EXSCALATE: An Extreme-Scale Virtual Screening Platform for Drug Discovery Targeting Polypharmacology to Fight SARS-CoV-2

Davide Gadioli, Emanuele Vitali, Federico Ficarelli, Chiara Latini, Candida Manelfi, Carmine Talarico, Cristina Silvano, *Fellow, IEEE,* Carlo Cavazzoni, Gianluca Palermo, *Senior Member, IEEE,* Andrea Rosario Beccari

Abstract—The social and economic impact of the COVID-19 pandemic demands a reduction of the time required to find a therapeutic cure. In this paper, we describe the EXSCALATE molecular docking platform capable to scale on an entire modern supercomputer for supporting extreme-scale virtual screening campaigns. Such virtual experiments can provide in short time information on which molecules to consider in the next stages of the drug discovery pipeline, and it is a key asset in case of a pandemic. The EXSCALATE platform has been designed to benefit from heterogeneous computation nodes and to reduce scaling issues. In particular, we maximized the accelerators' usage, minimized the communications between nodes, and aggregated the I/O requests to serve them more efficiently. Moreover, we balanced the computation across the nodes by designing an ad-hoc workflow based on the execution time prediction of each molecule. We deployed the platform on two HPC supercomputers, with a combined computational power of 81 PFLOPS, to evaluate the interaction between 70 billion of small molecules and 15 binding-sites of 12 viral proteins of SARS-CoV-2. The experiment lasted 60 hours and it performed more than one trillion ligand-pocket evaluations, setting a new record on the virtual screening scale.

Index Terms—Extreme-scale virtual screening, HPC, GPU, molecular docking, SARS-CoV-2, COVID-19

1 Introduction

Drug discovery is a long process that usually involve *in-silico*, *in-vitro*, and *in-vivo* stages. The outcome of this process is a molecule, named *ligand*, that has the strongest interaction with at least one binding site of the protein, also known as *pocket*, that represents the target of the experiment. The inhibition of the target that can be caused by this interaction is expected to have a therapeutic effect. Virtual screening is one of the early stages that aims to select a set of promising *ligands* from a vast chemical library. The introduction of this step led to an increase in the success rate of the next stages of the drug discovery [1], [2], [3], [4], [5]. The complexity of this operation is due to the ligand and pocket flexibility: both of them can change shape when they interact. Therefore, to estimate the interaction strength using a *scoring function*, we also need to predict the displacement of their atoms using a *docking* algorithm. This problem is computationally heavy, and it is well known in literature [6]. Moreover, to increase the probability of finding promising candidates, we would like to increase the size of the chemical library as much as possible, exacerbating the complexity of the virtual screening. Since the evaluation is *in-silico*, we can design new molecules by simulating known chemical reactions. Therefore the chemical library size is limited only by the system's computational power.

Urgent computing is an area of computer science that investigate the usage of computation resources to predict and prevent critical situation [7]. In this context, where the time required to find a therapeutic cure should be as short as possible, we re-designed the EXSCALATE platform with the goal of virtual screening as many ligands as possible in a reasonable time budget, i.e. from months to hours for screening billions of ligands. To maximize the throughput of the docking platform, we target *High-Performance Computing (HPC)* supercomputers since their design maximizes the number of arithmetic operations per second, using double-precision floating-point numbers *FLOP/s*. Indeed, the TOP500 list [1] ranks all the HPC supercomputers worldwide according to their throughput. When we focus on the top ten supercomputers, we can notice that most of them have heterogeneous nodes that heavily rely on accelerators, typically *GPU*s. Thus, we need to hinge on the node's accelerators and efficiently scale up to the available nodes to use all the computation power of the target machine. Even if we focus on the software level, there are multiple well-known issues [8], [9]. The most representatives of them are how to efficiently use accelerators, how to transfer data within a node to feed the accelerators, how to move data from storage devices to the machine's node and vice versa, how to minimize communications between nodes and synchro-

- *D. Gadioli, E. Vitali, C. Silvano, G. Palermo are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano. E-mail: name.surname@polimi.it*
- *F. Ficarelli, C. Latini are with SuperComputing Applications and Innovation, CINECA, Casalecchio di Reno. E-mail: f.ficarelli@cineca.it, c.latini@cineca.it*
- *C. Manelfi, c. Talarico, A. R. Beccari are with EXSCALATE, Dompé farmaceutici S.p.A, Napoli. E-mail: name.surname@dompe.it*
- *C.Cavazzoni is with Leonardo S.p.A., Genova. E-mail: carlo.cavazzoni@leonardocompany.com*

1. TOP500 website - https://top500.org/

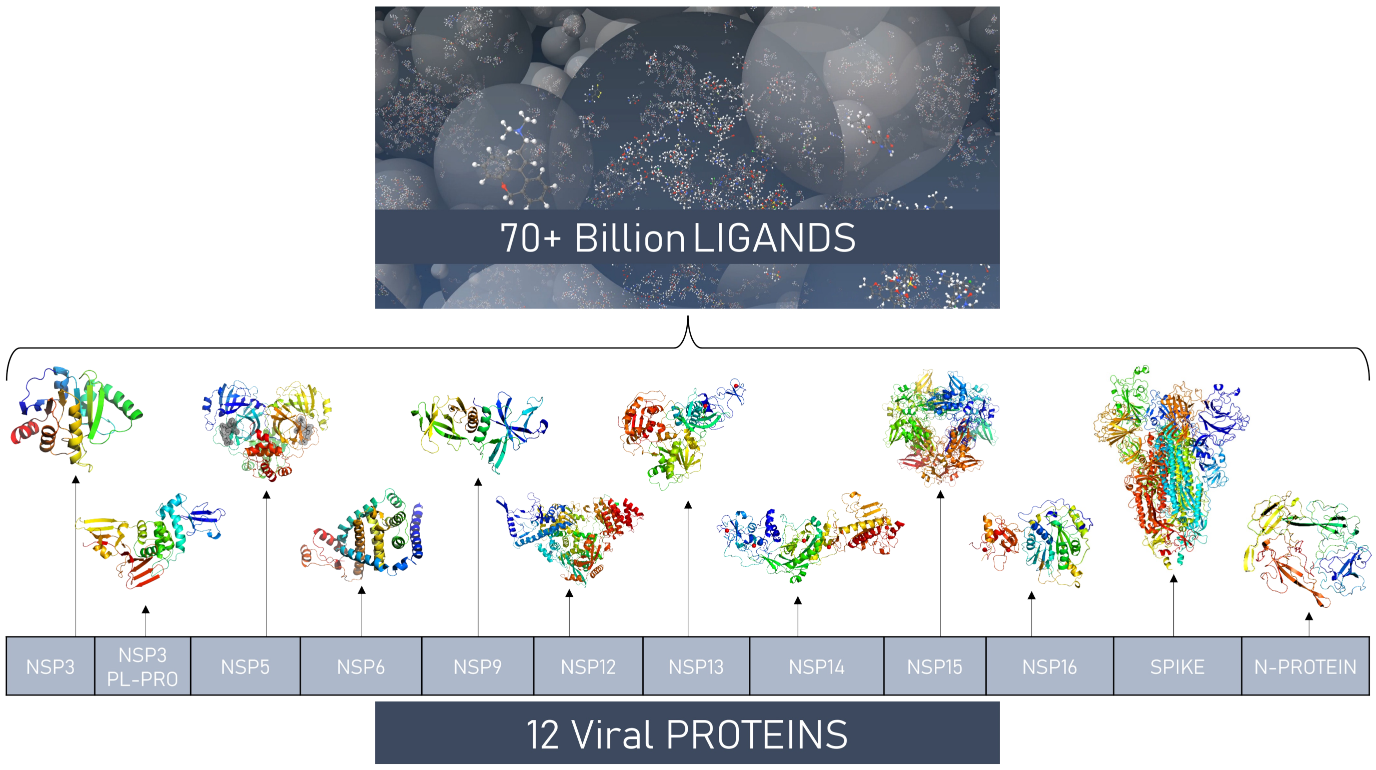


Fig. 1: Schematic representation of the dataset used for the EXSCALATE4CoV virtual screening experiment.

nizations between processes, and how to improve resilience to reduce the impact of faults in the time-to-solution.

