UNI-DOCK: A GPU-ACCELERATED DOCKING PROGRAM ENABLES ULTRA-LARGE VIRTUAL SCREENING

Yuejiang Yu
DP Technology
Peking University
yuyj@dp.tech

Chun Cai
DP Technology
caich@dp.tech

Zhengdan Zhu *
DP Technology
zhuzd@dp.tech

Hang Zheng *
DP Technology
zhengh@dp.tech

July 19, 2022

ABSTRACT

Molecular docking, a structure-based virtual screening method, is a reliable tool to enrich bio-active molecules from molecular databases. With the expansion of the size of virtual libraries, the speed of existing molecular docking programs becomes less than adequate to meet the demand for screening ultra-large libraries containing tens of millions or billions of molecules. Here we propose Uni-Dock, a GPU-accelerated molecular docking program supporting various scoring functions including vina, vinardo, and ad4, which achieves more than 1000-fold speed-up with high accuracy compared with the AutoDock Vina single-CPU-core version, outperforming reported GPU-accelerated docking programs including AutoDock-GPU and Vina-GPU. Uni-Dock divides molecules into batches and simultaneously docks batches of molecules using hundreds of concurrent threads for each molecule. The data flow between GPU and CPU is optimized to eliminate CPU hotspots and maximize GPU utility. We demonstrate and analyze the improved performance of Uni-Dock on the CASF-2016 and DUD-E datasets and recommend three combinations of hyperparameters corresponding to different docking scenarios. To demonstrate Uni-Dock's capability on routinely screening ultra-large libraries, we performed hierarchical virtual screening experiments with Uni-Dock on Enamine Diverse REAL drug-like set containing 38.2 million molecules to a popular target KRAS G12D in 12 hours using 100 NVIDIA V100 GPUs.

Keywords Molecular Docking · Virtual Screening · GPU Acceleration

1 Introduction

Virtual screening (VS), which aims to identify hit compounds as the start point for drug discovery, plays a crucial role in modern drug design due to its high efficiency, low cost, and freedom from industrial product limitations compared to high-throughput screening (HTS) [1, 2]. Among various VS approaches, molecular docking, a structure based method, has become one of the most popular tools [3]. Molecular docking searches for the binding pose of a ligand within a receptor, and calculates the binding affinity of the resulting receptor-ligand complex. Various benchmark studies have proved that molecular docking has a good ability to predict the binding pose and to enrich bio-active molecules [4, 5, 6, 7].

Previous works indicate that the screening of larger molecular libraries probably lead to candidate compounds with higher potency [8]. Considering the "low-hanging fruits" are rare in modern drug discovery, the pursuit of chemical diversity to empower drug discovery of targets with high value, as well as difficulty, is hoped. Possibly in response to this trend, a rapid increase in the size of commercial libraries was observed in recent years. A representative example is Enamine REAL Space[9], which comprises 22.7 billion make-on-demand molecules and is challenging the virtual screening efficiency [10]. On the other hand, urgent public health events such as COVID-19 pandemic and monkeypox virus infection place high demands on speedy screening of databases against new receptors [11]. Consequently, widely

*Corresponding authors.

used molecular docking methods, like AutoDock4 [12], AutoDock Vina [13] and Glide [14, 15], can no longer meet researchers' requirements for the speed and cost of virtual screening owing to the bottleneck of computational resources [16].

To speed up screening computation, several efforts have been made to devise better methodologies and achieve accelerations primarily based on CPUs. AutoDock Vina 1.2 [17] is compatible with ad4 scoring function, so that the speed of AutoDock4 can take advantage of the acceleration of MC/BFGS [18] searching method in AutoDock Vina. QuickVina2 [19] and QuickVina-W [20] reduce redundancy in searching by letting searching threads of AutoDock Vina communicate with an record of the explored searching space. However, the resulting improvement of docking speed is generally within one order of magnitude, which might not result in significant benefits when screening ultra-large databases.

