDNorm. The SaveMD function saves the MDEventWorkspace containing individual neutron events. Any additional data not in this file is stored in a second HDF5 file.

Lastly, the VanadiumFile and FluxFile are copied to the same directory. Benzil data totals 8.5GB, which Bixbyite data totals 206GB.

Installation and Deployment: milan0

```
mkdir build
cmake ..
make
```

Defiant

```
mkdir build
CC=craycc CXX=crayCXX cmake ..
make
```

Artifact Execution

On milan0, the proxy was executed from the terminal.

```
mkdir build
cmake ..
make
time mpirun --map-by numa:PE=8 -np 8 \
    --bind-to core ./benzil_corelli
time mpirun --map-by numa:PE=16 -np 4 \
    --bind-to core ./bixbyite_topaz
```

On Defiant, jobs were submitted to a Slurm queue and executed with srun.

```
time srun -n 8 -c 8 --cpu-bind=cores \
    ./benzil_corell
time srun -n 4 -c 16 --cpu-bind=cores \
    ./bixbyite_topaz
```

Artifact Analysis (incl. Outputs)

The output text file from each run can be loaded plotted with tools such as numpy and matplotlib. Bixbyite output should match Fig. 4 (lower right). Benzil output should match Fig 8b of reference 7

C. Computational Artifact A_3

Relation To Contributions

The MiniVATES.jl proxy extracts the minimal relevant code from Mantid to reproduce MDNorm and BinMD calculations for single crystal diffraction. This translation of the C++ proxy into Julia utilizes the JACC.jl performance portability framework to utilize GPUs.

Expected Results

This proxy application should reproduce the Garnet output, be easier to build and run on HPC hardware outside the SNS and HFIR facilities and utilize AMD and NVIDIA GPUs.

Expected Reproduction Time (in Minutes)

Execution times on the order of seconds (benzill) and minutes (bixbyite) are presented in Tables III-VI.

Artifact Setup (incl. Inputs)

Hardware: The MiniVATES proxy can be run on a single node. It utilizes the JACC.jl peformance portability framework to run on multiple CPU cores as well as AMD and NVIDIA GPUs.

Software: MiniVATES.jl v0.0.9 was built with Julia 1.10.4. A configuration script for Defiant is available in the Mini-Vates.jl repository

This script loads PrgEnv-cray-amd, cray-mpich and julia modules, sets MPIPreferences to use cray-mpich and JACC to use the "amdgpu" backend.

On milan0, MiniVATES was also built with Julia 1.10.4 and the same Project.toml file. JACC was set to use the "cuda" backend.

Datasets / Inputs: The input data to Mantid's MDNorm algorithm is saved into two HDF5 files per run. Python code such availble in the extract_mdnorm repository is inserted before Garnet or similar workflow calls MDNorm. The SaveMD function saves the MDEventWorkspace containing individual neutron events. Any additional data not in this file is stored in a second HDF5 file.

Lastly, the VanadiumFile and FluxFile are copied to the same directory. Benzil data totals 8.5GB, which Bixbyite data totals 206GB.

Installation and Deployment: Julia is a just-in-time compiled language.

Artifact Execution

On milan0, the proxy was executed from the terminal.

```
mpirun -np 1 julia \
    --project test/benzil_corelli.jl
mpirun -np 1 julia \
    --project test/bixbyite_topaz.jl
```

On Defiant, jobs were submitted to a Slurm queue and executed with srun.

```
srun -n 1 -c 1 --gpus-per-task=1 julia \
    --project test/benzil_corelli.jl
srun -n 1 -c 1 --gpus-per-task=1 julia \
    --project test/bixbyite_topaz.jl
```

Artifact Analysis (incl. Outputs)

MiniVATES.jl does not save any output files. Wall-clock times are printed to the terminal.