The EXSCALATE4CoV European project[2] aims at finding new potential drugs against the COVID-19 pandemic. During this project, we deployed the EXSCALATE platform on two HPC machines (CINECA Marconi100 and ENI HPC5) with a combined throughput peak of 81 PFLOPS, to rank a chemical library of more than 70 billion ligands against 15 binding-sites of 12 viral proteins of SARS-CoV-2 (Figure 1). Overall, the experiment lasted 60 hours and it performed a trillion of docking operations, becoming the largest structure-based virtual screening experiment with 50x more small molecules and 7.5x more targets than the previous record [10]. The knowledge generated by this experiment, in terms of top-ranked molecules for each protein pocket (binding sites), is publicly available through MEDIATE [11] to share our findings with researchers interested in de-novo drug design against SARS-CoV-2. The produced data are suitable for a polypharmacology approach where we focus our attention on drugs interacting with multiple targets. Indeed, molecules hitting more than one target might be more adequate to achieve a therapeutic effect [12]. Filters on the molecules showing a too promiscuous behavior can be applied on the output data. In-silico toxicity analyses are not in the scope of the current use of the platform since the targets of the extreme-scale experiment are only the viral proteins. The EXSCALATE platform, and its polypharmacology support, has been already proved to be a useful tool against SARS-CoV-2 by suggesting the Raloxifene [4] as a potential clinical candidate while adopting a pure repurposing strategy on existing drugs.

2. Project website: https://www.exscalate4cov.eu

The remainder of the document is organized as follows: Section 2 briefly describes the most related applications that can perform a virtual screening. Section 3 describes the EXSCALATE platform, highlighting the innovation introduced and the design choices that led to the performance of the virtual screening campaign that we reported in Section 5. Section 4 reports the configuration of the machines used for the experiment, both from the hardware and software perspectives. Finally, Section 6 concludes the paper.

2 Related Works

Molecular docking applications can serve different purposes, from virtual screening to accurate simulations. For this reason, we have a wide spectrum of algorithms and approaches [6], [13], [14], [15] that covers the performance-accuracy curve trade-off. Since we use molecular docking to select the most promising candidates, we are interested in fast approaches such as ICM [16], PSOVina [17], or EUDOCK [18]. However, the majority of these works are designed for the day-by-day use case scenario, rather than urgent computing where we aim at using all the computation power of the system. For this reason, there are few approaches [19], [20], [21] that can use GPUs to improve their performance with respect to a CPU implementation. For example, PLANTS [19] reported a speedup of 60X when using GPUs to carry out the computation. Besides improving the time-to-solution, the usage of GPUs is required to access the majority of the computation power in modern HPC machines. Moreover, there are even fewer approaches that designed the I/O to efficiently handle more than a ligand and pocket pair, while providing the ability to use accelerators. AMIDE v2 [22] focuses on inverse docking, where ligands are evaluated in a large number of proteins. They propose to divide each protein into twelve overlapping sub-grids to use as independent pockets. To orchestrate the elaboration they use scripts and SLURM job arrays. METADOCK 2 [23] focuses on blind docking, where the docking phase is not restrained in a specific pocket, but it can be docked in the whole protein's surface. They propose to use a combined OpenMP/CUDA approach to use all the computation power of a node, targeting HPC machines.

AutoDock [24] is the most related work, since it has been ported in CUDA (AutoDock-GPU [25]) and deployed on the Summit supercomputer, where they docked over one billion molecules on two SARS-CoV-2 proteins in less than two days [10]. They hinge on the Summit's NVMe local storage to dock batches of ligands in the target pocket and to store the intermediate results. In particular, AutoDock-GPU uses OpenMP to implement a threaded-based pipeline, where each thread reads ligands from the file, launches the CUDA kernels, waits for their completion, and it writes back the results. Since most docking algorithms use a fast but approximated scoring function to drive the estimation of the 3D pose of a ligand, it is common to re-score the most promising ones with a more accurate scoring function. They use a custom CUDA version of RFScore-VS [26] to perform such task and BlazingSQL for computing statistics and selecting the top scoring ligands. To orchestrate the workflow they use FireWorks [27] from an external cluster to ensure a consistent state in presence of faults in the computations nodes.

3 EXSCALATE PLATFORM

The approach that we followed to re-design the EXSCALATE platform differs from the works described in Section 2 in several ways. We use a monolithic application to dock and re-score the ligands, using MPI [28] to scale out, *C++11* threads to scale up and CUDA kernels to accelerate the compute-intensive sections, exploiting the V100 GPU on the target HPC clusters. The proposed solution can reach a high throughput without relying on the node's local storage, which is not available in the target HPC systems. Moreover, we envelop the application in a more complex workflow that compensates for its weak points.

The innovation introduced in this work can be categorized into three main contributions:

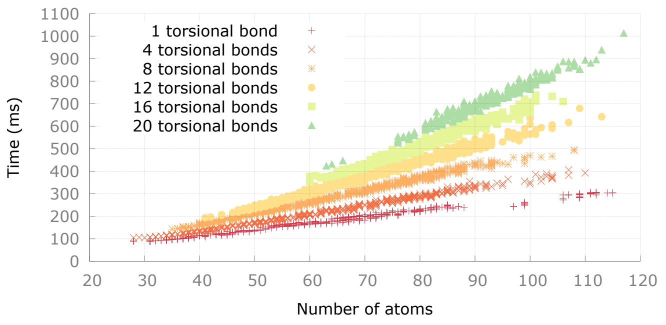
1) Algorithm level, with the CUDA porting and optimization of the docking and scoring phases.
2) Application level, with the complete rework of the application and the creation of the high-throughput molecular docking application.
3) Workflow level, with the creation of the EXSCALATE workflow that allows us to handle the operation in an easier and more resilient way.

3.1 The dock and score algorithm

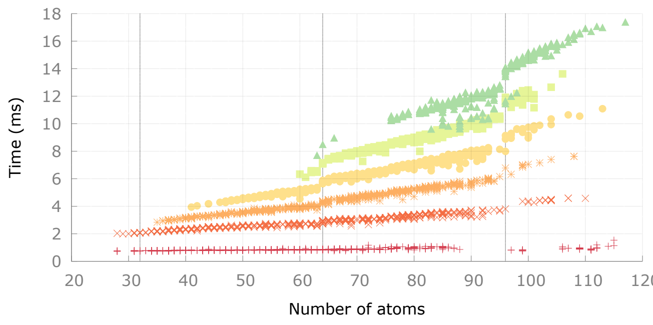
The final output of the algorithm is an estimation of the bond strength between a given ligand and the binding site of the target protein. In the virtual screening context, it is common to reduce the problem's complexity by using heuristics and empirical rules instead of performing a molecular dynamic simulation [29]. One implication of this choice is that the numeric score of a ligand is strongly correlated by the given 3D displacement of its atoms, which is not trivial to compute due to the high number of degrees of freedom involved in the operation. Besides the six degrees of freedom derived by rotating and translating a rigid object in a 3D space, we must consider the ligand flexibility. A subset of the ligand's bonds, named *torsional bonds* [30], partition the ligand's atoms in two disjoint sets, that can rotate along the bond's axis, changing the ligand's shape. A small molecule can have tens of torsional bonds. We further reduce the problem complexity by considering in the algorithms the protein as a rigid body [31]. It is then possible to consider different conformational states of the target protein by adopting ensemble docking strategies [32].

The entire process that we use in EXSCALATE to dock and score a ligand is composed of four steps. All the involved algorithms are deterministic from the functional point of view, which means that they are not based on any stochastic process to sample the conformational space. This feature is important because it permits a coherent replay of the docking phase and makes optional the storage of all the generated poses, drastically reducing the data transfer for large experiments. The first step is a ligand pre-processing that unfolds the ligand by rotating the torsional bonds to maximize the sum of the internal distances between all the molecule atoms. This computation is protein independent. The second step docks the ligand inside the binding site of the target protein by using a greedy gradient descent approach, with multiple restarts. The gradient is defined by a scoring function that considers only geometrical steric effects. Starting from different initial poses, the optimization algorithm initially considers the ligand as a rigid body, then in a later phase, it considers the internal flexibility with small perturbations. We take into account the ligand's flexibility, but we consider the pocket as a rigid body. In the experiment, we evaluated 256 different initial poses for each ligand. The third step sorts the generated poses to select only a few to re-score using the LiGen chemical scoring function [33] in the fourth step. In particular, we cluster the generated poses using a root mean square deviation (RMSD) of 3Å as the threshold to deem two poses as similar. Then we sort them to have in the first places the top-scoring pose for each cluster, then all the others; sorted according to the geometrical scoring function. In the experiment, we scored only the top 30 poses for each ligand. The score of the ligand is the score of the best pose that we found. The main idea is that we prefer to re-score poses far from each other, rather than just the ones that happen to have to highest geometrical score, to avoid generating the same score twice.

We implemented the algorithm in *C++17* to target CPUs, and we previously implemented an OpenACC implementation to target accelerators [34]. The target HPC machines are provided with NVIDIA GPUs, and it is known from previous benchmarking works [35], [36] that it is possible to gain significant speedup with a CUDA implementation over OpenACC on these GPUs. For this reason, we decided to rewrite using CUDA the docking and scoring kernels for this experiment.



