Evaluating Performance Portability of GPU Programming Models

Joshua H. Davis, Pranav Sivaraman, Isaac Minn, Abhinav Bhatele

[†]Department of Computer Science, University of Maryland
College Park, USA
{jhdavis,psivaram,iminn}@umd.edu,bhatele@cs.umd.edu

ABSTRACT

Maintaining a single codebase that can achieve good performance on a range of accelerator-based supercomputing platforms is of extremely high value for productive scientific application development. However, the large quantity of programming models available which claim to provide performance portability leaves developers with a complex choice when picking a model to use, potentially requiring an intensive effort to test each available model with kernels from their app. In order to better understand the current state of performance portable programming models, this project evaluates seven of the most popular programming models using two memory-bound mini-applications on two leadership-class supercomputers, Summit and Perlmutter. These results provide a useful evaluation of how well each programming model provides true performance portability in real-world usage for memory-bound applications.

KEYWORDS

performance portability, heterogeneous systems, programming models

1 SUMMARY

Heterogenous architectures in supercomputing platforms have become the dominant paradigm in high-performance computing. Eight of the top ten systems in the June 2023 TOP500 list employ co-processors or accelerators [16], with substantial diversity of hardware vendors and architectures in use. To provide a portable interface for programming such systems, a range of programming models such as OpenMP, Kokkos, and SYCL and others have emerged, all aiming to allow developers to maintain just one version of an application that can run on any desired supercomputing platform. However, the extent to which these portable programming models provide *performance portability*, a stronger property demanding not just functional portability but good performance across all targets systems, remains unclear.

From the perspective of a developer working with an application with no GPU support or GPU support through CUDA[1], choosing one of the numerous available popular programming models is an intimidating task. The choice is a major commitment in terms of time invested in both developer training and programming. If a programming model turns out to be a poor fit for an application, achieving unacceptable performance, then that investment is wasted. A team might decide to extract key kernels from their app, port them to the various candidate models, and test the performance of this resulting "mini-app". Yet this remains a substantial development effort, and still requires learning multiple models and writing multiple versions of the mini-app. Therefore, a deeper understanding of how well each available programming model currently enables performance portability would be a tremendous value to developers.

In order to provide this guidance, in this poster we empirically evaluate the performance portability enabled by seven of the most popular programming models for GPUs using two memory-bound mini-applications, BabelStream and XSBench, and evaluating on the OLCF Summit and NERSC Perlmutter supercomputers. The programming models represented in this poster are OpenMP target offloading, OpenACC, Kokkos, RAJA, SYCL, HIP, and CUDA. In the poster, we additionally evaluate our results with the well-known $\mathbf{\Phi}$ metric proposed by Pennycook, Sewall, and Lee [11–13, 15]. We conclude with general dicussion on the implications of the results for developers looking to chose a programming model for their memory-bound application and porting from Summit to Perlmutter.

1.1 Methodology

In this section we discuss the choice of mini-app codes, evaluation platforms, and software versions used.

1.1.1 Mini-app codes. To identify candidate codes for further study, we extensively surveyed the range of available proxy applications, mini-apps, and benchmarks for scientific computing on GPUs at the onset of this study. We sought out in particular mini-apps that represented a wide range of application categories, and codes that already had multiple implementations in different programming models of interest.

BabelStream is a memory bandwidth benchmark developed by the University of Bristol High Performance Computing group, previously evaluated by Deakin et al. and others [1–7, 9, 14]. This work is, to our knowledge, the first to compartively evalaute the performance of BabelStream from the production Summit system to the new Perlmutter system, using the hardware and software stack currently available to users. The BabelStream benchmark includes five kernels, of which we focus on three: copy, triad, and dot. Dot contains a reduction operation. We execute BabelStream with the command line argument `-n 800`.

XSBench is a memory-bound mini-app representing the continuous energy macroscopic neutron cross section lookup kernel from OpenMC (Monte Carlo), a neutron transport code [17]. XSBench has also been used as a performance portability case study before [8, 10], but to our knowledge no other work has evaluated XSBench comparitively with all of the seven programming models we evaluate on the in-production Summit and Perlmutter resources. XSBench contains one GPU kernel, relatively much larger than the BabelStream kernels, which processes independent cross-section lookups (binary searches) in parallel. As a part of this project effort,

[1]The non-portable proprietary language extension provided by NVIDIA.

we created a new Kokkos port of the XSBench code for performance portability evaluation, and are working to create a RAJA port as well. We execute XSBench with the command line arguments `-m event -s large`, to obtain an event-based simulation with the large problem size of 355 isotopes and 11,303 grid points.

