Unlocking performance portability on LUMI-G supercomputer: A virtual screening case study

Gianmarco Accordi
Davide Gadioli
Gianluca Palermo
gianmarco.accordi@polimi.it
davide.gadioli@polimi.it
gianluca.palermo@polimi.it
Politecnico di Milano - DEIB
Milano, Italy

Luigi Crisci
Lorenzo Carpentieri
Biagio Cosenza
lcrisci@unisa.it
lcarpentieri@unisa.it
bcosenza@unisa.it
Università degli studi di Salerno
Fisciano, Italy

Andrea R. Beccari
andrea.beccari@dompe.com
Dompé farmaceutici SpA
Napoli, Italy

ABSTRACT

High-Performance Computing is the target system for virtual screening applications, which aim to suggest which candidates to test in the drug discovery process. The HPC heterogeneity of modern systems raises the functional and performance portability challenge. LiGen is a well-known virtual screening application that can offload the most demanding computation on GPUs. It has been used to perform extreme-scale virtual screening campaigns on HPC systems equipped with NVIDIA cards using a CUDA implementation. This paper reports the experience of running its SYCL implementation on the LUMI-G HPC system that leverages AMD GPUs. Based on the experimental results, the LiGen SYCL implementation performs well on AMD GPUs, enabling LiGen to run a virtual screening campaign on LUMI-G HPC infrastructure.

CCS CONCEPTS

• **Software and its engineering** → **Software performance**; • **General and reference** → **Performance**; • **Applied computing** → Bioinformatics.

KEYWORDS

AMD, High Performance Computing, HPC, Virtual Screening, SYCL, LUMI

ACM Reference Format:
Gianmarco Accordi, Davide Gadioli, Gianluca Palermo, Luigi Crisci, Lorenzo Carpentieri, Biagio Cosenza, and Andrea R. Beccari. 2024. Unlocking performance portability on LUMI-G supercomputer: A virtual screening case study. In *International Workshop on OpenCL and SYCL (IWOCL '24), April 8–11, 2024, Chicago, IL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3648115.3648125

1 INTRODUCTION

The drug discovery process aims at finding a molecule that has strong interactions with at least one target protein. From domain knowledge, we expect this interaction to yield a therapeutic effect during clinical trials. Recent studies demonstrate how the introduction of a *virtual screening* stage that suggests which are the molecules to consider in the *in-vitro* experiments improve the success probability [2, 14]. In particular, a virtual screening application estimates the interaction strength between the target proteins and each drug candidate. Then, domain experts use these values to rank all the candidates and select the most promising ones to forward to the next stages of drug discovery. By simulating known chemical reactions, it is possible to create new virtual molecules. For this reason, the number of candidates we can virtually screen depends solely on the available computational power.

In the context of urgent computing, where it is important to find a solution in a short time frame, High-Performance Computing is the ideal system to carry out computation [10]. For example, the largest virtual screening campaign has been carried out in response to the recent pandemic using two HPC machines (Marconi100 at CINECA and HPC5 at ENI) [7]. Both of them use NVIDIA V100 cards to accelerate the computation. However, if we look at the latest update of the TOP500 list[1], as of November 2023, the top four supercomputers computational power is due to processors/accelerators of different vendors. One implication is that functional portability and performance portability are of paramount importance for virtual screening applications that target urgent computing scenarios.

In this paper, we focus on LiGen, the virtual screening application of the EXSCALATE drug discovery platform [7]. It has been designed from scratch targeting HPC [11], with an efficient CUDA implementation [23]. In the LIGATE EuroHPC project[2] we ported the computation kernels to SYCL 2020 [5]. This implementation has been tested in HPC machines that use NVIDIA cards. This paper aims to measure the performance portability of the SYCL implementation when we target the LUMI-G supercomputer at CSC, which uses AMD Instinct MI250X to accelerate the computation. From experimental results, we can notice how SYCL enables the usage of LiGen on LUMI-G, solving the functional portability challenge. For the performance portability challenge, we compare the LiGen throughput on LUMI-G with different compilers. Moreover, to provide a context for the performance, we consider in the comparison also the LiGen performance on an NVIDIA A100 card since both cards were launched in the same period.

