Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 240 (2024) 42–51

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the First EuroHPC user day

A Portable Drug Discovery Platform for Urgent Computing

Davide Gadioli[a,*], Gianmarco Accordi[a], Jan Krenek[c], Martin Golasowski[c], Ladislav Foltyn[c], Jan Martinovic[c], Andrea R. Beccari[b], Gianluca Palermo[a]

[a] *Politecnico di Milano - DEIB, Milano, Italy*
[b] *EXSCALATE - Dompé Farmaceutici SpA, Naples, Italy*
[c] *IT4Innovations, VSB – Technical University of Ostrava, Ostrava-Poruba, Czech Republic*

Abstract

Drug discovery is a long and costly process. Recent studies demonstrated how the introduction of an in-silico stage, named virtual screening, that suggests which molecule to test in-vitro, increases the drug discovery success probability. In the context of urgent computing, where it is important to find a therapeutic solution in a short time frame, the number of candidates that we can virtual screen is limited only by the available computation power. In this paper, we focus on LiGen, the virtual screening application of the EXSCALATE platform. In particular, we address two challenges of performing an extreme-scale virtual screening on a modern HPC system. The first one is posed by hardware heterogeneity, where GPUs of different vendors account for a large fraction of their performance. The second challenge concerns the operational difficulties of running the campaign since it requires significant effort and technical skills that are not common among domain experts. We show how hinging on SYCL and the LEXIS Platform, is the solution that the EXSCALATE Platform uses to address these challenges.

Keywords: HPC; Virtual Screening; Performance Portability; Urgent Computing

1. Introduction

Drug discovery aims at finding a molecule with a small molecular weight that yields a therapeutic effect against the target disease. This is a long and expensive process, with a success probability lower than 25% [15]. One challenge is due to the sheer size of the chemical space that we can access to design drug candidates, which is estimated to include 10^{33} molecules [20]. The problem arises when we consider that it is hard to test *in-vitro* more than 10^5 compounds per day [14], forcing pharmaceutical companies to rely only on their expertise to select which molecule to test. Recent studies demonstrated how the introduction of an *in-silico* stage that virtual screens an input chemical library, suggesting which are the most promising candidates, improves the drug discovery success probability [13, 3].

* Corresponding author.
E-mail addresses: davide.gadioli@polimi.it (Davide Gadioli)., gianmarco.accordi@polimi.it (Gianmarco Accordi)., jan.krenek@vsb.cz (Jan Krenek)., martin.golasowski@vsb.cz (Martin Golasowski)., ladislav.foltyn@vsb.cz (Ladislav Foltyn)., jan.martinovic@vsb.cz (Jan Martinovic)., andrea.beccari@dompe.com (Andrea R. Beccari)., gianluca.palemro@polimi.it (Gianluca Palermo).

10.1016/j.procs.2024.07.007

The virtual screening stage estimates the interaction strength between a drug candidate, named *ligand*, and at least one binding site of the target protein(s), representing the target disease. Domain experts can use this information to rank a chemical library for selecting the most promising candidates to test in vitro. To estimate the interaction strength we need to solve two well-known problems: molecular docking and scoring [17, 19, 21]. The former estimates the 3D displacement of the ligand atoms when they interact with the protein, which is a requirement for the latter to evaluate the intermolecular forces. Both tasks are complex and compute-intensive. Since the computation of each protein-ligand pair is independent of the others, this is an embarrassingly parallel problem. For these reasons, HPC systems are the ideal machines for a virtual screening campaign. In the context of urgent computing [11], where it is important to find a solution in a short time frame, HPC usage becomes of paramount importance. For example, the largest virtual screening campaign ranked 72 billions of ligands, against 15 binding sites of 12 viral proteins of the SARS-CoV-2 [7]. The computation lasted 60 hours, using a significant fraction of the computation nodes of Marconi100 at CINECA, and of HP5 at ENI, which have a combined sustained throughput of 81 petaflops. The results are under analysis and publicly available [1].

In this paper we focus on LiGen, the virtual screening application of the EXSCALATE platform [7], that has been designed from scratch to hinge on all the computation resources provided by an HPC system. In particular, we report the main two challenges of extreme-scale virtual screening campaigns from the computation perspective, and how we are addressing them. The first challenge is posed by hardware heterogeneity. If we look at the TOP500 list[2], which ranks the most powerful HPC systems in terms of FLOPS, we can notice how the top 4 systems use different architectures. While 3 of them rely on GPUs to accelerate computation, they are made by different vendors, each with its own native programming language. For the virtual screening campaign, we aim for performance portability, not only functional portability, because the probability of finding good candidates increases with the number of evaluated molecules, for any given computation budget. Indeed, we can spend the time that we spare from an increase in computation efficiency by evaluating more ligands. In the context of the EuroHPC project LIGATE[3], we address this challenge by porting in SYCL 2020 the computation kernels [6] and out-of-kernel optimizations [2]. We validated LiGen using NVIDIA and AMD-based GPUs [1].