1.1.2 Evaluation platforms and software versions. We evaluate portability of BabelStream and XSBench between the Summit system and Perlmutter system at Oak Ridge and Lawrence Berkeley National Labs, respectively. Summit is an IBM AC922 system employing 22-core IBM POWER9 CPUs and 16GB NVIDIA V100 GPUs. Perlmutter is an HPE Cray EX235n system with 64-core AMD EPYC 7763 CPUs and 40GB NVIDIA A100 GPUs. All tests are performed using a single GPU from the corresponding system.

On the Summit platform, all tests were conducted with the 11.5.2 CUDA module, employing LLVM verion 15 (2022-07-25 build) for OpenMP offloading and NVHPC 22.11 for OpenACC. On the Perlmutter platform, all tests were conducted with the 11.7 CUDA module, with the exception of OpenMP offloading which used CUDA 11.4 for compatibility with the system's LLVM 16 module. OpenACC was compiled with NVHPC version 22.7. On both platforms, HIP 5.4.3, Kokkos 4.1.00, SYCL DPC++ compiler version `sycl-nightly/20230621`, and RAJA version 2023.06.0 were used. In general, compiler and dependency versions were kept as similar in verion number as possible across systems while still relying on the available installations for CUDA and compilers on both systems as users typically would.

1.2 Results

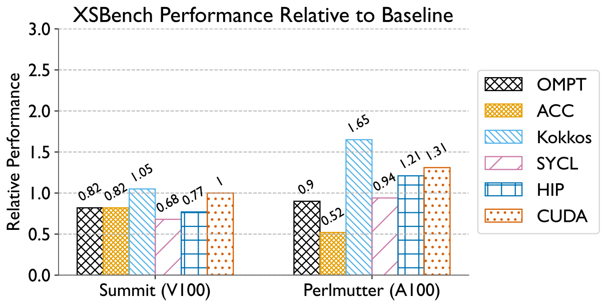

Figure 1: Performance comparison of portable programming models in the XSBench case on Summit and Perlmutter

Figure 1 shows relative performance of XSBench in six different programming models on a Summit V100 and Perlmutter A100. We use the CUDA Summit port as a performance baseline, as it uses the older architecture with the most low-level and vendor-specific programming model. The strong performance of Kokkos relative to the CUDA baseline is an immediate surprising insight from Fig. 1, given that Kokkos is a high-level abstraction. On Summit, other programming models achieve roughly similar performance, with SYCL achieving the worst performance and OpenMP and OpenACC the best compared to CUDA besides Kokkos. On Perlmutter, results vary much more significantly. All models get a performance benefit from moving to Perlmutter, as expected, except OpenACC, which worsens substantially.

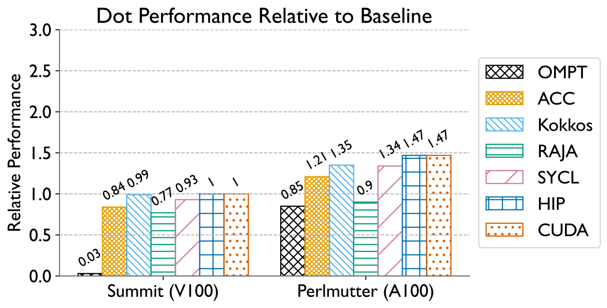

Figure 2: Performance comparison of portable programming models in the BabelStream dot kernel case on Summit and Perlmutter

Fig. 2 shows relative performance of the dot kernel from BabelStream in seven different programming models on Summit and Perlmutter. Again, the CUDA Summit version is the performance baseline. The dot kernel shows generally less variability across models when compared to XSBench, which may be due to its relative simplicity. OpenMP offload on Summit is an extreme low outlier, also achieving the worst performance on Perlmutter. The poor performance of some OpenMP compilers for reduction operations has been previously discussed [2]. OpenACC and RAJA also fall somewhat short, while the other programming models generally achieve parity with CUDA. Further results for BabelStream kernels are shown in the poster, as well as analysis with the Φ metric of all results.

1.3 Conclusion

In summary, we find that Kokkos, CUDA, and HIP achieve the best performance portability, and OpenMP and OpenACC the worst, for these two memory-bound mini-apps on the NVIDIA GPU-based Summit and Perlmutter systems. The results shown here can set expectations for developers looking to select a programming model for their memory-bound application, as well as for those looking to port a memory-bound application from Summit V100s to Perlmutter A100s. These results are only for systems with NVIDIA GPUs—porting to the upcoming Frontier and Aurora systems, with their AMD and Intel GPUs, will certainly prove to be a greater challenge for performance portability, underscoring the need for additional guidance on selecting a programming model beyond what we show here. Future work will include significantly expanding the breadth of application categories studied in terms of performance portability, adding new implementations in popular programming models to obtain full coverage, and carrying out more detailed performance analysis to give more specific recommendations to developers on how to select a programming model given the characteristics of their particular application.

