Parallelizing iGEMDOCK – a fundamental virtual screening software for recognizing pharmacological interactions in drug design

Wei-cheng, Chang Zheng-yu, Tong

January, 2023

Abstract

In this project, we leveraged CUDA, a parallel computing platform, to enhance the efficiency of virtual screening processes. By utilizing CUDA's capabilities to handle large-scale computations simultaneously, we accelerated the molecular docking. This advancement has resulted in a substantial reduction in the time required for virtual screenings, facilitating faster and more efficient identification of potential therapeutic compounds. Our contribution demonstrates the powerful impact of integrating CUDA in the field of drug design, opening new avenues for rapid drug discovery and development.

1 Introduction

Virtual screening is a crucial computational technique in the field of drug design and structural biology, which aids researchers in predicting the binding affinities between various ligands and the targeted protein. It plays an important role in understanding how small molecules, often drug candidates, interact with protein subpockets, which is a fundamental step in identifying potential drug candidates. However, as the demand for faster and more accurate drug discovery processes grows, it is evident that the computational workload associated with virtual screening is a bottleneck. The computational cost of docking millions of potential compounds against a protein target can be relatively high due to complicated physical and chemical properties.

A pair of recently released studies shed new light on the staggering cost of developing new drugs—an expense that now exceeds $2 billion per therapy on average. The average cost of developing a new drug among the top 20 global biopharmas rose 15% ($298 million) last year according to the investigation by the business services consultancy Deloitte. Therefore, developing a parallelized virtual screening software is crucial and valuable in drug design, which can provide useful information for researchers to screen promising drug-target pairs before investing a huge amount of cost in experiments.

iGEMDOCK[1] is a fundamental virtual screening software for recognizing pharmacological interactions, which allows users to input an interested protein and its drug candidates. This software operates by implementing a genetic algorithm to enable the drug candidates to find the most stable position within the protein's cavity.

2 Proposed approaches

The genetic algorithm[2], a method inspired by natural selection, iteratively evolves a population of candidate solutions toward an optimized state. In the context of iGEMDOCK, each candidate solution represents a potential orientation and positioning of a ligand within the protein's binding site. The procedure involves several key steps:

- Initial Population: The algorithm begins with a population of randomly generated candidate solutions. Each candidate represents a potential conformation and positioning of a ligand within the target protein's cavity.
- Fitness Evaluation: Each candidate solution is evaluated based on a fitness function, which is to calculate the interaction energy between the ligand and the protein's cavity. Lower energy leads to higher stability and binding affinity.
- Selection: Based on fitness evaluations, a subset of the population is selected to breed a new generation. The selection process in iGEMDOCK favors lower-fitness candidates.
- Crossover (Recombination): Pairs of candidates are combined to form new offspring. Crossover involves swapping portions of the candidates' data (e.g., molecular coordinates or conformational angles), which results in new candidate solutions.
- Mutation: To introduce additional variability into the population, some candidates undergo mutation, which involves randomly altering parts of their data. This step is crucial for exploring new parts of the solution space that might not be reached through crossover alone.
- New Generation: The offspring from crossover and mutation processes form a new generation of candidate solutions.
- Termination: This process repeats for a set number of iterations (generations).

The procedure allows iGEMDOCK to efficiently explore the vast space of possible interactions, identifying configurations where the drug candidates form the most stable and effective interactions with the target protein.

We utilized two profiling tools, gprof and perf, to run iGEMDOCK and discovered that the most time-consuming part is the Fitness Evaluation. This stage involves calculating the interaction energy between the ligand and the protein's cavity. Specifically, during the calculation of energy, each atom in the protein's cavity calculates the energy against each atom in the ligand. The energy is the summation of three types of molecular interactions: H-bond, Van der Waals Forces, and electrostatic force, and their calculations are encapsulated into three functions.

We parallelize the process of calculating energy, which means that each atom on the protein's cavity simultaneously computes the three aforementioned types of energy with all atoms of the ligand. We anticipate that this approach will accelerate this time-consuming process. Three functions related to energy are defined as device functions and the protein's cavity and the ligand will be copied to the CUDA memory to enable device kernel function to access. The kernel function handles the process of the function calls for each atom in the protein's cavity and stores the result to the array in the device. Once all threads finish the tasks, The energy will be summed up in the host and returned to the fitness evaluation.