(a) C++ implementation on CPU



(b) CUDA implementation on GPU

Fig. 2: Time required to dock and score a ligand by varying the number of atoms and torsional bonds. The C++ implementation use a single core IBM 8335-GTG 2.6 GHz. The CUDA implementation use a single NVIDIA V100. The vertical lines highlight when we use an addition warp to compute a ligand (e.g. after 64 atoms).

The algorithm's asymptotic complexity is $O(n \cdot m)$, where n is the number of atoms and m is the number of torsional bonds. We omit features of the target docking site since the docking site is constant during the docking application lifetime. Figure 2 measures the time required to dock and score a ligand on a Marconi100 node, by varying the implementation, the ligand's number of atoms, and torsional bonds. While the observed execution time of the C++ implementation (Figure 2a) increases linearly with the number of atoms and torsional bond, the CUDA implementation (Figure 2b) exhibits a less evident relationship with the number of atoms. This is due to the fact that we use hardware parallelism to perform the computation on the atoms, while we need to process the torsional bonds serially to preserve the molecule geometry. Moreover, since we organized the elaboration in bundles of 32 atoms (a CUDA warp size), we have a steep increase in the docking time when we need to process more atoms; for example after 64 and 96 atoms.

We can notice in both implementations how the docking time is heavily input dependent, where the difference between the fastest and slowest class of ligands is more than one order of magnitude. The maximum number of torsional bonds and atoms is a compile-time parameter. The current version is set to 156 atoms and 55 torsional bonds.

3.2 EXSCALATE high-throughput docking application

The only information required to dock and score a ligand in the target binding site is their description. Thus, the virtual screening process is an embarrassingly parallel problem. However, it is of paramount importance to design how the data can be read from the storage, transferred to the accelerator, and written back to the storage. Indeed, another innovation introduced with this work is the high-throughput docking application that aims at addressing all the issues that are not related to the docking and scoring kernels but are required for the experiment's success, such as data management, resources organization, and multi-node scaling.

Figure 3 shows an overview of the application abstraction and software stack of the EXSCALATE high-throughput docking application. We have chosen to write an MPI application that implements an asynchronous pipeline. In particular, we want to execute a single process for each node available. Then, each process spawns a pipeline to carry out the elaboration using all the computation resources of its node. We use a dedicated thread for each stage of the pipeline. Moreover, each stage may have a thread-safe queue that stores input data.

The first stage is the *reader*, which reads from the actual file that represents the chemical library a chunk of data that it enqueues in the *splitter*'s queue. The splitter stage inspects each chunk of data to separate all the ligands' descriptions that are contained in the chunk itself. Then it enqueues each ligand description in the *docker*'s queue. In the experiment, we describe a ligand using a custom binary format derived from the TRIPOS Mol2 [37] format, described in more detail in Section 5. The docker stage dequeues a ligand description, it constructs the related data structures, performs the dock and score steps described in Section 3.1, and it enqueues the ligand's score in the *writer*'s queue. The writer stage dequeues the ligand score and accumulates in an internal buffer the related output, which is the ligand's SMILES representation and its score value in a CSV-like fashion. When the accumulation buffer is full, the writer stage initiates the writing procedure.

The docker stage is the only one that can be composed of several threads that operate on the same queues, thus the workload is shared among all the workers of a single node. Moreover, it is possible to use different algorithm implementations, such as CUDA and C++, to leverage the node's heterogeneity. We refer to any docker thread as *worker*. All the workers that use the CUDA implementation are named *CUDA workers*, while the ones that use the C++ implementation are named *CPP workers*. Even if a single CUDA worker is tied to a single GPU, it is possible to have multiple CUDA workers tied to the same GPU.

We consider the target binding site constant during the elaboration. Therefore, each process will fetch the related information once at the beginning of the execution. Each algorithm implementation is free to store the pocket data structures in the most appropriate memory location during its initialization. In particular, the C++ implementation uses constant static memory, while the CUDA implementation uses texture memory.

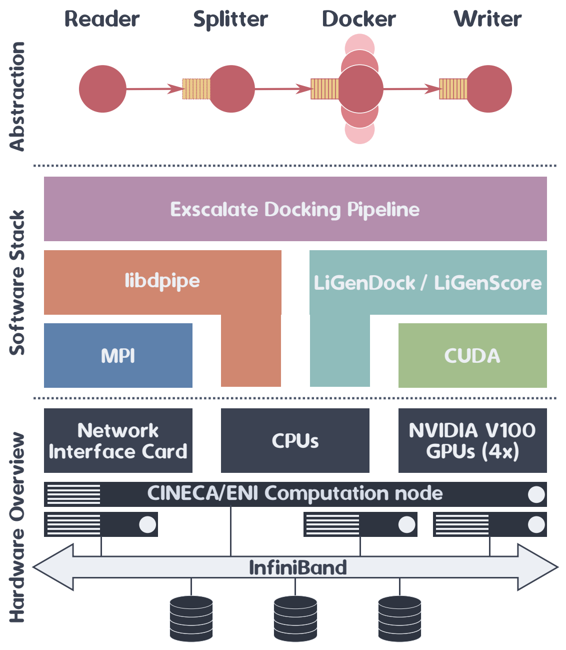


Fig. 3: Overview of the EXSCALATE docking application by varying the abstraction level.

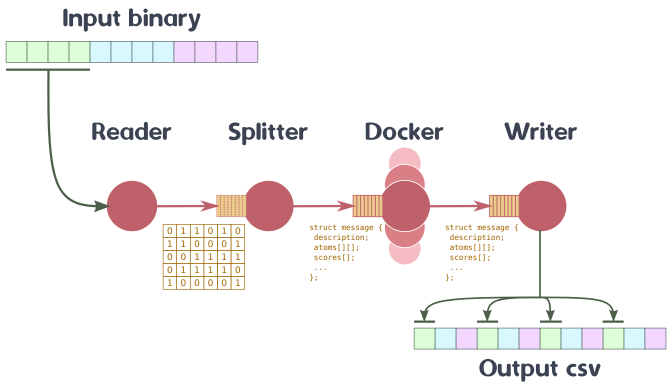


Fig. 4: Example of I/O synchronization with three Application instances, represented by different colors, that read the input ligands and store the results.

The EXSCALATE Docking Pipeline library contains all the application's stages implementations. To parallelize the computation it uses libdpipe, a custom library that implements high-level interfaces for MPI and *C++* threads. The LiGenDock/LiGenScore libraries implement the domain-specific functional concerns.

Even if the problem is embarrassing parallel, we need to synchronize all the application's instances using MPI when we perform I/O operations. Figure 4 depicts an example of I/O coordination with three MPI processes, represented by different colors. Since the computation pipeline is the same for all the processes, we depict only one MPI process. To distribute the computation workload among the MPI processes, we split the input file in even slabs according to the file size and the number of MPI processes. Since the size of a molecule description depends on the ligand's properties, such as the number of atoms, it rarely happens that a slab starts exactly with the beginning and stops at the

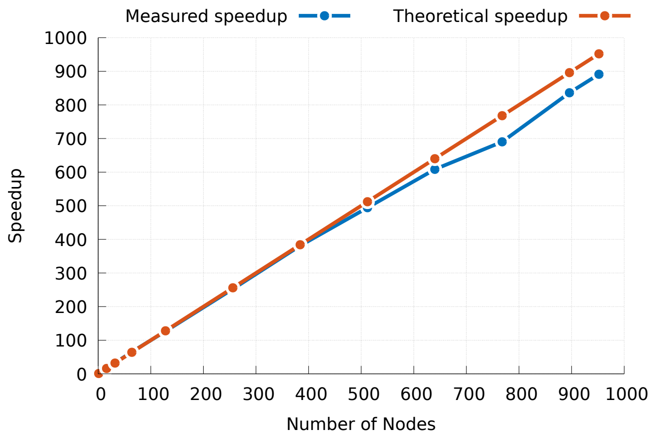


Fig. 5: Strong scaling experiment of the high-throughput molecular docking on the whole Marconi100 supercomputer.

ending of a molecule description. We use the convention that each task processes all the ligands whose description begins between the slab start and stop. The last ligand description may end after the slab stop.

On the main hand, we are using a very I/O-friendly access pattern because we read a file sequentially. On the other hand, the static data partition negates work-stealing among MPI processes. Therefore, the application throughput is equal to the throughput of the slowest process. However, Section 3.3 describes how we solved this issue using a preprocessing step. The frequency at which each process reads from the input file depends on the pipeline throughput.

The writer stage uses collective I/O operations to coalesce writing requests together before writing to the storage. Indeed, the user can configure the number of processes that issue I/O operations to reduce the pressure on the file system. Therefore, the synchronization among processes needs to concentrate the information that we want to write on the processes that issue the I/O request and they have to agree at which offset each MPI process starts to write. This access pattern is I/O-friendly because all the writing operations are parallel and sequential. Indeed, as we can see from Figure 5, the I/O does not represent a bottleneck and the scaling of the high-throughput molecular docking application is very close to the optimal theoretical scaling.