IWOCL '24, April 8–11, 2024, Chicago, IL, USA

ACM ISBN 979-8-4007-1790-1/24/04
https://doi.org/10.1145/3648115.3648125

[1]Website: https://www.top500.org/
[2]Website: https://www.ligateproject.eu/

The paper is structured as follows: Section 2 provides the background required to understand how LiGen maps the domain problem in the computation kernels, while Section 3 provides an overview of the state of the art. Section 4 describes in more detail the target application and the technical challenges of running LiGen on the LUMI-G supercomputer, while Section 5 reports the application performance. Finally, Section 6 concludes the paper.

2 BACKGROUND

The virtual screening problem has two inputs: a list of drug candidates, named *ligand*s, and at least one target protein. The evaluation of each ligand-protein pair is independent from the others, making the problem embarrassingly parallel. Moreover, for simplicity, we can focus on the case where we have only one protein without losing generality. Therefore, virtual screening aims to associate each ligand with a numerical value representing the interaction strength with the target protein. This is a complex task that requires to solve two well-known problems: molecular *docking* and *scoring* [15, 16].

Molecular docking aims at estimating the 3D displacement of the ligand atoms, named *pose* when interacting with the target protein. A ligand is a molecule with a small molecular weight, typically with less than one hundred atoms. A protein, instead, usually has tens of thousands of atoms. For this reason, the ligand might interact with the protein in several ways. Moreover, both of them are not rigid bodies. Protein-ligand interactions take into account all the degrees of freedom. A subset of the molecule bonds, named *rotamer*s, partitions the molecule atoms into two disjoint sets. Changing the molecule's geometry is possible by rotating each atom set along the bond axis without altering its chemical and physical properties. A ligand might have tens of rotamers.

Molecular scoring considers geometrical, chemical, and physical properties to estimate the interaction strength between a ligand pose and the target protein. Also, the evaluation of each pose-protein pair is independent of the others. Moreover, molecular docking usually provides multiple output poses. The ligand's score is equal to the score of the best pose.

3 RELATED WORKS

Interest in porting HPC applications to SYCL can be seen in the number of recent publications on the subject. The effort has been on the performance portability of large HPC codebase on exascale system [4, 6]. The Top500 list shows how HPC centers leverage heterogeneous systems to reach such computational scale. Benchmarking these systems and applications, in terms of performance portability and productivity, has been challenging, and different attempts are reported in literature [8, 9]. The widely adopted performance portability definition comes from Pennycook's [17, 18]. While there are different metrics used for productivity evaluation of porting [19]. The focus of our work is on the concept of performance portability, of which SYCL portings have yielded some interesting results [13, 20]. Examples of HPC applications that can benefit from a SYCL porting are molecular docking applications [3]. As mentioned, our study case is LiGen, which is undergoing an analysis of its own performance portability on different platforms [5, 22]. The EuroHPC infrastructure [3] has LUMI as its flagship supercomputer. The deployment of LUMI-G has brought its own challenges for HPC applications to support it. [12, 21].

[3]Website: https://eurohpc-ju.europa.eu/supercomputers/our-supercomputers_en

4 TARGET APPLICATION

LiGen is a C++ application that executes the same computation pipeline on all the nodes involved in the computation [7]. The idea is that each pipeline will compute its partition of the input data. Since the problem is embarrassingly parallel, the only synchronization point between the pipeline instances is when LiGen needs to write the results on the file system. The stage that docks and scores the input ligands has the highest computation demand, which we can offload to accelerators. LiGen uses a gradient descent with multiple restarts to dock a ligand, where each restart will dock a different initial pose. The gradient uses only geometrical information to drive the pose optimization. Then, a scoring function evaluates the interaction strength of the docked poses, taking into account also chemical and physical properties.