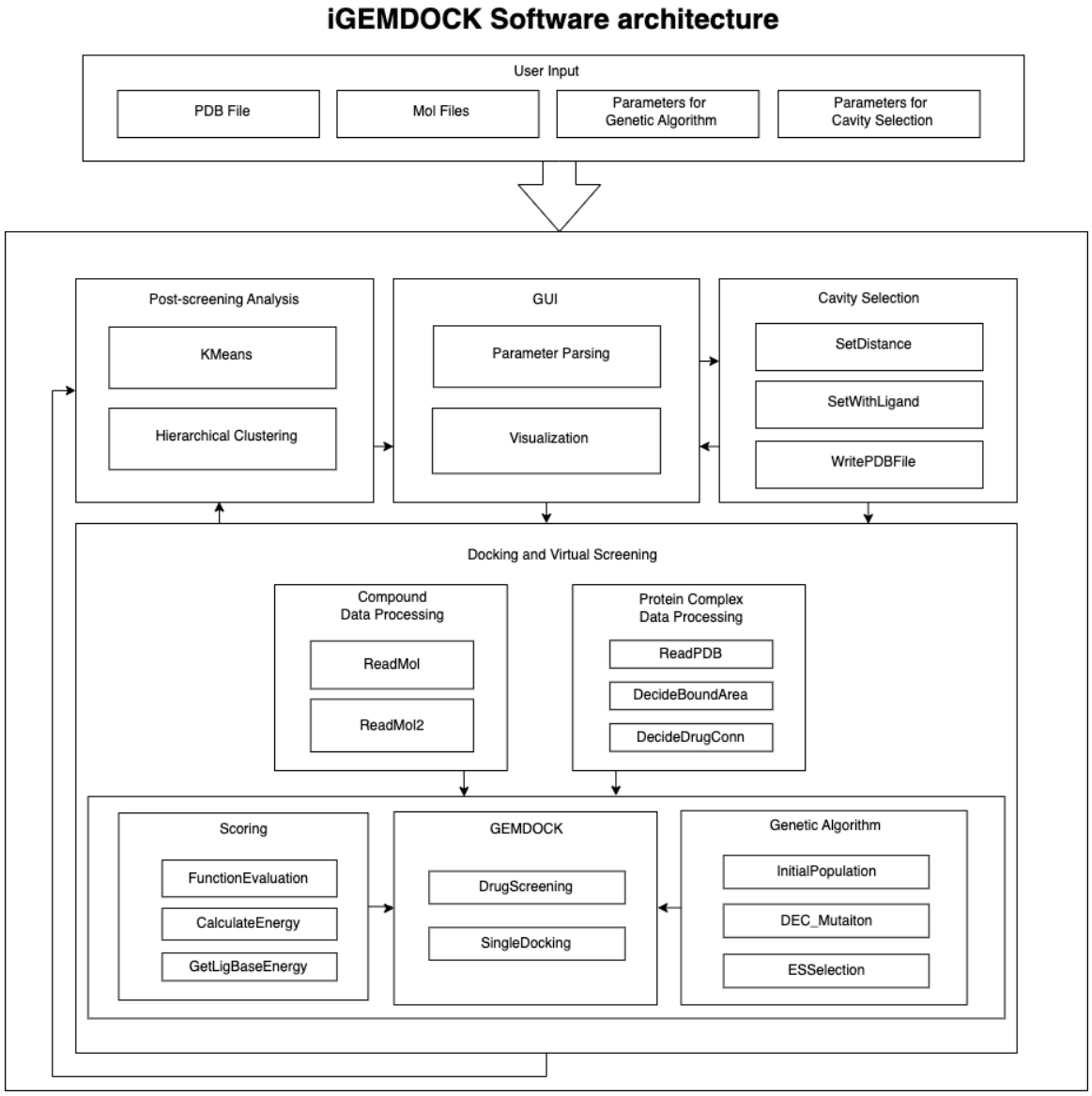

Figure 1: iGEMDOCK Software architecture

3 Experimental Methodology

We designed different scenarios for comparing performance between CUDA and CPU. The protein's cavity is fixed, but the number of ligands is separated into 1, 10, and 100. The atom number of ligands shown in the table is the average atom count of the ligand list with the different sizes. Each dataset is tested on different levels of population size: 2, 10, and 100.

