POSTER: An Hybrid Approach to accelerate a Molecular Docking Application for Virtual Screening in Heterogeneous Nodes

E. Vitali[*], D. Gadioli[*], A. Beccari[+], C. Cavazzoni[‡], C. Silvano[*], G. Palermo[*]
[*]Politecnico di Milano, Dipartimento di Elettronica Informazione e Bioingegneria, Milan, Italy
[+]Dompé Farmaceutici SpA, L'Aquila, Via Campo di Pile, Italy
[‡] Cineca, Supercomputing Innovation and Application Department, Bologna, Italy

ABSTRACT

Molecular Docking is a crucial task in the process of Drug Discovery. This task consists in the estimation of the position of a molecule inside the docking site. It is used in the early stages of the drug discovery process to perform a virtual screening of a large library of molecule candidates. This task is usually performed using High Performance Computing platforms, due to sheer number of candidates and due to complexity of the docking problem. In this work we ported and optimized a Molecular Docking Module to an heterogeneous system with one or more GPGPU accelerators, leveraging the directive languages OpenMP and OpenACC. We show that with the proposed approach, we are able to reach a better utilization of the available resources compared to the usual CPU/GPU data splitting, reaching a 25% throughput improvement within the single node.

CCS CONCEPTS

• **Computing methodologies** → **Massively parallel and high-performance simulations**; • **Software and its engineering** → *Software performance*; • **Applied computing**;

KEYWORDS

HPC, Molecular Docking, Heterogeneous Computing, OpenMP, OpenACC

ACM Reference Format:
E. Vitali[*], D. Gadioli[*], A. Beccari[+], C. Cavazzoni[‡], C. Silvano[*], G. Palermo[*]
. 2019. POSTER: An Hybrid Approach to accelerate a Molecular Docking Application for Virtual Screening in Heterogeneous Nodes. In *Proceedings of Proceedings of the 16th conference on Computing Frontiers (CF '19)*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The drug discovery process consists of several different tasks, from simulations performed `in-silico` to experiments performed `in-vivo`. The starting point is a huge set of candidates that is narrows in each stage of the process, to find the target drug. In the early phase of the process the focus is to select from an enormous amount of candidates, a small set of candidates molecules that have a strong interaction with the target binding site. This process is performed with computer simulations and it leverages molecular docking algorithms. They estimate the strength of the interaction between the candidate molecule, called *ligand*, and the target binding site, called *pocket*. The complexity is due to the dependency between the strength of the interaction and the displacement of the *ligand* atoms. Therefore, to correctly estimate the strength of the interaction, a molecular docking application needs to correctly estimate the *ligand* atoms displacement, named pose, when it interacts with the target *pocket*. Beside rigid transformation, it is possible to change the geometrical shape of *ligands*, without altering its chemical properties.

Given the high number of involved degrees of freedom, this task is very computationally expensive, and it is performed using High Performance Computing (HPC) machines. Since nowadays the performance of an HPC machine is limited from the power consumed, energy efficiency became a key issue. Modern HPC platforms tend to improve their energy efficiency using hardware accelerators such as GPGPUs [4]. Since the evaluation of different *ligands* are independent from each other, the first stage of the drug discovery process can be considered data parallel. Moreover, the evaluation of each pose of a *ligand* is still independent of the evaluations of other poses. This parallelization pattern matches the GPUs computation paradigm. For this reason we propose an optimized strategy to exploit all the computational resources of a node that includes one (or more) GPGPU.

2 PROPOSED APPROACH

There are two paths explored in literature to tackle the molecular docking problem [3]. The first one employs stochastic algorithms to sample the space of the *ligand* poses. The second approach uses geometrical and chemical properties of the target *pocket* and of the evaluated *ligand* to drive the docking process, following a heuristic.

In this work, we focus on one module of LiGenDock [1], called *GeoDock*, that relies on the second type of approaches. *GeoDock* is a docking application that considers only geometric features of the *ligand* and of the target *pocket*. It uses a greedy algorithm to refine the *ligand* pose, with multiple restarts. Given an initial pose of the *ligand*, the first phase of the algorithm performs rigid transformations to align the *ligand* pose with the target *pocket*. This kernel is called *alignment*. The second phase changes the *ligand* shape to optimize its pose according to the geometric heuristic. This kernel is called *optimize pose*. In this phase, we rotate sequentially different subsets of *ligand* atoms, named fragments. At the end of the second phase, when we have maximized the score of the *ligand*

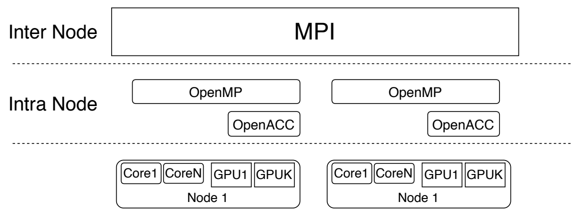

Figure 1: Overview of the proposed implementation of the docking algorithm. We want to maximize the utilization of all the available resources and we leverage OpenACC for the GPU utilization, OpenMP for the CPU workload sharing and MPI for the inter node communication and work sharing.

pose, we restart with another initial pose of the same *ligand*. From the *GeoDock* functions profiling, we identified these two phases as candidate kernels to be offloaded for acceleration.