In addition, some other works have attempted to reduce the overall running time of large-scale virtual screening through the parallelization on computing clusters. For example, VirtualFlow [16] takes advantage of the linear scalability of docking and calculates 1 billion compounds in approximately 2 weeks when leveraging 10,000 CPU cores simultaneously. In spite of the reduced wallclock time for all compounds, it does not change the average CPU running time for each molecule, thus the cost on computational resources still remains the bottleneck.

With the development of Artificial Intelligence (AI), some efforts like DeepDocking [10] and active learning Glide [21] attempt to train deep learning (DL) models to predict docking score on a subset of the large scale molecular database. However, since deep learning models rely on a large amount of labeled data to improve generalization, training a usable model requires millions of molecular docking first [10], which limits its wide application.

Hence, the urgent need of computationally feasible for virtual screening on ultra-large-scale molecular databases calls for the improvement to the docking efficiency on each molecule. The application of GPU is a promising strategy to achieve such a goal. As a widely used acceleration hardware, GPU has shown its powerful capability of high-performance computing, especially for highly parallelized docking workloads because the scalable array of multi-threaded streaming multiprocessors (SM) of GPU makes it natural performing docking calculations in parallel. Therefore, it would not be a surprise that attempts of migrating molecular docking software to GPU have been made, including AutoDock-GPU [22] and Vina-GPU [23]. However, the speed of AutoDock-GPU is only comparable to AutoDock Vina since it is developed on AutoDock4 which uses low-efficiency global searching algorithm. Vina-GPU makes use of the efficient Monte-Carlo searching method proposed in AutoDock Vina, but the computational power of GPU is not fully utilized, resulting in an average speed-up of about 20 times with 5% worse docking power than AutoDock Vina tested on AutoDock-GPU dataset [22]. The large-scale virtual screening ability of Vina-GPU is not qualified as well, since it doesn't provide the data of enrichment factor on widely-accepted benchmark datasets and only 9125 ligands from DrugBank [24] are docked in the showcase, far less than common scenarios of high-throughput virtual screening. Further, new docking features, such as vinardo and ad4 scoring functions, batch mode and grid map support, developed in version 1.2 of AutoDock Vina are not included in Vina-GPU.

In this work, we propose Uni-Dock, a GPU-accelerated docking method developed on AutoDock Vina 1.2, which reduces the average time cost of docking one molecule to a minimum of 0.1 seconds, enables ultra-large virtual screening, and supports vina, vinardo, and ad4 scoring functions.We demonstrate how we change the searching pattern of Monte-Carlo(MC) algorithm to exploit the marvelous parallelism and memory bandwidth of GPU using various methods.

Experiments on widely-accepted benchmark databases DUD-E [7] and CASF-2016 [5] are performed to demonstrate the screening power and docking capability of Uni-Dock with various parameter configurations and scoring functions. Moreover, to demonstrate the high-throughput virtual screening ability of Uni-Dock, we perform hierarchical virtual screening approach [2] on Enamine Diverse REAL drug-like set [9] and dock all 38.2 million ligands therein to a popular target KRAS G12D (PDBID: 7RPZ) [25] within 12 hours by using a GPU cluster containing 100 NVIDIA V100 GPUs.

2 Methodology

Our improvements consist of three stages. In stage one, We dock one ligand with hundreds of threads on GPU. In stage two, the docking is performed on multiple ligands in parallel with multiple threads running simultaneously. In stage three, we further reduce GPU memory footprint and boost computational speed by using single precision in GPU calculation. And with reduced memory usage of GPU and OpenMP enhanced CPU IO, we calculate ligand batch size on the fly. The architecture of Uni-Dock and its comparison with Vina is depicted in fig.1.

We use eight targets from DUD-E dataset to evaluate screening power and speed of our implementation and select AutoDock Vina operated on CPU with $exhaustiveness = 32$ as our baseline. In each stage, we analyze the drawbacks

of current implementation through detailed profiling results and use hardware-adapted method to accelerate without loss of accuracy.