The second challenge is posed by the operational effort required to carry out the computation. LiGen, as most of the HPC applications [4], hinge on MPI to scale over the available nodes. When an MPI process aborts due to a fault, the default response by the standard is to kill the remaining processes. In the context of an extreme-scale virtual screening campaign, it means that we will lose all the results that have not been stored in the file system. Since IO is a scarce resource in the HPC system, it is common to accumulate the output, either in RAM [12] or in the node local storage [8], before storing it on the shared file system. To solve this problem, the EXSCALATE platform divides the virtual screening campaign into smaller tasks, using custom reactive tools to execute them through the system job scheduler. While effective, this solution needs to be tailored for each HPC center and requires human supervision with technical skills that are not common among domain experts. For this reason, in the context of the EuroHPC project LIGATE, we integrated LiGen into the LEXIS Platform [9]. The latter enables access to HPC systems and automatizes workflows through an intuitive web interface. Moreover, it can abstract the execution across different HPC systems, either on-premises or European supercomputer centers. In the case of an urgent computing scenario, it is possible to spawn the virtual screening campaign across multiple locations without requiring a dedicated queue.

The remainder of the paper is structured as follows. Section 2 focuses on the performance portability challenges, providing more technical details on how LiGen can run efficiently on a wide range of HPC-grade GPUs and scale over a large number of nodes. Then, Section 3 focuses on the second challenge, describing the technologies behind the LEXIS Platform and how it can be used to carry out a virtual screen campaign. Finally, Section 4 concludes the paper.

[1] Scientific papers and campaign results: https://www.exscalate4cov.eu/contribute.html
[2] TOP500 list: https://www.top500.org
[3] Website: https://www.ligateproject.eu

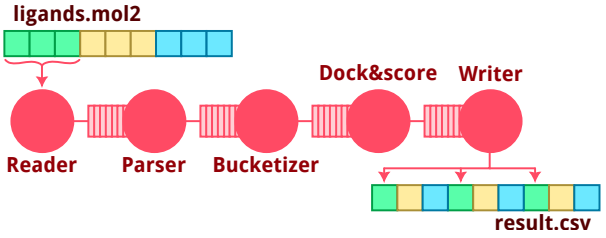

Fig. 1. Example of an MPI process that implements a LiGen computation pipeline for virtual screening. All the stages are represented by red circles, while input queues are by slim red rectangles. The input files are partitioned by a single IO request, and color-coded according to the rank of the MPI process that issues the request. In the example, we consider the MPI process related to the green color.

2. Performance portability in virtual screening

This section describes the LiGen application and how it can scale across the supercomputing resources. Then, we focus on the computation kernels and their SYCL porting, highlighting their optimization on the Karolina supercomputer at IT4i. Finally, we measure how LiGen can run efficiently on different HPC-grade GPUs.

2.1. The LiGen virtual screening application

LiGen is a `C++17` application that uses MPI to scale across different nodes. The idea is to use a single MPI process within each node that spawns the same asynchronous computation pipeline. Each stage of the pipeline uses at least one CPU thread to take a ligand from its input queue, perform a computation, and then enqueue the ligand in the next stage. The input queues have a maximum size. This means that the slower stage generates back-pressure on the queues stalling the previous stages. If we assign to the slower stage a number of CPU threads equal to the number of hardware threads, on average, we will use the node to compute the most expensive operation, which is the desired scenario. The only stages that require synchronization with the other MPI processes are the ones that perform IO operations.

In the virtual screening scenario, we have two types of input files. One type includes all the files required to describe the target protein(s). The second one represents the chemical library that we need to virtual screen. The output file decorates the input molecules with the interaction strength against the target protein(s). LiGen considers the target protein(s) as constant, thus it will read them using MPI IO collective functions at the beginning of the execution, providing the same information to all the MPI processes. In this way, all the stages can have access to target(s) without any need for synchronization. On the other hand, LiGen statically splits the ligand input file among the available processes based on the file size. For performance reason [12], each MPI process uses private IO operations to read the input file at a pace that depends on the throughput of the slower pipeline stage. To minimize the imbalance between different MPI processes, the EXSCALATE platform uses a pre-processing step to uniform the complexity within the input file [7]. However, we need to synchronize the write operation, to avoid that MPI processes will overwrite each other. In this case, we hinge on the fact that the order among ligands of the input file is not important. In particular, the user can select how many MPI processes will collect the output and issue the actual IO operation to the file system.