From the kernel's implementation point of view, we use a batched approach, where we dock and score more ligands in parallel on the device [23]. In the SYCL implementation, each work-group docks and scores a ligand, processing the poses serially. To benefit from group synchronizations and algorithms, we set the work-group size equal to the sub-group size of the device. Therefore, the number of ligands per batch defines the number of work-items in an ND-range. Moreover, to improve the computation efficiency, we execute ligands out-of-order clustering in the same batch ligands that we expect to have a similar execution time. To reach this goal, we use features of the inputs, such as the number of atoms and rotamers [1].

With this approach, the throughput depends on the number of ligands that we can process in parallel on the GPU, which is constrained by the available hardware resources. In the LiGen case study, the kernels are bound by the registers that store the properties of the ligand atoms, e.g., their 3D coordinates. For this reason, we use non-type template parameters to represent the maximum number of atoms. In this way, each cluster of ligands has a different instance of the kernel tailored for their maximum number of atoms, which might use a different number of registers.

The batch size for each ligand is computed at runtime based on hardware characteristics of the target GPU as shown in previous work [1, 23]. In particular, we set the number of ligands of each batch as a multiple of l as defined in Eq. 1,

$$l = \frac{wgs}{sgs} \times CU \tag{1}$$

CU is the number of available compute units, sgs is the device sub-group size, and wgs is the maximum number of active work-items in a work-group we can run concurrently in a single CU for the specific kernel. The idea is that l is the maximum number of ligands that can be active on the device. If we select a number of ligands lower than this threshold, we are going to underuse the GPUs. As reported in previous work, the best strategy is to select a multiple of this number according to the available memory on the device.

While CU and sgs depend only on the device, we also need to consider the most demanding kernel to compute wgs. In particular, wgs is the `kernel_device_specific::work_group_size` property defined in the SYCL 2020 standard. Unfortunately, not

all SYCL compilers provide a meaningful value, requiring us to manually tune the l parameter for each cluster of ligands.

5 EXPERIMENTAL EVALUATION

This section reports the LiGen performance obtained on the LUMI-G supercomputer using the SYCL implementation. At first, we focused on a single GPU, using two different SYCL compilers. To provide context, we include the LiGen performance on other GPUs and implementations in the comparison. Finally, we consider all the GPUs available in a LUMI-G compute node.

5.1 Experimental Setup

This section describes the software stacks and the hardware characteristics we used to perform the experiments. We used the average throughput in terms of ligands/sec to measure the performance, including all the communication overheads (GPUs and file system).

5.1.1 Software stack. LUMI-G does not officially support SYCL. We build GCC 11.3 from upstream to get the updated STD Library using the GCC provided by the LUMI/22.08 software stack. Then, we used GCC 11.3 to compile LLVM 15.0.6. With these versions of GCC and LLVM, we have built AdaptiveCpp 0.9.4 from the corresponding Git repository and oneAPI DPC++ Compiler 2022-12 from the Intel LLVM Git Project, which was the latest stable release available when we had access to LUMI. We used NVCC 11.7 to compile the CUDA implementation. Moreover, we used the HIPIFY tool[4] to automatically generate a HIP implementation from the CUDA one, based on HIP 5.3. We have used the ROCm LLVM's to perform a code build of the HIP version on AMD GPUs.

5.1.2 Hardware characteristics. Mainly the data have been collected on the LUMI-G partition. Each LUMI-G node comprises one AMD Epyc 7A53 CPU, 512 GB of RAM, and four AMD Instinct MI250X. We have also included the data collected on Nvidia A100 and AMD MI100 GPUs for performance portability analysis. The MI100 data have been collected on a machine equipped with an AMD EPYC 7313 16-Core Processor, 512 GB of RAM, and an AMD Instinct MI100. The A100 data were collected from a machine equipped with one AMD EPYC 7282 16-core processor, 64 GB of RAM, and an Nvidia A100.