In a first attempt, we used OpenACC directives to offload both the kernels to the GPU, refactoring their implementation to match the GPU computation paradigm [5]. Therefore, we removed from the CPU all the computation intensive kernels. This GPU version of the application suffers from two main drawbacks. On one hand, the CPU is under utilized. On the other hand, not all the kernels seems suitable for GPU acceleration. While the *alignment* kernel has a 16x speedup, the *optimize pose* kernel has only a 2x speedup.

To solve both these issues, we decided to create an hybrid version of the application that maps each kernel to the more suitable computation resource. From the implementation point of view, we use OpenMP threads to assign to each CPU of the node a task that docks a different *ligand*. The task offloads the alignment kernel to the GPU. Figure 1 depicts the application structure overview: we use MPI to leverage data parallelism at node level. We use OpenMP to leverage intra-node parallelism, by docking different ligands in each CPU. Finally, we use OpenACC to leverage the node heterogeneity, by offloading the alignment kernel to the GPUs.

3 RESULTS

To evaluate the benefits of the proposed approach, we performed an experimental campaign using one GPU node of the GALILEO2 machine at CINECA[1]. The target node is equipped with a 2x8-core Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz CPU and two NVIDIA Tesla K80 GPU cards. The operating system is CentOS 7.0, and we compiled the program using PGI Compiler 17.10 enabling the *fastsse* flag to activate the vectorization on top of the *O3* optimization level.

In this experiment we use as *baseline* the traditional approach of data partitioning among CPUs and GPUs. In particular, we run a GPU version of the application for each physical GPU available in the node, while we run the CPU version on the remainder CPUs of the node. We compare the baseline with the proposed hybrid approach that uses all the heterogeneous node (CPUs+GPUs). To measure the throughput of the application, we profiled several runs using a different set of *ligands* and *pocket*. In each experiment the application has to dock 1500 *ligands*. This experimental setup represents a workload of a slave MPI process running on a single node, in the context of the larger master-slave MPI application. We tuned the parameters of the application in terms of GPU kernels, such as grid sizes, and number of OpenMP threads to feed the GPUs to maximize the average throughput for the hybrid case.

[1]http://www.hpc.cineca.it/hardware/galileo-0

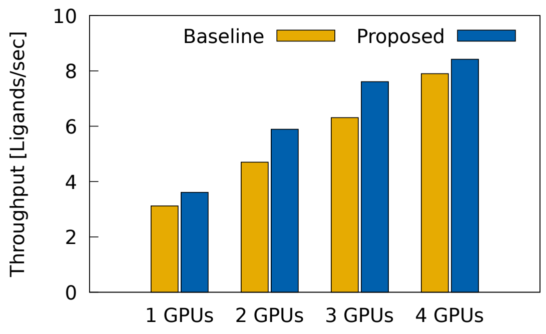

Figure 2: Comparison of the baseline configuration of *GeoDock* with the proposed hybrid approach, by changing the number of GPUs.

Figure 2 shows the application throughput, by varying the number of available GPUs. In all cases, the hybrid version reaches an higher throughput than the original version. This is due to the best exploitation of the GPUs architectures. In particular, the performance improvement in the case of 1, 2, 3 and 4 GPUs is respectively of 15%, 25%, 20%, and 6%. When we have 4 GPUs, the hybrid approach is not able to fully exploit all the GPUs.

4 CONCLUSIONS AND ONGOING WORK

In this work, we analyzed an approach to harness the heterogeneity of a computation node by mapping the application kernels to their most suitable computation devices. Experimental results show how *GeoDock* reaches an higher throughput using the proposed approach rather than partition the data among GPUs and CPUs.

The *GeoDock* algorithm exposes parameters to enable approximate computing, exploiting the accuracy-throughput trade-off. The approximation level and features of the *ligand* (e.g. the number of atoms) affects the selection of the best grid sizes for the GPUs computation as well as the number of OpenMP threads to be used to feed the GPUs. Therefore, we are investigating the benefits of tuning those parameters dynamically [2], according to approximation level and to the features of the actual input, rather than considering the average behavior.

REFERENCES

[1] C. Beato, A. Beccari, C. Cavazzoni, S. Lorenzi, and G. Costantino. 2013. 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* 53, 6 (2013), 1503–1517.
[2] D. Gadioli, E. Vitali, G. Palermo, and C. Silvano. 2018. mARGOt: a Dynamic Autotuning Framework for Self-aware Approximate Computing. *IEEE Trans. Comput.* (2018). https://doi.org/10.1109/TC.2018.2883597
[3] Nataraj S Pagadala, Khajamohiddin Syed, and Jack Tuszynski. 2017. Software for molecular docking: a review. *Biophysical reviews* 9, 2 (2017), 91–102.
[4] Top500.org. 2018. TOP 500 List. https://www.top500.org/lists/2018/11/.
[5] E. Vitali, D. Gadioli, G. Palermo, A. Beccari, and C. Silvano. 2018. Accelerating a Geometric Approach to Molecular Docking with OpenACC. In *Proceedings of the 6th International Workshop on Parallelism in Bioinformatics (PBio 2018)*. ACM, New York, NY, USA, 45–51. https://doi.org/10.1145/3235830.3235835