Figure 1 depicts an example of a pipeline to perform virtual screening. In the example, three MPI processes replicate the same pipeline that works on different slabs of the input file. The first stage reads a chunk of text that contains the description of zero, one, or more ligands, depending on the chunk size, which is a configuration parameter [12]. The next stage parses the description of each ligand to create and populate the related data structures that can be used by all the other stages. Then, a bucketizer stage uses input features to cluster ligands that have similar sizes, in terms number of heavy atoms and rotatable bonds. Once the bucketizer fills a bucket of ligands it will forward them to the dock & score stage that performs the actual computation. Finally, the writer stage will generate a CSV file that relates a molecule with the interaction strength against the target(s).

LiGen uses MPI to scale across different nodes, and `std::thread` to scale within the cores of a node. In particular, it is possible to configure the number of threads that carry out the computation of each stage. They use work stealing for the input queue of ligands. Usually, it is the dock & score stage that defines the pace of the whole pipeline. This implies that the thread that executes the writer stage will be idle waiting for input, while all the threads involved in the

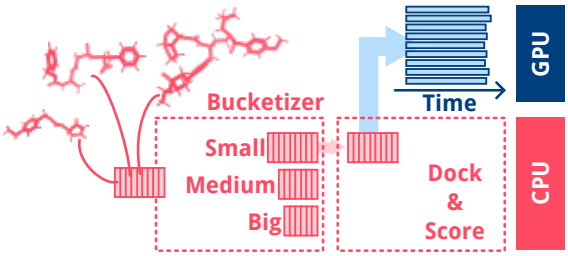

Fig. 2. Interaction between the bucketizer and dock & score stage. The red part represents CPU computation, while the blue part represents the GPU computation.

reader, parser, and bucketizer stages will be waiting due to the back pressure. In this scenario, all the nodes involved in the computation will be busy docking and scoring for most of the time, which is the desired outcome.

To address the hardware heterogeneity, the dock & score stage may use different implementations to carry out the computation. Besides the `C++17` one, we can offload computation to GPUs using a CUDA and a SYCL implementation. At the beginning of the execution, LiGen will allocate memory on each available GPU, setting up a double buffering approach. Then, when a CUDA or SYCL implementation needs to compute a bucket, it will handle memory transfer and computation on the first available GPU. As a rule of thumb, we need to use a number of dock & score threads, that use a CUDA or SYCL implementation, equal to twice the number of available GPUs, to benefit from double buffering. By using this approach we can hinge on all the resources available in a computation node.

2.2. The computation kernel structure and SYCL implementation

The first LiGen version that could offload computation on a GPU, was using a CUDA implementation that spread the computation of a ligand on all the GPU resources. It seldom happens that a ligand is big enough to use all the resources of an HPC-grade GPU. For this reason, each thread that uses the CUDA implementation in the dock & score stage computes the ligand in its own CUDA stream. The main issue with this approach is that the synchronization required by some kernels hindered the computation efficiency[23].

To achieve full GPU occupancy, we shifted from this latency-oriented approach to a throughput one. In the throughput version, a batch of ligands is computed in parallel at a given point in time. The main difference is that we use few resources to compute each ligand. While the execution time required to process a single ligand increases, the overall throughput is better [23]. To improve computation efficiency, ligands with similar resource usage are packed into the same batch, also called buckets. At each moment in time, the GPU is computing only one bucket. This approach can yield a throughput that is 5× the classical one if we can fine-tune batches to reach full GPU occupancy [2]. In our case, input features like the number of ligand atoms, influence the resources required to compute a ligand. Therefore, the size and number of batches depend on the input feature and how we choose to cluster them. To reach full GPU occupancy, we need to compute as many ligands in parallel as they fit in the GPU hardware and make sure that their computation will last for a similar amount of time [2].

Figure 2 shows how the bucketizer and dock & score stage solve this problem. In particular, the bucketizer bundles all the ligands that have a similar number of heavy atoms and rotatable bonds in the same bucket. Indeed, these input features have a strong correlation with the execution time [7]. When we look at the distribution of the number of atoms in a dataset, it is very common to find ligands with 30-40 atoms, but there are also few ligands with more than 150 atoms. The higher the number of ligand atoms, the higher the resource usage. For this reason, the computation kernels have non-type template parameters that define the maximum number of atoms. To increase the number of ligands that run in parallel, each kernel is instantiated for a suitable maximum number of atoms. The idea is that given the same GPU resource, batches containing smaller ligands may pack a higher number of ligands[2]. At compile time, LiGen fixes the ranges of atom numbers a kernel has to compute, thus the batch number. While at runtime, LiGen evaluates the running environment and the resources available, selecting how many ligands we can pack in a batch [2]. This step is of paramount importance to automatically tailor the execution on the underlying hardware.