	Dataset 1	Dataset 2
ligand Atom Number	82	30
Protein Cavity Atom Number	916	916

Table 1: Datasets with different average ligand atom number

4 Experimental Results

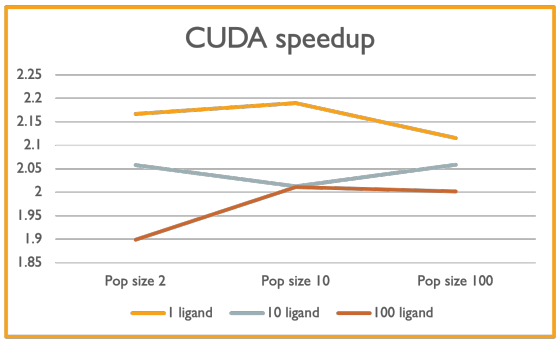

Figure 2: Speedup of Dataset 1

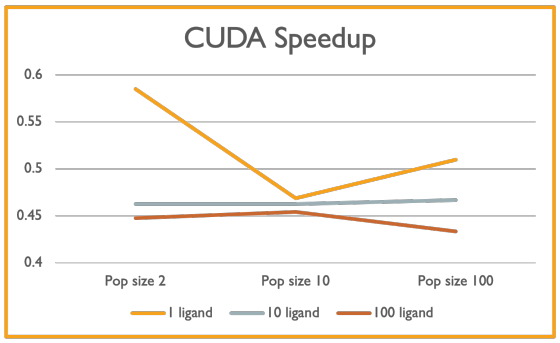

Figure 3: Speedup of Dataset 2

In both figures, as the size of the ligand list increases, the speedup decreases. The population size does not significantly affect the effectiveness of parallelism. However, we could notice that the larger the disparity in the number of atoms between the protein's cavity and the ligand, the lower the effectiveness of CUDA parallelism. Below is the analysis of possible reasons leading to the situation.

Load Imbalance: When there is a large difference in the number of protein and ligand atoms, it can lead to load imbalance in parallel computing. Some threads or processing units may have more work (more atom interactions to compute) compared to others. This imbalance can result in inefficient use of computational resources, as some threads will finish their tasks earlier and remain idle while others are still processing.

Memory Utilization: Efficient parallelization often relies on optimal memory usage. A significant difference in the number of atoms can lead to challenges in memory management, particularly in ensuring that data is evenly and efficiently distributed across the memory hierarchy (like shared memory, caches, and registers in GPUs).

Memory Transferring and Communication Overhead: In parallel computing, especially in CUDA computing platform, there is a need for communication between the host and the device as data might need to be synchronized across different units, and data copying before and after computing. A disproportionate number of atoms can increase the overhead, leading to potential bottlenecks.

Algorithmic Complexity: The algorithms used for calculations (like energy computation, docking simulations, etc.) may have varying efficiencies depending on the number of atoms. A larger discrepancy in atom counts can exacerbate these inefficiencies, as the algorithms might not scale linearly with the number of atoms.

5 Related work

The development of virtual screening (VS) has seen significant advancements over the years, with a focus on parallelization for improved performance and efficiency. Originally, many VS tools were sequential, running on single-core CPUs. However, as computational resources have advanced, so has the parallelization of VS. Here, we discuss the evolution of VS tools and their parallel implementations, showcasing key developments in this field[3].

Early implementations of VS relied on sequential processing, with software like Dock5[4] and Dock6[5] being widely used. They primarily running on single-core CPUs and use MPI for acceleration. These tools utilized physics-based scoring functions (geometric shape-matching approach) but were limited by their inability to leverage multiple CPU cores effectively.