3.3 EXSCALATE workflow

As we have seen from the strong scaling experiment, it is possible to store all the ligands that we would like to dock in a single file and to deploy the docking application on the whole machine. However, this approach has several drawbacks. The main concern is fault resiliency. The default action to respond to a fault in an MPI communicator, for example after a node failure, is to terminate all the processes [28], which can lead to losing a significant amount of computation effort. This is a well-known problem in HPC [38], [39], [40]. Another concern lies in the application performance. Figure 2 shows how docking and scoring a large and complex ligand required much more time than a small and simple ligand. Therefore, we have a significant imbalance between the MPI processes if all the ligands with many atoms and torsional bonds are close.

For these reasons we propose the EXSCALATE workflow: this approach addresses these issues with a pre-processing phase on the chemical library to have a relatively small number of jobs that can run in parallel using a plain job array to coordinate the execution, such as the one provided by SLURM or PBS. The job array is in charge of controlling the execution of these jobs, and it is helped by custom reactive tools in identifying failing jobs, re-running them, and excluding failed nodes. To achieve this goal, we divide the amount of ligands (70 billions) into ∼3400 smaller sets. For every set, we create a job that runs on a subset of 32 nodes.

Figure 6 depicts the EXSCALATE workflow which requires two different kinds of input from the domain knowledge. On one hand, we require the binding sites of the target proteins. A possible procedure to derive the druggable cavities is reported in a previous work [41]. This procedure is also the one adopted for the extreme-scale experiment. On the other hand, we require the chemical library of molecules that we want to evaluate. It is possible to represent a molecule in a wide range of formats according to the amount of information that we want to store. In our case, we assume that the chemical library is stored using the SMILES format [42], which can encode a molecule in a single string that contains only the structure of the molecule (ignoring the hydrogen) since it is the most compact.

The next step in the ligand pre-processing is to broadly classify them in buckets according to their expected execution time, to reduce as much as possible the imbalance during the computation. As shown in Figure 2, the number of torsional bonds and atoms seem good predictors. However, it is not trivial to extract these properties from the SMILES representation. For this reason, we trained a model that predicts the execution time given properties that are more accessible at this point of the workflow: the number of heavy atoms, the number of rings, and the number of chains. We also consider interactions between them. We use a decision tree model with a maximum depth of 16 to predict the ligand's execution time.

Figure 7 shows the experimental campaign that we used to train a decision tree regressor [43] written in Python. Figure 7a shows the measured execution time of a dataset with 21 million of ligands with a different number of atoms and torsional bonds. We use the 80% of the data to train the model, while the remaining data are used to compute the prediction error reported in Figure 7b. The model has a negligible mean error (-0.00088 ms), with a standard deviation of 3.81 ms.

Even if the average error is almost zero, the standard deviation suggests that we cannot use the proposed model to predict the docking time of a single ligand. However, since this pre-process aims at avoiding imbalance in the computation while processing a large set of ligands, we are interested only in the average behavior. In the experiment, we cluster the ligands in buckets of 10 ms to further account for this variability.

After the ligands classification according to their complexity, we can perform the last pre-processing step. For each ligand we add the hydrogen atoms, we generate the initial displacement of its atoms in the 3D space, and we unfold the molecule (Section 3.1). This elaboration is required once and it can be re-reused in all the virtual screening campaigns.

Finally, once we have the target binding sites and the ligand binaries, we can perform the virtual screening campaign. The idea is to launch the docking application on all the ligand files, one pocket at a time. The output of the virtual screening is the ranking of the chemical library against each docking site. Even if the scoring function uses approximations that lead to a loss in accuracy, we can use these results to select a much smaller set of molecules to investigate more. For example, by using more accurate simulation techniques that also require a higher computation effort per ligand. When domain experts selected the ligands that have a strong interaction with multiple docking sites or proteins, it is possible to re-create the 3D displacement of the ligand's atoms on demand. For this reason, we can store only the structure of the molecule, using the SMILES notation.

4 Experimental Setup

This section reports the experimental setup for the virtual screening experiment where we evaluated 70 billion ligands against 15 binding sites of 12 viral proteins of SARS-CoV-2.

4.1 Target Dataset

The ligands that we evaluated in the experiment is part of the EXSCALATE-library owned by Dompé that was built starting from a database of millions of available commercial reagents that were combined using a set of robust synthetic reactions, in order to obtain a tangible chemical space, meaning that this is truly achievable in one reaction step. The steps to build the 3D structure of the molecule from SMILES can be summarized as follows: an initial parsing of the SMILES representation reconstructs the topological structure of the molecule: atoms, bonds, and their types, rings, and stereoisomerism. A force-field characterization, based on MMFF94 [44], is then assigned to each atom, permitting an initial guess of the coordinates based on known bond lengths and angles, constrained by the expected ring layout and the stereoisomers information from the parsing step. The missing hydrogens are added to the heavy atoms and the coordinates are assigned accordingly with the same strategy. As a final step, the energy minimization is performed via an iterative BFGS approach. Molecules not reaching convergence are filtered out. Further filtering is demanded for subsequent analyses: the focus of this work was scanning the widest number of candidates, to build up a set of promising compounds, to be later analyzed with increased precision methods. The list of target proteins used in the experiments has been reported in Table 1, with the corresponding PDB code. The crystal structures of the main functional units of the SARS-CoV-2 proteome were obtained from the Protein Data Bank[3]. Homology models of the proteins for which the crystal structure is not available were generated and used. The procedure to derive the homology models together with the identification of the druggable cavities are reported in a previous work [41].

3. Protein Data Bank website - https://www.rcsb.org/

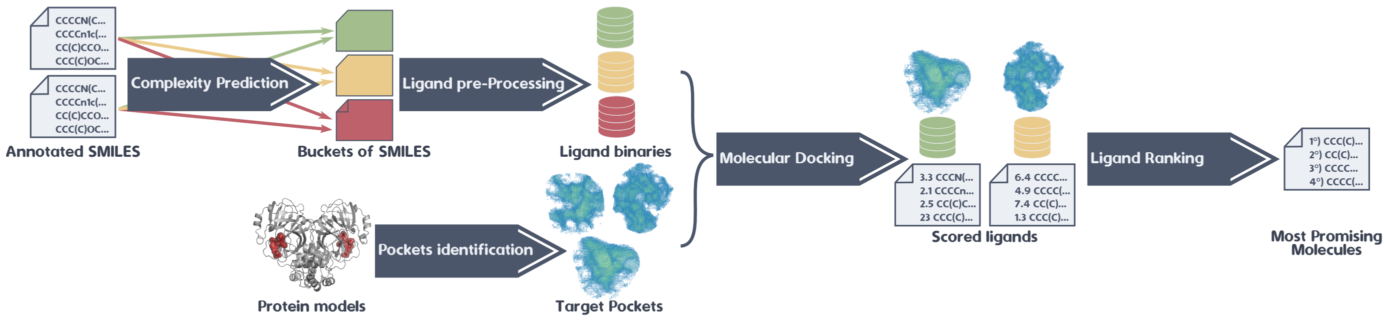
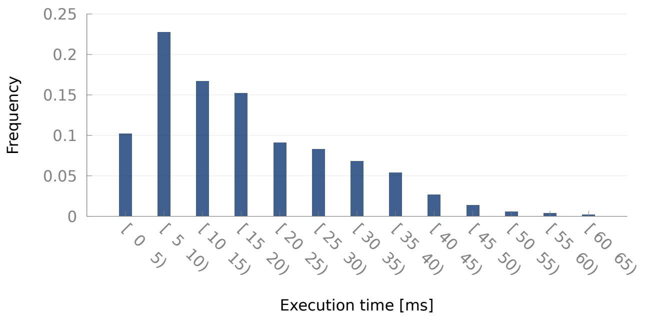
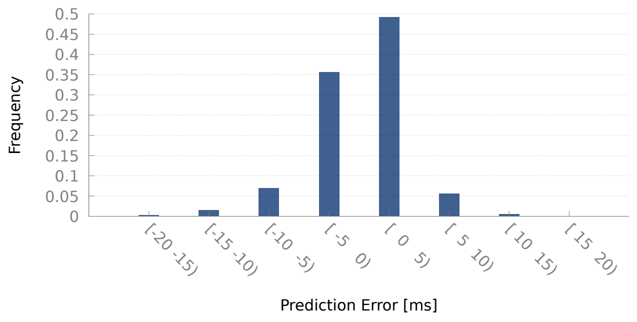


Fig. 6: EXSCALATE workflow, from the input (ligand's chemical library and the protein models) on the left; to the final outcome (most promising set of molecules) on the right.