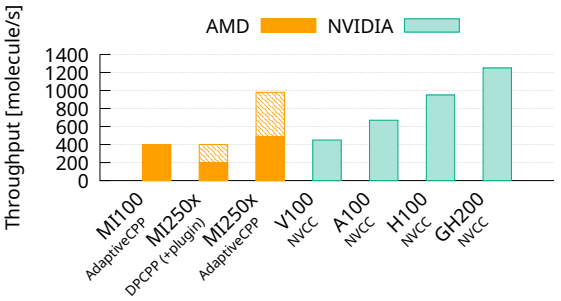

Fig. 3. LiGen average throughput, by varying target GPU, language, and compiler. The performance refers to a single physical card. For the MI250x GPU, we need to consider two virtual GPUs to make a fair comparison.

In the LIGATE European project, we developed a SYCL porting of LiGen [5] to unlock performance portability on several EuroHPC supercomputing centers. This development and optimization activity has been carried out using mainly NVIDIA cards. To validate the performance portability approach, we had access to LUMI-G, which ships AMD cards [1].

2.3. Performance evaluation

If we consider the best-performing supercomputers ranked in the TOP500 list, we can notice how the largest fraction of their throughput hinge on accelerators. In almost all cases, the accelerator is a GPU. In the past only NVIDIA cards were available, leading to an effort of porting existing code in the CUDA language. However, modern supercomputers use GPUs from different vendors, with their own native language.

In the case of urgent computing, where it is important to run a virtual screening campaign on a supercomputer, we need to handle this hardware heterogeneity. This section aims at evaluating and comparing the LiGen performance on a wide range of HPC-grade GPUs, to prove that we can reach performance portability. Figure 3 reports the average throughput, in terms of ligands screened per second. This measures the time of the whole execution, taking into account also communication overhead with the file system and the GPU memory. To make a fair comparison, we consider only a single physical card. In the case of the AMD MI250x, we need to take into account that each physical card is composed of two logical ones, that's why we have highlighted the related bar with two different patterns. The main takeaway message from this experiment is the relative difference in throughput between the different HPC-grade GPUs. When we focus on each vendor, we can notice how the performance will grow after each new generation. If we compare the GPU of different vendors, we can notice how the AMD MI250x, which is the GPU card equipped by the LUMI-G supercomputer, is greater than the NVIDIA A100, which is the GPU card equipped by the Karolina supercomputer. This result confirms that we reached performance portability across European supercomputers since the two cards were launched in a similar period of time. Moreover, we had a grant for a development access call on the Karolina supercomputer that we used to optimize and tune the LiGen performance, in terms of both quality and performance. Furthermore, we used a benchmark access call on LUMI to measure the performance and check whether the performance was portable, on a completely different supercomputer [1]. However, if we compare the FLOPS declared by the two GPUs, we can notice how the AMD card is supposed to yield a much higher throughput than the NVIDIA one. For example, we noticed how the SYCL compiler produces a high register pressure [5]. This hints that there is still room for improvement in the SYCL implementation.

3. Orchestrate an extreme-scale virtual screening campaign

This section describes more in detail how the LEXIS Platform can orchestrate a virtual screening campaign. We begin with a general description of the LEXIS Platform [4] [9] and its features, including generic workflow orchestration

[4] LEXIS Platform documentation - https://docs.lexis.tech

and data movement across multiple sites. Finally, we show how the user launches the screening campaign through the platform's web interface.

3.1. The LEXIS Platform

The LEXIS Platform provides services for complex computing workflow orchestration and distributed data management across connected HPC sites. The platform provides an API and a web-based user interface for easy and unified access to multiple powerful HPC resources. The workflows orchestrated by the platform can be described as directed acyclic graphs (DAGs) that consist of various dependent tasks that trigger and monitor data movement, Kubernetes container, and HPC batch job operations. The orchestrator used by the platform is built on top of Apache Airflow [10], which uses Python files to describe the workflow task dependencies and their parameters. The platform also allows a declarative approach to workflow description based on YAML files using the TOSCA standard [18]. The Apache Airflow has a concept of DAG runs, which can be seen as instances of a particular workflow with a set of parameter values defined at the beginning of the execution. Thus, executions of the same workflow with various parameter values can be tracked in a unified way.

The LEXIS Platform defines an abstraction to work with different compute projects, user groups, and resource allocations. The basic element of the platform is a LEXIS Project, which has its own set of users with various associated roles and a set of resource allocations. Each project has its own datasets, workflows, and users. The resource abstraction is used by an HPC-as-a-Service middleware called HEAppE [22] deployed near each connected HPC cluster to manage credentials and access to particular HPC applications through HEAppE Command Templates. These templates are essentially parameterized job scripts used to launch only a precisely defined set of commands on the clusters through the HEAppE API which is used by the LEXIS Orchestrator. In this way, the user is completely isolated from the cluster-specific configuration and does not have the possibility to directly access the executable files of the application. Technical details of this implementation are out of the scope of this paper and can be found in [9]. These templates are usually created by an HPC application specialist who can use them to hide site-specific options (i.e. MPI parameters) inside from the regular users of the application.