5.2 Single GPU performance portability

This section measures the LiGen performance on a single physical GPU for assessing performance portability. To provide context, we also report the application performance on an NVIDIA A100 and an AMD MI100 GPU, which are HPC-graded GPUs launched in a similar period. On all the GPUs, we report the throughput with both SYCL compilers. Moreover, we include an automatically generated HIP version for AMD GPUs, while for NVIDIA GPUs, we include a hand-optimized CUDA version.

Figure 1 reports the LiGen average throughput. The color and pattern of each bar identify the kernel implementation. If we focus only on the MI250X GPU card, we can notice how AdaptiveCPP produces a significantly faster LiGen version than oneAPI. This gap needs further investigation. We can also notice how the automatically generated HIP version yields slightly lower throughput than

[4]Repository: https://github.com/ROCm/HIPIFY

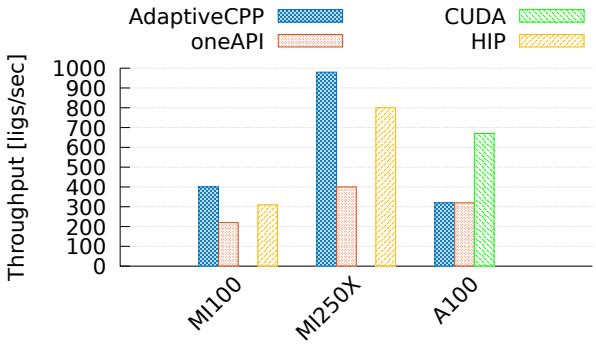

Figure 1: Throughput of LiGen on different GPUs using different compilers.

AdaptiveCPP. This is a remarkable result, considering the difference in effort behind the SYCL and HIP implementation. The trend on the three LiGen versions is consistent with the AMD MI100. This is expected since a single MI250X is packed with two chips, each with a number of compute units similar to the one on an MI100 (110 vs 120). If we compare the performance between the AdaptiveCpp implementation on MI250X and the CUDA implementation on A100, we can notice how the former is 1.46× faster than the latter. This implies we can leverage SYCL to use the GPU efficiently, enabling LiGen to run a virtual screening campaign on LUMI-G. However, if we compare the peak FLOPs declared by NVIDIA for the A100 and by AMD for the MI250X, we expect a more significant performance improvement, hinting that there is room for improvement.

5.3 Multi-GPUs performance

Each computation node on the LUMI-G system is composed of four AMD GPUs. While the computation kernel uses only a single GPU to compute a batch of ligands, LiGen can distribute batches on multiple GPUs in parallel using a work-stealing policy. This experiment aims at assessing the application performance while varying the number of used GPUs. The experiment aims to demonstrate that the multi-GPU LiGen version continues to scale linearly by varying the number of LUMI nodes and GPUs, as in the previous virtual screening campaign [7].

Table 1 reports the throughput of LiGen when running on 16 nodes, using one and all the available GPUs. The reported throughput is for each single node. From the experimental result, we notice that the SYCL implementation can use them efficiently, enabling multi-GPU performance portability.

Table 1: Throughput of LiGen upon using 16 LUMI-G nodes with a varying number of GPUs per node.

GPUs per Node	Throughput(lig/s)	Speed-up
1	488	1
8	3785	7.75

6 CONCLUSION

Virtual screening is an early stage of the drug discovery process that aims at suggesting which molecules to test in vitro. In the urgent computing context, a virtual screening application can hinge on the computation power of HPC systems to provide solutions in a short timeframe. However, the heterogeneity of the HPC landscape presents challenges in terms of functional and performance portability.

In this paper, we focus on the SYCL implementation of the LiGen virtual screening application to overcome them when we target the LUMI-G HPC system at CSC, which heavily relies on AMD GPU cards. On the bright side, the SYCL implementation can use all the available GPUs of a computation node, reaching an average throughput, in terms of ligands per second, higher than the CUDA implementation on an NVIDIA A100 card. These results enable the usage of LiGen on the LUMI-G supercomputer for a large-scale virtual screening campaign. However, when we compare the peak FLOPS declared by NVIDIA for the A100 and AMD for the MI250X, we can notice how the LiGen performance is below expectations. The reason for this behavior is still an open question that we are going to investigate. Moreover, we are surprised by the performance we were able to reach with the HIP implementation if we consider the effort required by porting the computation kernels in SYCL and the technical difficulties of building a SYCL toolchain from scratch using the Cray compilation environment.