(a) Measured docking time



(b) Prediction error

Fig. 7: Frequency distribution of the measured docking time, using the CUDA implementation, and its prediction error, defined as the difference between the observed execution time and its predicted vale. We discared values with a frequency lower than 0.001 for conciseness purposes.

TABLE 1: The 3D targets used in the molecular docking experiments. A target might have different pockets.

Protein	PDB code
3CL protease (NSP5)	6LU7
N-protein	6VYO
NSP3	6W02
NSP6	De novo model
NSP9	6W4B
NSP12	7BV2
NSP13	6XEZ
NSP14	Homology Model
NSP15	6W01
NSP16	6W4H
PL protease	6W9C
Spike-ACE2	6M0J

4.2 Hardware Environment

We deployed the EXSCALATE platform on Marconi100 at CINECA and HPC5 at ENI, aggregating around 81 petaflops of performance peak (respectively 29.3 and 51.7 petaflops). A Marconi100 node is equipped with 32 IBM POWER9 AC922 cores (128 hardware threads) and 4 NVIDIA V100 GPU, with NVLINK 2.0. The computation node of HPC5 is very similar since it also uses 4 NVIDIA V100 GPUs, but it relies on Intel Xeon Gold 6252 24C as CPU (24 cores and 48 hardware threads) and it uses NVLINK only for the GPU to GPU interconnection. The CPU to GPU connection uses PCI-Express. The experiment has been run using a reservation of 800 out of 970 CINECA-Marconi100 nodes, and 1500 out of 1820 ENI-HPC nodes for 60 hours in each machine.

4.3 Software Environment

All software components were built on top of the same software stack for both of the production systems: upstream GCC 9.3, CUDA toolkit 11.0 and upstream MPICH 3.4.1. The main difference between the two systems was in the job scheduler: SLURM on Marconi100 and PBS on HPC5. On the former, the single 32-nodes jobs were managed in multiple job arrays, each one covering the whole set of docking targets, while on the latter an ENI's internal, proprietary workflow management tool (Beat) was used to schedule single jobs and deal with transient faults.

For the post-processing phase, a custom Dask pipeline dealing with statistical descriptors and threshold selection has been developed and ran on an environment deployed using upstream conda-forge (Dask 2.21.0 on Python 3.8.3). The same Python environment was used for the pre-processing phase as well, where a custom regression model was trained, serialized and deployed using scikit-learn 0.22.1.

4.4 Performance Measurements

In the virtual screening context, the most important metric to measure the application performance is throughput. Since we run a single MPI process on each node, we can measure the node's throughput using *std::chrono::steady_clock* facilities. Each time that we need to compute the throughput, we divide the number of ligands that the application has processed by the elapsed time. We log this information in the application output during the evolution of the elaboration. To compute the average node's throughput, we compute the average value among the final throughput logged by the applications.

To measure the machine throughput, we divide the total number of ligands by the wall time of the computation, i.e. the time required to complete the job array. In this way, the measure includes all the overheads related to the execution. Since a pocket elaboration lasts for hours, the accuracy of the measure is compatible with the method used.

5 Performance Results

This section collects the experiment's extra-functional properties in terms of storage and throughput.

5.1 Evaluating the storage requirements

One of the main concerns in HPC systems is storage. When scaling an experiment to the scale of a trillion docking operation, it requires us to evaluate in detail what we want to read and write, giving attention to the format. To perform the virtual screening, we need information about the binding sites of the target proteins and the chemical library of ligands that we want to analyze. The former is not an issue since it requires total storage of 29MB and the information needed is read once when the application starts. The latter needs more careful consideration.

Domain experts use to work with SMILES format to represent a ligand since it is the most compact. The chemical library evaluated in the experiment encoded in the SMILES format requires a total of 3.3TB. However, the docking application requires a richer description of the molecule, as described in Section 3.3. The most widely used format to store the required information is the TRIPOS Mol2 [37], which is encoded in ASCII characters and focuses on readability rather than efficiency. For this reason, we use a custom binary format that stores only the information required by the docking application, such as the atom's position, type, and bonds. By comparing the size of the same molecules, the Mol2 format requires $5-6$X more space with respect to the binary format. Nonetheless, the whole binary chemical library for the experiment requires 59TB of storage.

Storing all the docked poses is unfeasible since we need to store the whole molecule description for each pose that we want to consider. Since we are targeting 15 binding site and we re-score 30 alternative poses for each input ligand because it would require 26PB of storage, which exceeds the combined capacity of the target HPC machines. For this reason, we store only the SMILES of the molecule and its best score in a CSV-like file. Then, we can re-generate the docked pose on demand since the docking algorithm is deterministic. The size of the final output is 69TB of data.

On average, the docking application requires a relatively small I/O bandwidth: 1.68 GB/s for reading and 0.12 GB/s for writing on the Marconi100 machine, while 2.53 GB/s for reading and 0.18 GB/s for writing on the HPC5 machine. Despite this result, we needed to carefully tune the software knobs related to the I/O (Section 3.2) to avoid scaling issues on large systems [45].

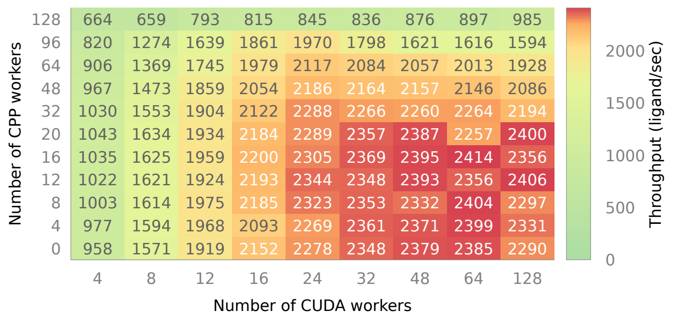


Fig. 8: Throguhput of the docking application in terms of ligands per seconds, by varying the number of CPP and CUDA workers.

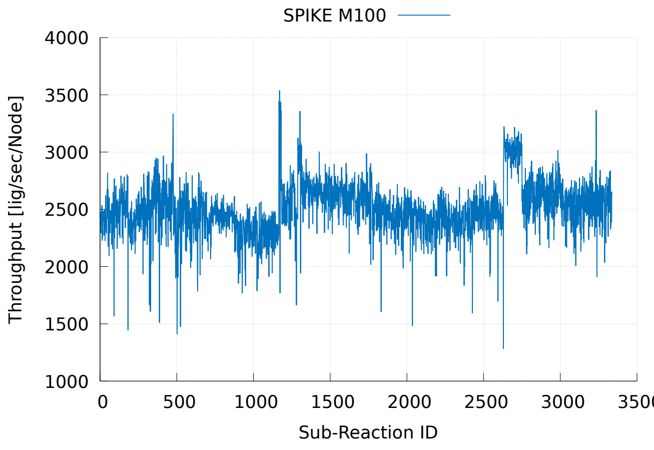
5.2 Exploiting a node heterogeneity

The availability of multiple implementations of the dock and score algorithm grants access to heterogeneous resources. Figure 2 shows the time elapsed by the CPP and CUDA workers to perform the docking operation with different ligand characteristics. We can notice that the CUDA implementation has, on average, a 64x speedup with respect to the CPU version Therefore using only the CUDA implementation seems to be the most efficient solution. However, the relationship between the number of CUDA and CPP workers (Section 3.2) and the application throughput is not trivial. Figure 8 shows the application throughput in terms of docked ligands per second, by varying the number of CUDA and CPP workers, when we deploy the application on a Marconi100 node, which has 32 IBM POWER9 AC922 cores (128 hardware threads) and 4 NVIDIA V100 GPU. The application binds each CUDA worker to a single GPU in a round-robin fashion. For example, when we use 24 CUDA workers, we have 6 threads that feed data and retrieve the results for each GPU in the node.

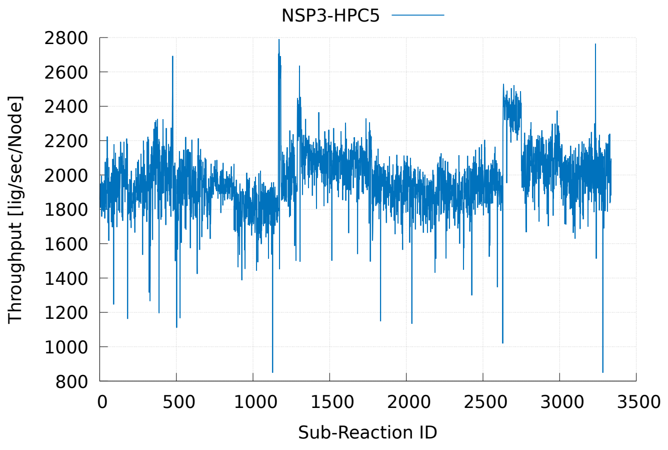
From the throughput, we can notice how the application reaches the peak performance for a high number of CUDA workers. Moreover, when we increase the number of CPP workers to match the number of hardware threads, we harm the application performance. This behavior implies that, in our case study, it is better to use CPUs to support accelerators and I/O operations rather than contribute to the elaboration itself. Furthermore, to benefit most from a GPU it is not enough to use a single CUDA worker. We expect this result because the CUDA worker needs to parse the ligand description and initialize the related data structures before launching any CUDA kernel. Thus, by using more CUDA workers we can hide these overheads and fully utilize the GPU. To perform this analysis we used the dataset *"Commercial Compound MW<330"* from the MEDIATE website [11]. It is composed of 5 million of small molecules and it is publicly available.