The described infrastructure is used by the LEXIS Orchestrator during the execution of the workflow. Thanks to the branching capability provided by the Airflow DAGs, various situations can be handled, such as job failures at a particular site. The DAG can contain a branch that handles this situation by triggering a transfer of all available checkpoint data to another site and triggering a job submission there. The simulation then loads the transferred checkpoints and continues with its execution. The workflow also contains cleanup tasks which are usually triggered only for successful execution, leaving the logs and potential by-products to be examined by a specialist in case of job failure (displayed in Figure 4). The user has the possibility, not only in case of DAG failure, to check the *stdout* generated by the applications through the graphical user interface and a part of the DAG can be re-executed if needed.

3.2. Data movement

The LEXIS Platform provides data movement capabilities through its distributed data interface (DDI). The DDI has functions for data transfer between the user and a target site, as well as staging features for moving data to and from HPC clusters and other compute resources. The data in the DDI are defined as datasets, which can be seen as a classical directory with a rich set of metadata [16]. These metadata are indexed in a centralized Elastic Search index and at the same time in iRODS zones which are typically hosted by the HPC sites. The data is stored inside the iRODS collections. The DDI then uses a set of custom-built APIs and asynchronous worker processes that perform the actual data movement using native protocols. For example, data transfers between locations use native iRODS protocol, which leverages parallel TCP streams, native POSIX protocols common in the HPC environment such as SFTP or RSync and HTTPS chunked transfers for web-based front-ends or similar clients.

These functionalities are leveraged by the multi-site workflow of LiGen, where the input data are stored as DDI datasets, and reference to them is passed to the workflow at execution time. The workflow then uses the DDI capabilities to trigger dataset movement to a target location. The same mechanism is used to store the result dataset along with the necessary metadata. The DDI recognizes multiple levels of access to the datasets:

- *user* – datasets are visible only by a particular user in a given LEXIS Project;

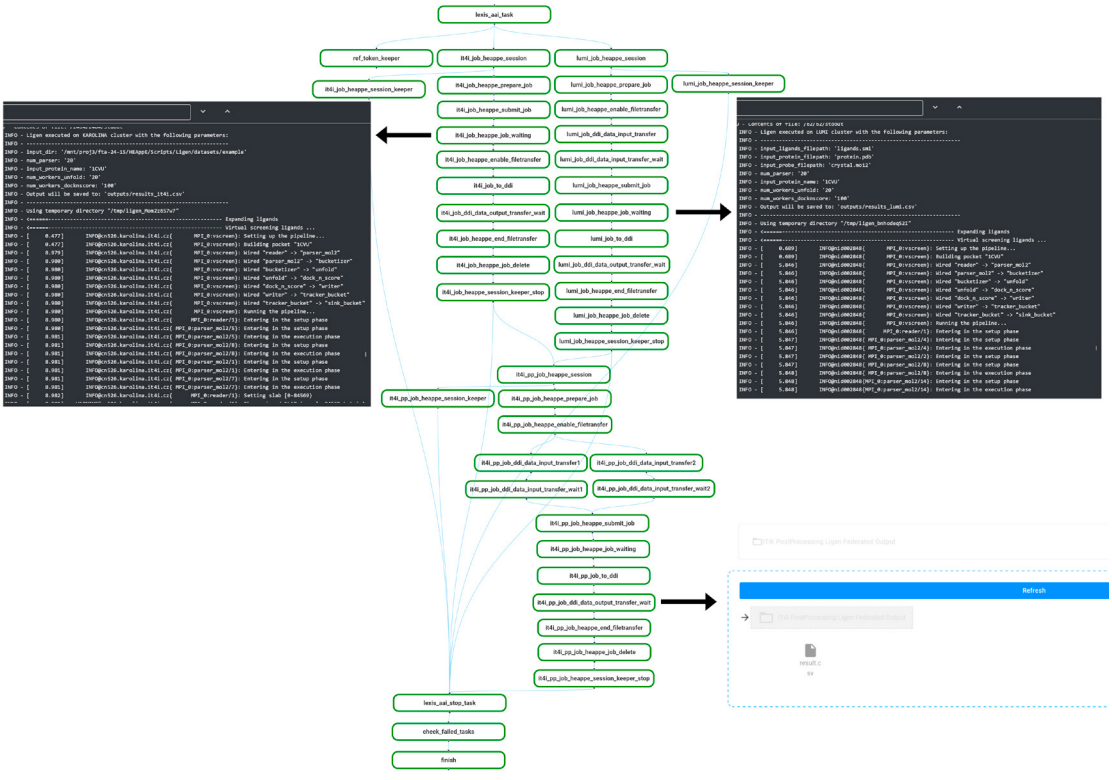

Fig. 4. Example LEXIS workflow with HPC application logs, and workflow results stored in LEXIS DDI

- *project* – shared among all project users;
- *public* – available to all logged-in users of the LEXIS Platform.