Overall, Uni-Dock achieves more than 1000× acceleration comparing with single CPU core with the same accuracy. The speed-up ratio and screening performance indicator, enrichment factor (EF) of top 10%, is illustrated in fig.2. The acceleration ratio is calculated by the execution time of Vina on one 2.5 GHz Intel® Xeon® Platinum 8269CY (Cascade Lake) CPU and Uni-Dock on one NVIDIA V100 32G GPU. So far, we support rigid and semi-flexible docking, which means ligands can have rotatable bonds whereas the receptor must be rigid.

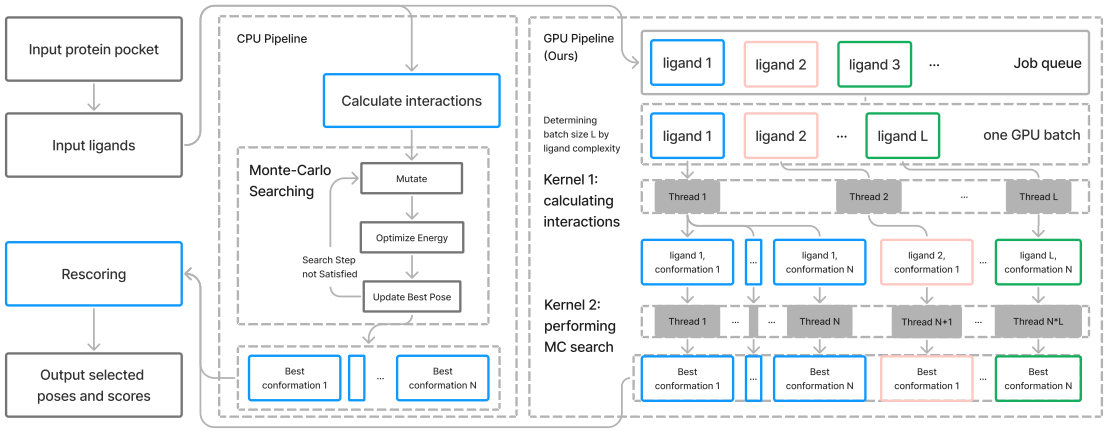

Figure 1: A comparison between the architecture of the algorithm used by AutoDock Vina 1.2 and Uni-Dock: Vina CPU put MC threads in a queue called task container. Each CPU core executes one MC thread to search for good poses as output. The MC search is iterated for a given step. In each step, random mutation of rotatable bonds, evaluate energy and force and gradient descent are performed one by one. Uni-Dock uses thread-level parallelization to utilize the GPU's computational power. Input ligands are processed in parallel. Their MC threads are executed simultaneously on GPU to rapidly cover the whole search space. The specific operations inside each MC search are identical to Vina.

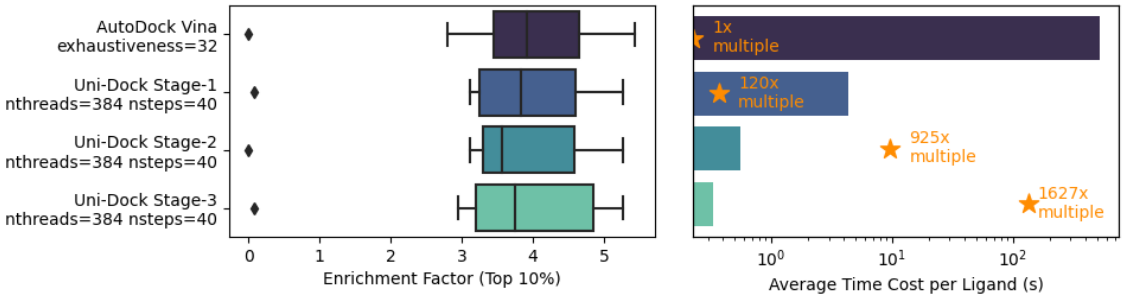

Figure 2: The performance improvements of three critical stages: 1627-fold speed-up with consistent accuracy.