REFERENCES

[1] Hartwig Anzt, Yuhsiang M. Tsai, Ahmad Abdelfattah, Terry Cojean, and Jack Dongarra. 2020. Evaluating the Performance of NVIDIA's A100 Ampere GPU for Sparse and Batched Computations. In *2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. 26–38. https://doi.org/10.1109/PMBS51919.2020.00009

[2] 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.

[3] Tom Deakin, James Cownie, Wei-Chen Lin, and Simon McIntosh-Smith. 2022. Heterogeneous Programming for the Homogeneous Majority. In *2022 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*. 1–13. https://doi.org/10.1109/P3HPC56579.2022.00006

[4] Tom Deakin and Simon McIntosh-Smith. 2020. Evaluating the Performance of HPC-Style SYCL Applications. In *Proceedings of the International Workshop on OpenCL* (Munich, Germany) *(IWOCL '20)*. Association for Computing Machinery, New York, NY, USA, Article 12, 11 pages. https://doi.org/10.1145/3388333.3388643

[5] Tom Deakin, Simon McIntosh-Smith, James Price, Andrei Poenaru, Patrick Atkinson, Codrin Popa, and Justin Salmon. 2019. Performance Portability across Diverse Computer Architectures. In *2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*. 1–13. https://doi.org/10.1109/P3HPC49587.2019.00006

[6] Tom Deakin, Andrei Poenaru, Tom Lin, and Simon McIntosh-Smith. 2020. Tracking Performance Portability on the Yellow Brick Road to Exascale. In *2020 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*. 1–13. https://doi.org/10.1109/P3HPC51967.2020.00006

[7] Tom Deakin, James Price, Matt Martineau, and Simon McIntosh-Smith. 2018. Evaluating Attainable Memory Bandwidth of Parallel Programming Models via BabelStream. *Int. J. Comput. Sci. Eng.* 17, 3 (jan 2018), 247–262.

[8] Johannes Doerfert, Marc Jasper, Joseph Huber, Khaled Abdelaal, Giorgis Georgakoudis, Thomas Scogland, and Konstantinos Parasyris. 2023. Breaking the Vendor Lock: Performance Portable Programming through OpenMP as Target Independent Runtime Layer. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques* (Chicago, Illinois) *(PACT '22)*. Association for Computing Machinery, New York, NY, USA, 494–504. https://doi.org/10.1145/3559009.3569687

[9] Jeff R. Hammond, Tom Deakin, James Cownie, and Simon McIntosh-Smith. 2022. Benchmarking Fortran DO CONCURRENT on CPUs and GPUs Using BabelStream. In *2022 IEEE/ACM International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. 82–99. https://doi.org/10.1109/PMBS56514.2022.00013

[10] JaeHyuk Kwack, John Tramm, Colleen Bertoni, Yasaman Ghadar, Brian Homerding, Esteban Rangel, Christopher Knight, and Scott Parker. 2021. Evaluation of Performance Portability of Applications and Mini-Apps across AMD, Intel and NVIDIA GPUs. In *2021 International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*. 45–56. https://doi.org/10.1109/P3HPC54578.2021.00008

[11] S. John Pennycook and Jason D. Sewall. 2021. Revisiting a Metric for Performance Portability. In *2021 International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*. 1–9. https://doi.org/10.1109/P3HPC54578.2021.00004

[12] Simon J Pennycook, Jason D Sewall, and Victor W Lee. 2016. A metric for performance portability. In *Proceedings of the 7th International Workshop in Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*. https://arxiv.org/abs/1611.07409

[13] Simon J Pennycook, Jason D Sewall, and Victor W Lee. 2019. Implications of a metric for performance portability. *Future Generation Computer Systems* 92 (2019), 947–958.

[14] Goutham Kalikrishna Reddy Kuncham, Rahul Vaidya, and Mahesh Barve. 2021. Performance Study of GPU applications using SYCL and CUDA on Tesla V100 GPU. In *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. 1–7. https://doi.org/10.1109/HPEC49654.2021.9622813

[15] Jason Sewall, S. John Pennycook, Douglas Jacobsen, Tom Deakin, and Simon McIntosh-Smith. 2020. Interpreting and Visualizing Performance Portability Metrics. In *2020 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*. 14–24. https://doi.org/10.1109/P3HPC51967.2020.00007

[16] TOP500.org. 2023. June 2023 TOP500. https://www.top500.org/lists/top500/2023/06/

[17] John R Tramm, Andrew R Siegel, Tanzima Islam, and Martin Schulz. 2014. XSBench-the development and verification of a performance abstraction for Monte Carlo reactor analysis. *The Role of Reactor Physics toward a Sustainable Future (PHYSOR)* (2014).