Each site used in execution has an HEAppE Middleware and a DDI worker process deployed (usually on a virtual machine). The HEAppE is used apart from the HPC job management and also for data transfer credentials management by the DDI worker process.

3.3. Interaction with the user

The user interaction with the LiGen through the LEXIS Platform is done using a web-based interface. A regular user of LiGen wants to have the capability to specify input data, launch the LiGen application on multiple HPC sites without the need to interact with a particular HPC cluster or job scheduler and download and examine the results. The user can log in to the LEXIS Portal using one of the identity providers (EUDAT B2ACCESS [5] or MyAccessID [6]) which cover most of the European institutional identities as well as some of the public commercial ones.

Once logged in, the user can request access to a particular LEXIS Project, which has the LiGen workflow associated, as well as the allocation on one of the desired HPC clusters. The owner of the LEXIS Project needs to approve the user's request. Once approved, the user sees the LiGen workflow in the workflow list, as well as project and public datasets. The user may choose to upload their own dataset or use one of the existing ones.

[5] B2ACCESS: https://eudat.eu/service-catalogue/b2access
[6] MyAccessID: https://wiki.geant.org/display/MyAccessID

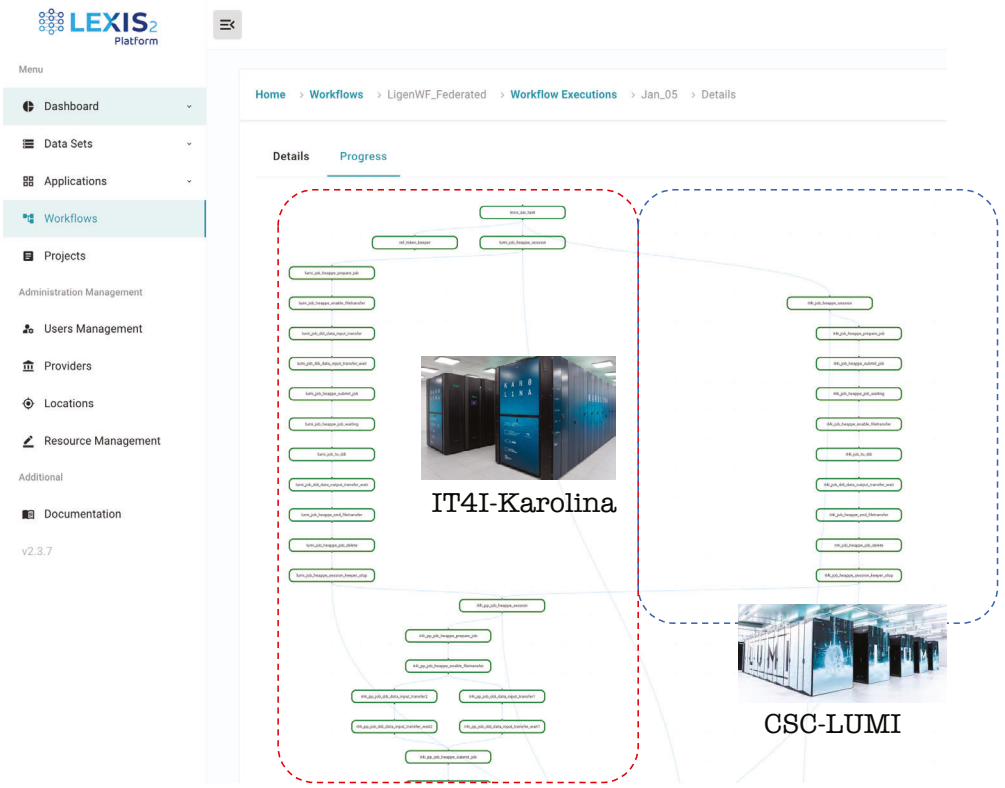

Fig. 5. LEXIS multisite execution of the LiGen-based Drug-Discovery workflow on IT4Innovations-Karolina and CSC-LUMI supercomputers

Once done, the user selects a particular workflow for execution, selects the input datasets, and may change a set of the input parameter values of the application. This workflow is tied to two locations, the Karolina petascale (located at IT4Innovations in Ostrava, Czech Republic) and LUMI pre-exascale (located at CSC's data center in Kajaani, Finland) supercomputers. The principal investigator for each resource allocation must approve this association in advance. Details of this process are beyond the scope of this paper. The screenshot from the LEXIS Portal containing workflow execution is displayed in Figure 5. It is also possible to use a dynamic selection of locations using smart scheduler capabilities, through the selection of a scheduling policy. In both cases, the LEXIS Project must have compute time allocations on all involved HPC sites associated. When we consider the virtual screening workflow for urgent computing, we could reach up to 12M ligands per second of throughput if we target all computation nodes on the LUMI and Karolina supercomputing centers. In an extreme-scale experiment for urgent computing, the size of the input ligands is in the order of TBs in SMILES format [7]. However, the data staging on each location happens before starting the computation, without impacting the throughput.