2.1 Stage I: Single ligand, multiple threads

For the first stage of migrating CPU code to GPU, we implement the calculation of MC search in CUDA without any modifications in the searching strategy. This is because each MC thread calculates the derivatives of atoms in the force field needed in BFGS. Uncoalesced random memory access the force map between atoms requires undoubtedly becomes the most critical memory bottleneck. The scheduling of searching tasks is implemented with a CPU task queue. MC jobs with different initial conformations are put into the task queue, and asynchronously distributed to

CPU threads. Like the CPU threads, each GPU thread handles the searching of one initial conformation. It is more advanced that the SIMT (single instruction, multiple threads) architecture of GPU processes the conformations of one ligand simultaneously. Because the GPU computing unit is not able to directly access the main memory, for one GPU batch of ligands, we manage the data flow in the following steps: calculate the interaction maps of the pocket, allocate GPU memory for data and result, transfer the prepossessed data to GPU, call a function that requires GPU data, and gather the result back to CPU memory. To adapt GPU usage, we migrate the data structure from array-of-structure to structure-of-array for transferring memory between CPU and GPU in stride.

After the migration, the main workload MC searching is processed on GPU with the same number of threads ($nthreads$, namely $exhaustiveness$ in Vina) of the CPU pipeline. A typical value of exhaustiveness is 8. However, the number of GPU threads is in the order of 10,000, which means the degree of parallelism is not properly utilized yet. Within our expectation, the running time temporarily becomes longer compared with AutoDock Vina running on CPU. The main cause is the lower single-thread capacity of GPU and the additional overhead of transferring data.

For conformations, the search space is determined by search width - the number of searching threads with different initial poses, and search depth - the times of sampling in MC searches. Since the searching space is high-dimensional, enough threads with random initialized poses are able to exploit possible outcomes and find poses with good scores. This strategy is used and validated in Vina-GPU [23] with a consistent accuracy. Therefore, we change our searching strategy from depth-first search to breadth-first search, utilizing the parallel capacity of GPU. Setting width of over 1000 threads, we limit the search depth to 10-40 steps without missing favorable poses in the search space. With these threads running simultaneously on GPU, the searching speed is drastically improved. We note that the combination of the number of threads per ligand ($nthreads$) and search depth (namely $nsteps$, max_evals in Vina) can significantly affect speed and accuracy. Vina also provides user options to set $exhaustiveness$ and max_evals to balance the speed and accuracy of the docking process. The search depth of Vina is a heuristic value based on ligand properties while $exhaustiveness$ is given by users. Likewise, we test the accuracy and speed of various combinations of these two parameters and we recommend three combination settings: $fast$, $balanced$, and $detailed$. We will discuss these settings in detail with more experimental results in sec.3.

2.2 Stage II: Multiple ligands, multiple threads

In this stage, we perform searching on multiple ligands in one GPU batch with one kernel launch. Since the threads used by one ligand in stage one are still far below the capacity of GPU, we put the processed data of ligands in an array, and transfer it to GPU memory. Each thread can get the assigned input data. Most ligands have 20-40 atoms while some might have more than 100 atoms, and the unit capacity of a fixed-length array is square to the max atom numbers of ligands in a dataset. To maximize memory usage, we allocate memory by the required space of interacting forces between atom pairs of each ligand.

In addition to computational improvement, Uni-Dock take fault tolerance into consideration as well. Various reasons cause errors and faults to happen in large-scale virtual screening. One failed compound should not crash the docking of other ligands. To be specific, we require few broken ligands in a batch not affect the others when it comes to GPU batch mode in Uni-Dock. Therefore, we add error detection and recovery mechanisms in ligand parsing and searching. To ensure enough GPU memory, we check available memory of GPU before MC searches.

The computation speed and user experience benefit from above modifications, since we calculate the most time-consuming part in parallel with low error rate. GPU has a higher memory bandwidth compared with CPU; Therefore, the more ligands we put in one batch, the higher the efficiency is.

The efficiency of the whole system can be further improved. Input ligands require memory to store its properties and interaction maps. Since the memory size of GPU limits the maximum number of ligands in one batch, we fix the batch size to about 200 to avoid out-of-memory failure. In addition, I/O expense and preprocessing calculations on the CPU become new hotspots which take more than 60% of overall running time.