5.3 Scaling on the target HPC machine

We used the EXSCALATE workflow described in Section 3.3 to overcome the limitations of using a single MPI application that runs on the whole supercomputer. For this experiment, we decided to evaluate the binding sites sequentially.



(a) SpikeACE on Marconi100



(b) NSP3 on HPC5

Fig. 9: Execution track of two entire job arrays targeting two different protein pockets on the two different supercomputers.

TABLE 2: The throughput reached per node and per machine for each binding site evaluated in the experiment. *The NSP13ortho binding site has been partially computed in both machines.

Binding site	Throughput (ligands/sec/node)	Throughput (ligands/sec)	HPC machine
PLPRO	2496	1996800	M100
SPIKEACE	2498	1998400	M100
NSP12thumb	2499	1999200	M100
NSP13palm	2486	1988800	M100
3CL	2427	1941600	M100
NSP13allo	2498	1998400	M100
Nprot	2010	3015000	HPC5
NSP16	1980	2970000	HPC5
NSP3	1969	2953500	HPC5
NSP6	1985	2977500	HPC5
NSP12ortho	2001	3001500	HPC5
NSP14	1965	2947500	HPC5
NSP9	1996	2994000	HPC5
NSP15	1990	2985000	HPC5
NSP13ortho*	2454	1963200	M100
NSP13ortho*	1987	2980500	HPC5

With this configuration, for every binding site, we have a job array of ~3400 jobs, where each job is composed of 32 MPI processes that last for about 5 minutes and targets a single binding site.

Table 2 reports for each binding site the average throughput of a node and the whole machine. On average, the throughput of a single node is $2.4k$ ligands per second on Marconi100 nodes, while is a little lower, $2k$ ligands per second, on HPC5 nodes. Both supercomputer nodes are equipped with 4 NVIDIA V100 GPU, and we have seen that most of the throughput of the application is given by the CUDA accelerated kernels. For this reason, this difference in performance by the two nodes is unexpected. However, there is a big difference in the architectures of the Marconi100 node and the HPC5 node, that is how the GPUs and CPUs are connected: Marconi100 has NVLINK2.0, while HPC5 relies on PCI-Express. In our case study, NVLINK is better at transferring the ligands' data.

If we only consider the throughput of a Marconi100 node, we can notice that the throughput that we measured while scaling to the whole machine is comparable to the one that we measured while tuning the number of CUDA and CPP workers, reported in Figure 8. Therefore, the EXSCALATE platform was able to exploit all the available resources, reaching a combined throughput of $5M$ ligands per second on both supercomputers.

Finally, Figure 9 shows the execution track of the job arrays on two different proteins running on the two supercomputers. Also in this case it is visible that the difference in performance is not due to the target protein but mainly due to the different node architecture. Despite the performance are almost stable across the workload, the difference in throughput between the jobs is due the average complexity (i.e. number of atoms and rotatable bonds) of the ligands included in the sub-reactions associated to the jobs. This can be also noticed by the similar profile of the plots on the two different machines. These results show how the throughput is strongly influenced by the input data.

5.4 Data pre/post-processing

The main challenge of the experiment is to generate the chemical knowledge of the virtual screening. However, it requires a pre-processing phase to prepare the ligands: this phase is described in detail in Section 3.3 and it must be performed only once as the same pre-processed chemical space is evaluated against all docking sites.

The output of the experiment is a list of files ranking the ligands according to the strength of their interaction with the target protein in a CSV-like file. Even if the output can be used as-is, we chose for the sake of convenience to perform a preliminary post-processing step to join all the scores for the same ligand across all docking sites thus obtaining a single global table for the whole experiment.

The actual post-processing phase involves several steps aimed at obtaining statistical descriptors for the full score's distributions (mean, median, standard deviation, several percentiles); these descriptors are then used to extract the best-scoring compounds for each docking site to form the final released dataset. The computation has been carried out using a Dask distributed pipeline on the Marconi100 system. To identify the best prospect molecules we took

TABLE 3: Time and resources required to complete the experiment's phases.

Phase	Time	Resources
Pre-processing	5 days	100 M100 nodes (no GPUs)
Dock & Score	60 hours	800 M100 + 1500 HPC5 nodes
Post-processing	5 days	19 M100 nodes (no GPUs)

into account, for each docking site, all the compounds that have been scored higher than 3 standard deviations from the distribution's mean. The resulting data set, containing more than 570 million top-scoring compounds, is freely available upon registration.

Table 3 summarizes the computational resources involved in each of the phases above.

6 Conclusions

In the context of urgent computing, where we want to reduce the social and economic impact of a pandemic as much as possible, we re-designed the EXSCALATE molecular docking platform targeting HPC systems to make possible an extreme-scale virtual screening campaign in a reasonable time. We virtual screened more than 70 billion of ligands against 15 binding sites of 12 viral proteins of SARS-CoV-2. The raw results account for 69TB of data that describes how the chemical library interacts with the targets. The set of most promising compounds filtered for each target has been made available on the MEDIATE portal [11] to permit researchers to start more detailed de-novo campaigns from a reduced set of compounds.

In this work, we outlined how we were able to scale to two full HPC systems, Marconi100 at Cineca and HPC5 at ENI S.p.A., at the time of the experiment the two most powerful supercomputers in Europe, to run a *one-trillion-docking experiment*[4] in 60 continuous hours of production. The GPU porting of the entire code for the docking and scoring part of the workflow has been the key factor in reaching the target throughput goal since we were dealing with Top500, heterogeneous, GPU-accelerated HPC installations. Another valuable outcome of this work is the availability of an end-to-end high throughput virtual screening pipeline, capable of evaluating trillions of compounds against tens of viral targets by improving time-to-solution by means of extreme scalability up to full Top500 sites. We tested the real-world use of the pipeline on multiple HPC sites that were capable of supporting us by providing procedures to deal with our urgent computing needs that required severe reduction or even complete stoppage of regular production.

Finally, the approach we presented has the potential to provide a tool for robust in-silico analysis across multiple targets. In the long term, we believe that the EXSCALATE platform can support de-novo drug design challenges moving the problems from the generation of the data (i.e. binding affinity calculation) to the use of them (i.e. hit identification). Despite the main driver for the enhancement of the EXSCALATE platform and for the experiment has been the COVID-19 pandemic, we believe that the value generated by our effort goes beyond it, having now a ready to use platform not only for the next pandemic but also for other targets, such as rare disease or antibiotic-resistant pathogens.

4. Experiment website - https://1trilliondock.exscalate4cov.eu/

References

[1] A. Corona, K. Wycisk, C. Talarico, C. Manelfi, J. Milia, R. Cannalire, F. Esposito, P. Gribbon, A. Zaliani, D. Iaconis, A. R. Beccari, V. Summa, M. Nowotny, and E. Tramontano, "Natural compounds inhibit sars-cov-2 nsp13 unwinding and atpase enzyme activities," *ACS Pharmacology & Translational Science*, vol. 5, no. 4, pp. 226–239, 2022.

[2] H. Matter and C. Sotriffer, *Applications and Success Stories in Virtual Screening*. John Wiley & Sons, Ltd, 2011, ch. 12, pp. 319–358.

[3] O. M. Becker, D. S. Dhanoa, Y. Marantz, D. Chen, S. Shacham, S. Cheruku, A. Heifetz, P. Mohanty, M. Fichman, A. Sharadendu, R. Nudelman, M. Kauffman, and S. Noiman, "An integrated in silico 3d model-driven discovery of a novel, potent, and selective amidosulfonamide 5-ht1a agonist (prx-00023) for the treatment of anxiety and depression," *Journal of Medicinal Chemistry*, vol. 49, no. 11, pp. 3116–3135, 2006.

[4] M. Allegretti, M. C. Cesta, M. Zippoli, A. Beccari, C. Talarico, F. Mantelli, E. M. Bucci, L. Scorzolini, and E. Nicastri, "Repurposing the estrogen receptor modulator raloxifene to treat sars-cov-2 infection," *Cell Death & Differentiation*, vol. 29, pp. 156–166, 2022.