Finally, the user fills out the necessary parameters and triggers the execution. The execution progress can be monitored in the web-based UI as well as logs from a particular HPC job can be read for debugging purposes if needed. Once finished, the execution detail contains a link to a result dataset, which can be downloaded through the web browsers and examined. This approach enables an end-user to use HPC resources and third-party software without having direct access to them. This permits on one side to offer a virtual screening-as-a-service platform to researchers. On the other side, it protects the intellectual properties of the pharmaceutical company, as in our scenario.

4. Conclusion

Virtual screening is becoming an important stage in drug discovery for suggesting which are the candidates to the test *in-vitro*. Since the problem is embarrassingly parallel and the size of the chemical library that we want to

virtual screen is limited only by the available computation power, HPC systems are the ideal candidate to carry out the computation. In this paper, we focus on the challenge of performance portability across heterogeneous hardware and, in the context of urgent computing, orchestrate an extreme-scale virtual screening campaign. In particular, we show how we are addressing them in LiGen, the EXSCALATE platform virtual screening application. From experimental results, we can notice how a SYCL implementation enables LiGen to be deployed on supercomputers that also have a non-NVIDIA GPU. If we compare its performance with the native hand-optimized CUDA version on NVIDIA GPU, we can notice that LiGen run uses the resources efficiently, even if there is still room for improvement. Moreover, the usage of the LEXIS Platform provides an elegant way for domain experts to perform an extreme-scale virtual screening campaign on one or more European supercomputers.

Acknowledgements

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 also acknowledge EuroHPC-JU for awarding us access to Karolina at IT4Innovations, Czech Republic (EHPC-DEV-2021D02-049), and to LUMI-G at CSC, Finland (EHPC-BEN-2022B12-001). This work was also partly-supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254).

References

[1] Accordi, G., Gadioli, D., Palermo, G., Crisci, L., Carpentieri, L., Cosenza, B., Beccari, A.R., 2024a. Unlocking performance portability on lumi-g supercomputer: A virtual screening case study, in: Proceedings of the 12th International Workshop on OpenCL and SYCL, Association for Computing Machinery, New York, NY, USA. URL: https://doi.org/10.1145/3648115.3648125, doi:10.1145/3648115.3648125.

[2] Accordi, G., Gadioli, D., Vitali, E., Crisci, L., Cosenza, B., Beccari, A., Palermo, G., 2024b. Out of kernel tuning and optimizations for portable large-scale docking experiments on GPUs. The Journal of Supercomputing , 1–18.

[3] Allegretti, M., Cesta, M.C., Zippoli, M., Beccari, A., Talarico, C., Mantelli, F., Bucci, E.M., Scorzolini, L., Nicastri, E., 2022. Repurposing the estrogen receptor modulator raloxifene to treat SARS-CoV-2 infection. Cell Death & Differentiation 29, 156–166.

[4] Bernholdt, D.E., Boehm, S., Bosilca, G., Gorentla Venkata, M., Grant, R.E., Naughton, T., Pritchard, H.P., Schulz, M., Vallee, G.R., 2020. A survey of mpi usage in the us exascale computing project. Concurrency and Computation: Practice and Experience 32, e4851. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4851, doi:https://doi.org/10.1002/cpe.4851, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4851. e4851 cpe.4851.

[5] Crisci, L., Carpentieri, L., Cosenza, B., Accordi, G., Gadioli, D., Vitali, E., Palermo, G., Beccari, A.R., 2024. Enabling performance portability on the ligen drug discovery pipeline. Future Generation Computer Systems URL: https://www.sciencedirect.com/science/article/pii/S0167739X24001195, doi:https://doi.org/10.1016/j.future.2024.03.045.

[6] Crisci, L., Salimi Beni, M., Cosenza, B., Scipione, N., Gadioli, D., Vitali, E., Palermo, G., Beccari, A., 2022. Towards a portable drug discovery pipeline with sycl 2020, in: Proceedings of the 10th International Workshop on OpenCL, Association for Computing Machinery, New York, NY, USA. URL: https://doi.org/10.1145/3529538.3529688, doi:10.1145/3529538.3529688.