2.3 Stage III: Exploit GPU

We further exploit GPU memory and calculating ability in this stage.

Firstly, we move computationally intensive preprocessing to GPU by calculating interaction forces between atoms of each ligand simultaneously on GPU with a higher throughput. This improvement also eliminates the large transfer of preprocessed data from CPU to GPU.

Besides, we replace double-precision floating point numbers with single-precision ones. As a result, single-precision data saves memory and speeds up arithmetic operations. Docking score is a rough prediction of experimental binding

affinity [4] compared to other methods [26, 27], and the score is coarse-grained (for non-outliers, $[-15.0, 0.0]$ with an accuracy of 0.1). In single precision, we save more than 30% of kernel running time while preserving almost identical energy scores and we can increase batch size by up to two times. As shown in fig.3, the correlation coefficient between the result of end-to-end searching using double-precision float and the result of single-precision float is 0.87, tested on eight targets from DUD-E dataset. The difference between single and double precision in scoring alone is negligible.

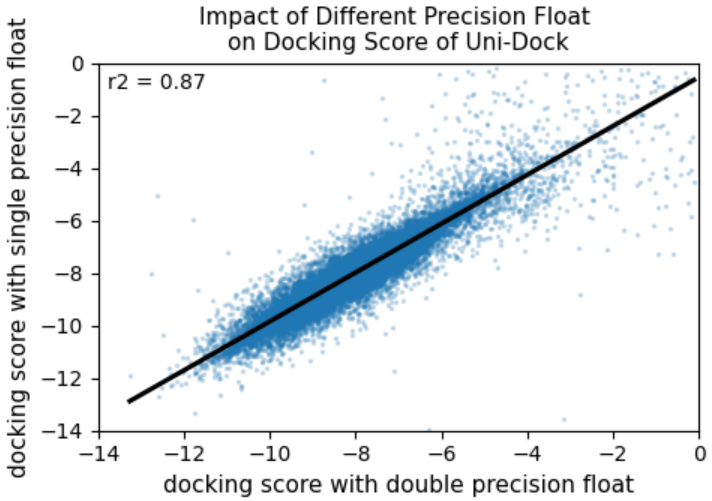

Figure 3: The correlation of docking scores under double precision float and single precision float.

To fit maximum docking computation on GPU, we need to know the peak memory usage *(*a priori), ahead of GPU computation before data transfer to GPU. A linear regression model for memory usage is developed on ligand properties and docking settings to achieve such a goal. Our regression function of vina/vinardo scoring function based on NVIDIA V100 32G GPU is:

$$memory = 0.011978 \times \sum_{i=1}^{batch\_size} atom\_number_i^2 + 1.214869 \times batch\_size + 0.003852 \times thread + 20017 \quad (1)$$

Here, $memory$ (in MB) is the predicted peak global memory usage on GPU, $atom\_number_i$ is the number of atoms in the i-th ligand in current batch, $batch\_size$ is the number of ligands put in one batch to be searched in parallel, $thread$ is the overall number of MC thread in this batch, which equals to $nthreads \times batch\_size$, and the constant stands for temporary memory usage during MC searches. The memory function of ad4 scoring function is slightly different due to the complex energy terms of ad4, which is:

$$memory = 0.079216 \times \sum_{i=1}^{batch\_size} atom\_number_i^2 + 1.911645 \times batch\_size + 0.003910 \times thread + 20052 \quad (2)$$

We put ligands into one batch until the predicted memory hits hardware limitation. The R^2 of our regression reaches 0.999 on real data, which enables Uni-Dock to predict peak memory usage accurately. With this method, we make use more than 95% of the GPU memory with appropriate batch size. We apply this method to NVIDIA T4 16G GPU and get accurate memory usage predictions as well. Uni-Dock can automatically choose the correct memory function according to GPU architecture and parameters.

3 Results

3.1 Performance of Uni-Dock with typical parameter combinations

The application of molecular docking software in different scenarios has different requirements for accuracy and speed. For example, high efficiency is often required in virtual screening of large-scale databases, while high precision is required in binding mode prediction.