As computational power increased, the need for parallelization became apparent. Tools like Autodock Vina[6] emerged, offering multi-threading options to run on multi-CPU by using OpenMP. It retained the physics-based scoring functions but introduced the ability to use multiple CPU cores simultaneously for molecular docking calculations. Autodock Vina allowed for both single-threaded and multi-threaded options, providing researchers with the flexibility to choose their preferred mode of operation.

By 2011 to 2013, the introduction of MPAD4[7], VinaLC[8], and VinaMPI[9] represented a hybrid scheme for parallelization across nodes. They used MPI and OpenMP to allow efficient allocation of computing resources, resulting in these scalable and high-throughput VS tools. The introduction of GPU acceleration marked a significant milestone in VS. Autodock-GPU[9], for example, was developed to run on multiple-node parallel computers with GPU accelerators by using OpenCL. The use of GPUs greatly improved the performance of VS tools, as they could handle massive parallelism efficiently.

Over time and technology development, machine learning-based scoring functions were introduced, offering better accuracy in predicting binding affinities and improving the efficiency of lead compound identification. These scoring functions were integrated into tools like GNINA[7], enhancing their predictive capabilities.

6 Conclusions

In our study on parallelizing the virtual screening software iGEMDOCK using CUDA, we have observed that a greater disparity between the number of ligand and protein receptor atoms may increase the likelihood of overhead. This overhead can diminish the advantages offered by parallelization, as the workload is not evenly distributed, leading to less efficient use of computational resources.

To conclude, to maximize the benefits of parallelization, more flexible strategies should be employed. For varying numbers of ligand atoms, pthreads with CPU shared memory can be effectively parallelized to manage smaller atom counts. In cases where the atom count is high, utilizing CUDA becomes more advantageous due to its superior handling of larger, more complex calculations.

Our contribution lies in identifying the optimal conditions under which either CPU-based pthreads or GPU-based CUDA can be leveraged to enhance the efficiency of virtual screening processes. This adaptive approach ensures that the parallelization technique aligns with the computational demands of the task, thereby streamlining the virtual screening pipeline for faster and more accurate drug discovery.

References

[1] Shen-Rong Lin Jinn-Moon Yang Kai-Cheng Hsu, Yen-Fu Chen. igemdock: a graphical environment of enhancing gemdock using pharmacological interactions and post-screening analysis. *Bioinformatics*, 12(1):33, 2011.

[2] John H. Holland. Genetic algorithms. *Scientific American*, 267(1):66, 1992.

[3] Natarajan Arul Murugan, Artur Podobas, Davide Gadioli, Emanuele Vitali, Gianluca Palermo, and Stefano Markidis. A review on parallel virtual screening softwares for high-performance computers. *Pharmaceuticals*, 15(1):63, 2022.

[4] Demetri T Moustakas, P Therese Lang, Scott Pegg, Eric Pettersen, Irwin D Kuntz, Natasja Brooijmans, and Robert C Rizzo. Development and validation of a modular, extensible docking program: Dock 5. *Journal of computer-aided molecular design*, 20:601–619, 2006.

[5] William J Allen, Trent E Balius, Sudipto Mukherjee, Scott R Brozell, Demetri T Moustakas, P Therese Lang, David A Case, Irwin D Kuntz, and Robert C Rizzo. Dock 6: Impact of new features and current docking performance. *Journal of computational chemistry*, 36(15):1132–1156, 2015.

[6] Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010.

[7] Andrew T McNutt, Paul Francoeur, Rishal Aggarwal, Tomohide Masuda, Rocco Meli, Matthew Ragoza, Jocelyn Sunseri, and David Ryan Koes. Gnina 1.0: molecular docking with deep learning. *Journal of cheminformatics*, 13(1):1–20, 2021.

[8] Xiaohua Zhang, Sergio E Wong, and Felice C Lightstone. Message passing interface and multithreading hybrid for parallel molecular docking of large databases on petascale high performance computing machines. *Journal of computational chemistry*, 34(11):915–927, 2013.

[9] Sally R Ellingson, Jeremy C Smith, and Jerome Baudry. Vinampi: Facilitating multiple receptor high-throughput virtual docking on high-performance computers, 2013.