ACKNOWLEDGMENTS

This project has received funding from EuroHPC-JU - the European High-Performance Computing Joint Undertaking - under grant agreement No 956137 (LIGATE). The JU receives support from the European Union's Horizon 2020 research and innovation program and Italy, Sweden, Austria, the Czech Republic, and Switzerland. We acknowledge EuroHPC Joint Undertaking for awarding us access to LUMI-G at CSC, Finland (EHPC-BEN-2022B12-001).

REFERENCES

[1] Gianmarco Accordi, Davide Gadioli, Emanele Vitali, Luigi Crisci, Biagio Cosenza, Andrea Beccari, and Gianluca Palermo. 2024. Out of kernel tuning and optimizations for portable large-scale docking experiments on GPUs. *The Journal of Supercomputing* (Feb. 2024). https://doi.org/10.1007/s11227-023-05884-y

[2] Marcello Allegretti, Maria Candida Cesta, Mara Zippoli, Andrea Beccari, Carmine Talarico, Flavio Mantelli, Enrico M Bucci, Laura Scorzolini, and Emanuele Nicastri. 2022. Repurposing the estrogen receptor modulator raloxifene to treat SARS-CoV-2 infection. *Cell Death & Differentiation* 29, 1 (2022), 156–166.

[3] Leonard Apanasevich, Yogesh Kale, Himanshu Sharma, and Ana Marija Sokovic. 2023. A Comparison of the Performance of the Molecular Dynamics Simulation Package GROMACS Implemented in the SYCL and CUDA Programming Models. (2023).

[4] Reuben D. Budiardja, Mark Berrill, Markus Eisenbach, Gustav R. Jansen, Wayne Joubert, Stephen Nichols, David M. Rogers, Arnold Tharrington, and O. E. Bronson Messer. 2023. Ready for the Frontier: Preparing Applications for the World's First Exascale System. In *High Performance Computing*, Abhinav Bhatele, Jeff Hammond, Marc Baboulin, and Carola Kruse (Eds.). Springer Nature Switzerland, Cham, 182–201.

[5] Luigi Crisci, Majid Salimi Beni, Biagio Cosenza, Nicolò Scipione, Davide Gadioli, Emanuele Vitali, Gianluca Palermo, and Andrea Beccari. 2022. Towards a Portable Drug Discovery Pipeline with SYCL 2020. In *International Workshop on OpenCL* (Bristol, United Kingdom, United Kingdom) *(IWOCL'22)*. Association for Computing Machinery, New York, NY, USA, Article 5, 2 pages.

[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] Davide Gadioli, Emanuele Vitali, Federico Ficarelli, Chiara Latini, Candida Manelfi, Carmine Talarico, Cristina Silvano, Carlo Cavazzoni, Gianluca Palermo, and Andrea Rosario Beccari. 2023. EXSCALATE: An Extreme-Scale Virtual Screening Platform for Drug Discovery Targeting Polypharmacology to Fight SARS-CoV-2. *IEEE Transactions on Emerging Topics in Computing* 11, 1 (2023), 170–181.

[8] Zheming Jin and Jeffrey S. Vetter. 2023. A Benchmark Suite for Improving Performance Portability of the SYCL Programming Model. In *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 325–327. https://doi.org/10.1109/ISPASS57527.2023.00041

[9] 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

[10] Núria López, Luigi Del Debbio, Marc Baaden, Matej Praprotnik, Laura Grigori, Catarina Simões, Serge Bogaerts, Florian Berberich, Thomas Lippert, Janne Ignatius, Philippe Lavocat, Oriol Pineda, Maria Grazia Giuffreda, Sergi Girona, Dieter Kranzlmüller, Michael M. Resch, Gabriella Scipione, and Thomas Schulthess. 2021. Lessons learned from urgent computing in Europe: Tackling the COVID-19 pandemic. *Proceedings of the National Academy of Sciences* 118, 46 (2021), e2024891118.