[7] Gadioli, D., Vitali, E., Ficarelli, F., Latini, C., Manelfi, C., Talarico, C., Silvano, C., Cavazzoni, C., Palermo, G., Beccari, A.R., 2022. 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, 170–181.

[8] Glaser, J., Vermaas, J.V., Rogers, D.M., Larkin, J., LeGrand, S., Boehm, S., Baker, M.B., Scheinberg, A., Tillack, A.F., Thavappiragasam, M., Sedova, A., Hernandez, O., 2021. High-throughput virtual laboratory for drug discovery using massive datasets. The International Journal of High Performance Computing Applications 35, 452–468. doi:10.1177/10943420211001565.

[9] Golasowski, M., Martinovič, J., Křenek, J., Slaninová, K., Levrier, M., Harsh, P., Derquennes, M., Donnat, F., Terzo, O., 2022. The lexis platform for distributed workflow execution and data management, in: HPC, Big Data, and AI Convergence Towards Exascale. Taylor & Francis.

[10] Harenslak, B., de Ruiter, J., 2021. Data Pipelines with Apache Airflow. Manning Publications. ISBN: 9781617296901.

[11] López, N., Debbio, L.D., Baaden, M., Praprotnik, M., Grigori, L., Simões, C., Bogaerts, S., Berberich, F., Lippert, T., Ignatius, J., Lavocat, P., Pineda, O., Giuffreda, M.G., Girona, S., Kranzlmüller, D., Resch, M.M., Scipione, G., Schulthess, T., 2021. Lessons learned from urgent computing in europe: Tackling the covid-19 pandemic. Proceedings of the National Academy of Sciences 118, e2024891118. doi:10.1073/pnas.2024891118.

[12] Markidis, S., Gadioli, D., Vitali, E., Palermo, G., 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), pp. 9–14. doi:10.1109/PDSW54622.2021.00007.

[13] Matter, H., Sotriffer, C., 2011. Applications and Success Stories in Virtual Screening. John Wiley & Sons, Ltd. chapter 12. pp. 319–358. ISBN: 9783527633326.

[14] Mayr, L.M., Fuerst, P., 2008. The future of high-throughput screening. SLAS Discovery 13, 443–448. URL: https://www.sciencedirect.com/science/article/pii/S2472555222082260, doi:https://doi.org/10.1177/1087057108319644.

[15] Morgan, S., Grootendorst, P., Lexchin, J., Cunningham, C., Greyson, D., 2011. The cost of drug development: A systematic review. Health Policy 100, 4–17. URL: https://www.sciencedirect.com/science/article/pii/S0168851010003659, doi:https://doi.org/10.1016/j.healthpol.2010.12.002.

[16] Munke, J., Hayek, M., Golasowski, M., García-Hernández, R.J., Donnat, F., Koch-Hofer, C., Couvee, P., Hachinger, S., Martinovič, J., 2022. Chapter 4 Data System and Data Management in a Federation of HPC/Cloud Centers. Taylor & Francis.

[17] Murugan, N.A., Podobas, A., Gadioli, D., Vitali, E., Palermo, G., Markidis, S., 2022. A review on parallel virtual screening softwares for high-performance computers. Pharmaceuticals 15, 63.

[18] OASIS, 2020. Tosca simple profile in yaml version 1.3. URL: https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/TOSCA-Simple-Profile-YAML-v1.3.pdf.

[19] Pagadala, N.S., Syed, K., Tuszynski, J., 2017. Software for molecular docking: a review. Biophysical reviews 9, 91–102.

[20] Polishchuk, P.G., Madzhidov, T.I., Varnek, A., 2013. Estimation of the size of drug-like chemical space based on gdb-17 data. Journal of computer-aided molecular design 27, 675–679.

[21] Su, M., Yang, Q., Du, Y., Feng, G., Liu, Z., Li, Y., Wang, R., 2019. Comparative assessment of scoring functions: The casf-2016 update. Journal of Chemical Information and Modeling 59, 895–913. doi:10.1021/acs.jcim.8b00545.

[22] Svaton, V., Martinovic, J., Krenek, J., Esch, T., Tomancak, P., 2020. Hpc-as-a-service via heappe platform, in: Barolli, L., Hussain, F.K., Ikeda, M. (Eds.), Complex, Intelligent, and Software Intensive Systems, Springer International Publishing. pp. 280–293. doi:10.1007/978-3-030-22354-0_26.

[23] Vitali, E., Ficarelli, F., Bisson, M., Gadioli, D., Accordi, G., Fatica, M., Beccari, A.R., Palermo, G., 2024. GPU-optimized approaches to molecular docking-based virtual screening in drug discovery: A comparative analysis. Journal of Parallel and Distributed Computing 186, 104819. URL: https://www.sciencedirect.com/science/article/pii/S0743731523001892, doi:https://doi.org/10.1016/j.jpdc.2023.104819.