[5] J. Singh, C. E. Chuaqui, P. Boriack-Sjodin, W.-C. Lee, T. Pontz, M. J. Corbley, H.-K. Cheung, R. M. Arduini, J. N. Mead, M. N. Newman, J. L. Papadatos, S. Bowes, S. Josiah, and L. E. Ling, "Successful shape-based virtual screening: The discovery of a potent inhibitor of the type i tgfβ receptor kinase (tβri)," *Bioorganic & Medicinal Chemistry Letters*, vol. 13, no. 24, pp. 4355–4359, 2003.

[6] N. S. Pagadala, K. Syed, and J. Tuszynski, "Software for molecular docking: a review," *Biophysical Reviews*, vol. 9, no. 2, pp. 91–102, 2017.

[7] N. López, L. D. Debbio, M. Baaden, M. Praprotnik, L. Grigori, C. Simões, S. Bogaerts, F. Berberich, T. Lippert, J. Ignatius, P. Lavocat, O. Pineda, M. G. Giuffreda, S. Girona, D. Kranzlmüller, M. M. Resch, G. Scipione, and T. Schulthess, "Lessons learned from urgent computing in europe: Tackling the covid-19 pandemic," *Proceedings of the National Academy of Sciences*, vol. 118, no. 46, p. e2024891118, 2021.

[8] S. Ashby, P. Beckman, J. Chen, P. Colella, B. Collins, D. Crawford, J. Dongarra, D. Kothe, R. Lusk, P. Messina, T. Mezzacappa, P. Moin, M. Norman, R. Rosner, V. Sarkar, A. Siegel, F. Streitz, A. White, and M. Wright, "The opportunities and challenges of exascale computing," *Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee*, pp. 1–72, 2010.

[9] R. Thakur, P. Balaji, D. Buntinas, D. Goodell, W. Gropp, T. Hoefler, S. Kumar, E. Lusk, and J. L. Träff, "Mpi at exascale," *Procceedings of SciDAC*, vol. 2, pp. 14–35, 2010.

[10] J. Glaser, J. V. Vermaas, D. M. Rogers, J. Larkin, S. LeGrand, S. Boehm, M. B. Baker, A. Scheinberg, A. F. Tillack, M. Thavappiragasam, A. Sedova, and O. Hernandez, "High-throughput virtual laboratory for drug discovery using massive datasets," *The International Journal of High Performance Computing Applications*, vol. 35, no. 5, pp. 452–468, 2021.

[11] Exscalate4CoV, "MEDIATE - molecular docking at home," https://mediate.exscalate4cov.eu/, Accessed: 2021-09-17.

[12] A. Lavecchia and C. Cerchia, "In silico methods to address polypharmacology: current status, applications and future perspectives," *Drug Discovery Today*, vol. 21, no. 2, pp. 288–298, 2016.

[13] N. A. Murugan, A. Podobas, D. Gadioli, E. Vitali, G. Palermo, and S. Markidis, "A review on parallel virtual screening softwares for high-performance computers," *Pharmaceuticals*, vol. 15, no. 1, p. 63, 2022.

[14] J. Biesiada, A. Porollo, P. Velayutham, M. Kouril, and J. Meller, "Survey of public domain software for docking simulations and virtual screening," *Human Genomics*, vol. 5, no. 5, pp. 497–505, 2011.

[15] E. Yuriev, J. Holien, and P. A. Ramsland, "Improvements, trends, and new ideas in molecular docking: 2012-2013 in review," *Journal of Molecular Recognition*, vol. 28, no. 10, pp. 581–604, 2015.

[16] M. A. Neves, M. Totrov, and R. Abagyan, "Docking and scoring with icm: the benchmarking results and strategies for improvement," *Journal of computer-aided molecular design*, vol. 26, no. 6, pp. 675–686, 2012.

[17] M. C. Ng, S. Fong, and S. W. Siu, “PSOVina: The hybrid particle swarm optimization algorithm for protein-ligand docking,” *Journal of Bioinformatics and Computational Biology*, vol. 13, no. 3, p. 1541007, 2015.

[18] Y.-P. Pang, T. J. Mullins, B. A. Swartz, J. S. McAllister, B. E. Smith, C. J. Archer, R. G. Musselman, A. E. Peters, B. P. Wallenfelt, and K. W. Pinnow, “Eudoc on the ibm blue gene/l system: Accelerating the transfer of drug discoveries from laboratory to patient,” *IBM Journal of Research and Development*, vol. 52, no. 1.2, pp. 69–81, 2008.

[19] O. Korb, T. Stützle, and T. E. Exner, “Accelerating molecular docking calculations using graphics processing units,” *Journal of Chemical Information and Modeling*, vol. 51, no. 4, pp. 865–876, 2011.

[20] Y. Fang, Y. Ding, W. P. Feinstein, D. M. Koppelman, J. Moreno, M. Jarrell, J. Ramanujam, and M. Brylinski, “Geauxdock: accelerating structure-based virtual screening with heterogeneous computing,” *PloS one*, vol. 11, no. 7, p. e0158898, 2016.

[21] I. Sánchez-Linares, H. Pérez-Sánchez, J. M. Cecilia, and J. M. García, “High-Throughput parallel blind Virtual Screening using BINDSURF,” *BMC Bioinformatics*, vol. 13, no. SUPPL 14, 2012.

[22] P. Darme, M. Dauchez, A. Renard, L. Voutquenne-Nazabadioko, D. Aubert, S. Escotte-Binet, J.-H. Renault, I. Villena, L.-A. Steffenel, and S. Baud, “Amide v2: High-throughput screening based on autodock-gpu and improved workflow leading to better performance and reliability,” *International journal of molecular sciences*, vol. 22, no. 14, p. 7489, 2021.

[23] B. Imbernón, A. Serrano, A. Bueno-Crespo, J. L. Abellán, H. Pérez-Sánchez, and J. M. Cecilia, “METADOCK 2: a high-throughput parallel metaheuristic scheme for molecular docking,” *Bioinformatics (Oxford, England)*, vol. 37, no. 11, pp. 1515–1520, 2021.

[24] G. M. Morris, R. Huey, W. Lindstrom, M. F. Sanner, R. K. Belew, D. S. Goodsell, and A. J. Olson, “AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility,” *Journal of Computational Chemistry*, vol. 30, no. 16, pp. 2785–2791, dec 2009.

[25] S. LeGrand, A. Scheinberg, A. F. Tillack, M. Thavappiragasam, J. V. Vermaas, R. Agarwal, J. Larkin, D. Poole, D. Santos-Martins, L. Solis-Vasquez, A. Koch, S. Forli, O. Hernandez, J. C. Smith, and A. Sedova, “Gpu-accelerated drug discovery with docking on the summit supercomputer: Porting, optimization, and application to covid-19 research,” in *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. New York, NY, USA: Association for Computing Machinery, 2020.

[26] M. Wójcikowski, P. J. Ballester, and P. Siedlecki, “Performance of machine-learning scoring functions in structure-based virtual screening,” *Scientific Reports*, vol. 7, no. 46710, pp. 1–10, 2017.

[27] A. Jain, S. P. Ong, W. Chen, B. Medasani, X. Qu, M. Kocher, M. Brafman, G. Petretto, G.-M. Rignanese, G. Hautier, D. Gunter, and K. A. Persson, “Fireworks: a dynamic workflow system designed for high-throughput applications,” *Concurrency and Computation: Practice and Experience*, vol. 27, no. 17, pp. 5037–5059, 2015.

[28] M. P. I. Forum, *MPI: A Message-passing Interface Standard, Version 3.1 ; June 4, 2015*. High-Performance Computing Center Stuttgart, University of Stuttgart, 2015.

[29] T. Cheng, Q. Li, Z. Zhou, Y. Wang, and S. H. Bryant, “Structure-based virtual screening for drug discovery: a problem-centric review,” *The AAPS journal*, vol. 14, no. 1, pp. 133–141, 2012.

[30] D. F. Veber, S. R. Johnson, H.-Y. Cheng, B. R. Smith, K. W. Ward, and K. D. Kopple, “Molecular properties that influence the oral bioavailability of drug candidates,” *Journal of medicinal chemistry*, vol. 45, no. 12, pp. 2615–2623, 2002.

[31] N. S. Pagadala, K. Syed, and J. Tuszynski, “Software for molecular docking: a review,” *Biophysical reviews*, vol. 9, no. 2, pp. 91–102, 2017.

[32] R. E. Amaro, J. Baudry, J. Chodera, Ö. Demir, J. A. McCammon, Y. Miao, and J. C. Smith, “Ensemble docking in drug discovery,” *Biophysical Journal*, vol. 114, no. 10, pp. 2271–2278, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0006349518303242

[33] C. Beato, A. Beccari, C. Cavazzoni, S. Lorenzi, and G. Costantino, “Use of experimental design to optimize docking performance: the case of ligendock, the docking module of ligen, a new de novo design program.” *Journal of Chemical Information and Modeling*, vol. 53, no. 6, pp. 1503–1517, 2013.