[11] Stefano Markidis, Davide Gadioli, Emanuele Vitali, and Gianluca Palermo. 2021. Understanding the I/O Impact on the Performance of High-Throughput Molecular Docking. In *2021 IEEE/ACM Sixth International Parallel Data Systems Workshop (PDSW)*. 9–14.

[12] George S. Markomanolis, Aksel Alpay, Jeffrey Young, Michael Klemm, Nicholas Malaya, Aniello Esposito, Jussi Heikonen, Sergei Bastrakov, Alexander Debus, Thomas Kluge, Klaus Steiniger, Jan Stephan, Rene Widera, and Michael Bussmann. 2022. Evaluating GPU Programming Models for the LUMI Supercomputer. In *Supercomputing Frontiers: 7th Asian Conference, SCFA 2022, Singapore, March 1–3, 2022, Proceedings* (Singapore, Singapore). Springer-Verlag, Berlin, Heidelberg, 79–101.

[13] Aristotle Martin, Geng Liu, William Ladd, Seyong Lee, John Gounley, Jeffrey Vetter, Saumil Patel, Silvio Rizzi, Victor Mateevitsi, Joseph Insley, and Amanda Randles. 2023. Performance Evaluation of Heterogeneous GPU Programming Frameworks for Hemodynamic Simulations. In *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W '23)*. Association for Computing Machinery, New York, NY, USA, 1126–1137.

[14] Hans Matter and Christoph Sotriffer. 2011. *Applications and Success Stories in Virtual Screening*. John Wiley & Sons, Ltd, Chapter 12, 319–358.

[15] Natarajan Arul Murugan, Artur Podobas, Davide Gadioli, Emanuele Vitali, Gianluca Palermo, and Stefano Markidis. 2022. A Review on Parallel Virtual Screening Softwares for High-Performance Computers. *Pharmaceuticals* 15, 1 (2022), 63.

[16] Nataraj S Pagadala, Khajamohiddin Syed, and Jack Tuszynski. 2017. Software for molecular docking: a review. *Biophysical reviews* 9, 2 (2017), 91–102.

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

[18] 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

[19] S. John Pennycook, Jason D. Sewall, Douglas W. Jacobsen, Tom Deakin, and Simon McIntosh-Smith. 2021. Navigating Performance, Portability, and Productivity. *Computing in Science and Engineering* 23, 5 (2021), 28–38. https://doi.org/10.1109/MCSE.2021.3097276

[20] Esteban Miguel Rangel, Simon John Pennycook, Adrian Pope, Nicholas Frontiere, Zhiqiang Ma, and Varsha Madananth. 2023. A Performance-Portable SYCL Implementation of CRK-HACC for Exascale. In *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W '23)*. Association for Computing Machinery, New York, NY, USA, 1114–1125.

[21] Srikanth Sathyanarayana, Matteo Bernardini, Davide Modesti, Sergio Pirozzoli, and Francesco Salvadore. 2023. High-speed turbulent flows towards the exascale: STREAmS-2 porting and performance. arXiv:2304.05494

[22] Leonardo Solis-Vasquez, Edward Mascarenhas, and Andreas Koch. 2023. Experiences Migrating CUDA to SYCL: A Molecular Docking Case Study. In *Proceedings of the 2023 International Workshop on OpenCL (IWOCL '23)*. Association for Computing Machinery, New York, NY, USA, Article 15, 11 pages.

[23] Emanuele Vitali, Federico Ficarelli, Mauro Bisson, Davide Gadioli, Gianmarco Accordi, Massimiliano Fatica, Andrea R. Beccari, and Gianluca Palermo. 2024. GPU-optimized approaches to molecular docking-based virtual screening in drug discovery: A comparative analysis. *J. Parallel and Distrib. Comput.* 186 (2024).