[34] E. Vitali, D. Gadioli, G. Palermo, A. Beccari, C. Cavazzoni, and C. Silvano, “Exploiting openmp and openacc to accelerate a geometric approach to molecular docking in heterogeneous hpc nodes,” *The Journal of Supercomputing*, vol. 75, no. 7, pp. 3374–3396, 2019.

[35] G. A. de Araujo, D. Griebler, M. Danelutto, and L. G. Fernandes, “Efficient nas parallel benchmark kernels with cuda,” in *2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. IEEE, 2020, pp. 9–16.

[36] S. R. M. Rostami and M. Ghaffari-Miab, “Finite difference generated transient potentials of open-layered media by parallel computing using openmp, mpi, openacc, and cuda,” *IEEE Transactions on Antennas and Propagation*, vol. 67, no. 10, pp. 6541–6550, 2019.

[37] TRIPOS-International, “SYBYL 7.1 - Mol2 file format,” 2005.

[38] M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson, A. A. Chien, P. Coteus, N. A. DeBardeleben, P. C. Diniz, C. Engelmann, M. Erez, S. Fazzari, A. Geist, R. Gupta, F. Johnson, S. Krishnamoorthy, S. Leyffer, D. Liberty, S. Mitra, T. Munson, R. Schreiber, J. Stearley, and E. V. Hensbergen, “Addressing failures in exascale computing,” *The International Journal of High Performance Computing Applications*, vol. 28, no. 2, pp. 129–173, 2014.

[39] W. Bland, A. Bouteiller, T. Herault, G. Bosilca, and J. Dongarra, “Post-failure recovery of mpi communication capability: Design and rationale,” *The International Journal of High Performance Computing Applications*, vol. 27, no. 3, pp. 244–254, 2013.

[40] R. Rocco, D. Gadioli, and G. Palermo, “Legio: fault resiliency for embarrassingly parallel mpi applications,” *The Journal of Supercomputing*, vol. 78, no. 2, pp. 2175–2195, 2022.

[41] S. Gervasoni, G. Vistoli, C. Talarico, C. Manelfi, A. R. Beccari, G. Studer, G. Tauriello, A. M. Waterhouse, T. Schwede, and A. Pedretti, “A comprehensive mapping of the druggable cavities within the sars-cov-2 therapeutically relevant proteins by combining pocket and docking searches as implemented in pockets 2.0,” *International Journal of Molecular Sciences*, vol. 21(14), no. 5152, 2020.

[42] D. Weininger, “Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules,” *Journal of Chemical Information and Computer Sciences*, vol. 28, no. 1, pp. 31–36, 1988.

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[44] T. A. Halgren, “Merck molecular force field. i. basis, form, scope, parameterization, and performance of mmff94,” *Journal of computational chemistry*, vol. 17, no. 5-6, pp. 490–519, 1996.

[45] S. Markidis, D. Gadioli, E. Vitali, and G. Palermo, “Understanding the i/o impact on the performance of high-throughput molecular docking,” in *2021 IEEE/ACM Sixth International Parallel Data Systems Workshop (PDSW)*. IEEE Computer Society, 2021, pp. 9–14.

Davide Gadioli received his his Master of Science degree in Computer Engineering in 2013, while in 2019 he received the Ph.D degree in Computer Engineering, from Politecnico di Milano (Italy). Currently, he is a postdoc at Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) of Politecnico di Milano. In 2015, he was a Visiting Student at IBM Research (The Netherlands). His main research interests are in application autotuning, autonomic computing and approximate computing.

Emanuele Vitali graduated in 2015 from Politecnico di Milano (Italy) after completing his Master of Science in Computer Engineering, and in 2021 he received the PhD degree from the same university. Currently, he is a postdoc at Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) of Politecnico di Milano. In 2019, he has been Visiting Student at Dividiti (UK). His main research interests include GPGPU architectures and programming, application autotuning and high throughput molecular docking.

Federico Ficarelli graduated in Computer Science at the University of Milan in 2008. He is currently a senior HPC software engineer in the High Performance Computing dept. at Cineca where he leads several co-design and development activities in the industrial R&D team. He is involved in several research projects as responsible for application and hardware-software codesign activities focusing on novel computing architectures, programming paradigms for heterogeneous platforms and compiler technologies.

Chiara Latini received the M.Sc. degree with honors in particle Physics at the University of Bologna. She earned a PhD in Electrical Engineering at the University of Bologna with the thesis titled ”Numerical strategies for the solution of the magneto-fuid-dynamic problem at Low Magnetic Reynolds Numbers”. Since 2008, when she joined the HPC department at CINECA, she has been working as a developer for industrial and scientific applications.

Candida Manelfi graduated in Chemistry in 2006 at the Rome University ”La Sapienza”. Since 2015 she is computational chemistry researcher at Dompé Farmaceutici, supporting the early drug discovery projects and providing Chemoinformatics and Bioinformatics services as compound library design and HTS data analysis. She is also part of the Joint Bioinformatics Group (JBG) at the Institute of Protein Biochemistry of CNR in Naples. Previously, she was applications specialist at S-IN Soluzioni Informatiche providing computer-assisted solutions in chemistry-related frameworks, and computational chemistry researcher at the Angelini Research Center. She participated to the H2020-ANTAREX project and she is co-PI to the PRACE granted project called Antarex for Zika.

Carmine Talarico graduated in Pharmacy at the University ”Magna Graecia” of Catanzaro in 2014 and got a PhD in Life Sciences and Technologies (curriculum Pharmaceutical Sciences) at the same university in 2017. He was awarded in 2018 with Award of excellence Paul Ehrlich MedChem Euro-PhD Network referring to his PhD thesis. From 2017 he is author and coauthor of a book chapter and eleven scientific papers published in international journals. He works as Computational Chemist at Dompé from 2018, and he is a member of the EXSCALATE group. He has been Project Leader of the PRACE granted project called Antarex for Zika.

Cristina Silvano is a Full Professor at the Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano. Her main research interests are in energy-efficient embedded systems, manycore architectures and application autotuning. She has published more that 160 scientific papers in peer-review journals and conferences, and she holds several patents in collaboration with Group Bull and STMicroelectronics. She was Project Coordinator of three European projects: H2020-ANTAREX, FP7-2PARMA and FP7-MULTICUBE. She has served in the organizing and program committees of several major conferences in computer architectures, embedded systems and electronic design automation. She is Associate Editor of ACM TACO and IEEE TC. In 2017, she has been elevated to the grade of IEEE Fellow.

Carlo Cavazzoni is head of Cloud Computing in Leonardo, and director of the Leonardo HPC Lab. He spent more than 20 years in Cineca, where he become head of HPC R&D, with responsibility for the evolution and exploitation of the national and european HPC infrastructure. He is a member of the EuroHPC Research and Innovation Advisory Board, steering board member of the ETP4HPC association, and Leonardo representative in GAIA-X. He has been graduated in physics at the University of Modena in 1994, and he attained the PhD degree at ISAS-SISSA in 1998. He was responsible for the co-design and exploitation of the EURORA PRACE 2IP prototype and D.A.V.I.D.E. PRACE 3IP PCP prototype, and for the parallel design of QUANTUM ESPRESSO suite of codes. He published more than 100 peer review articles.

Gianluca Palermo received his Master of Science degree in Electronic Engineering, in 2002, and the PhD degree in Computer Engineering, in 2006, from Politecnico di Milano (Italy). He is currently an Associate Professor at the Department of Electronics, Information and Bioengineering (DEIB) at the same University. Previously, he was part of the Low-Power Design Group of AST - STMicroelectronics working on Network-on-Chip architectures, and Research Assistant at the Advanced Learning and Research Institute (ALaRI) of the Universita’ della Svizzera Italiana. His research interests include design methodologies and architectures for embedded and HPC systems focusing on autotuning aspects. Since 2003, he published more than 100 scientific papers in peer-reviewed conferences and journals.

Andrea R. Beccari is currently responsible for the Drug Discovery Platform of Dompé Farmaceutici SpA and leader of the EXSCALATE team. Since 2015, responsible of the Joint Bioinformatics Groups at the IBP Institute of the National Research Council of Italy. He was promoter and coordinator of the open innovation initiative: Italian Drug Discovery Network and co-founder and member of the board of the Avicenna Alliance (Brussel). He was the originator and chairman of the Computational Driven Drug Discovery and Italian Drug Discovery Summit series of meetings. He has co-organized several initiatives with the European Commission and parliament promoting the use of in-silico simulation to increase the awareness towards the potentiality of high-performance computing in healthcare. He is project coordinator of the H2020-EXSCALATE4CoV and EuroHPC-LIGATE projects. He published more than 20 publications in peer review journals and co-author for 7 patents.