To meet the requirements of different scenarios, We recommend three combinations of hyperparameters controlling calculation speed and precision, namely $fast$ mode ($nthreads$ is 128 and $nsteps$ is 20), $balanced$ mode ($nthreads$ is 384 and $nsteps$ is 40) and $detailed$ mode($nthreads$ is 512 and $nsteps$ is 40).

To demonstrate the performance of Uni-Dock under different hyperparameter combinations, we test the docking power (*i.e.*, the ability to predict the native ligand binding pose) and screening power(*i.e.*, the ability to find bio-active ligands among a pool of bio-active ligands and decoys) of Uni-Dock using CASF-2016 dataset and DUD-E dataset with vina scoring function, respectively. Docking power is calculated by calculating the proportion of the total that the RMSD of the binding pose given by Uni-Dock and crystal structure is less than the threshold. Screening power is calculated by calculating the proportion of the total that the enrichment factor given by Uni-Dock is lager than the threshold. For 285 targets in CASF-2016 dataset, cocrystallized ligand for each is docked. For 102 targets in DUD-E dataset, active and decoy sets for each are docked. The receptor structures and ligands were processed on the web service Hermite™(`https://hermite.dp.tech`). Ligands are set flexible during docking, receptor is rigid.

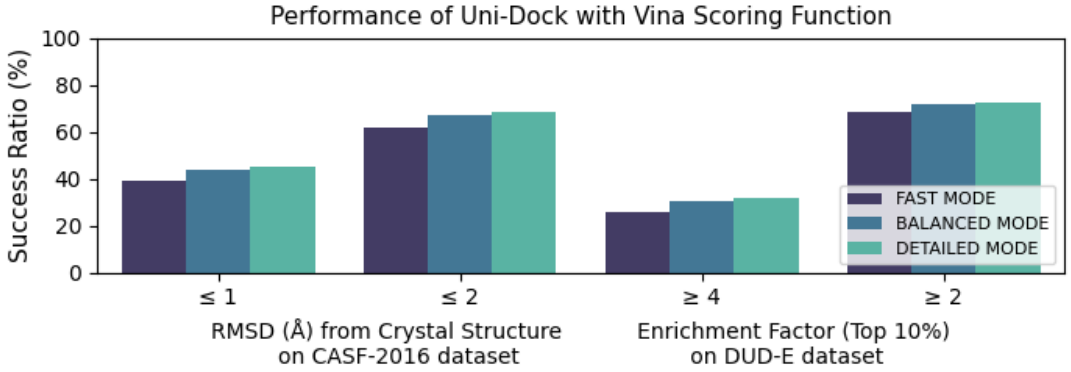

Figure 4: Performance of Uni-Dock with typical parameter combinations. Uni-Dock $detailed$ mode can achieve the best docking performance on both docking power and screening power.

As shown in fig. 4, Uni-Dock $detailed$ mode can achieve the best performance on both docking power and screening power. In the docking power task, the RMSD of binding pose predicted by Uni-Dock $detailed$ mode from the crystal structure is within 2Å in 68.07%(194/285) of targets in CASF-2016 dataset, and within 1Å in 46.67%(133/285) of targets in CASF-2016 dataset. In screening power task, the enrichment factor carried out by Uni-Dock $detailed$ mode is larger than 2 in 72.55%(74/102) of targets in DUD-E dataset, and larger than 4 in 31.37%(32/102) of targets in DUD-E dataset. The performance of Uni-Dock $balanced$ mode is slightly worse than Uni-Dock $detailed$ mode, so as $fast$ mode than Uni-Dock $balanced$ mode. On the other hand, the success ratio of $EF_{top10\%} \geq 2$ for Uni-Dock $fast$ mode reaches 68.63%, only 3.92% smaller than the score of Uni-Dock $detailed$ mode, which indicates Uni-Dock $fast$ mode can perform as well as Uni-Dock $detailed$ mode on many systems that Uni-Dock $detailed$ mode has good performance on.

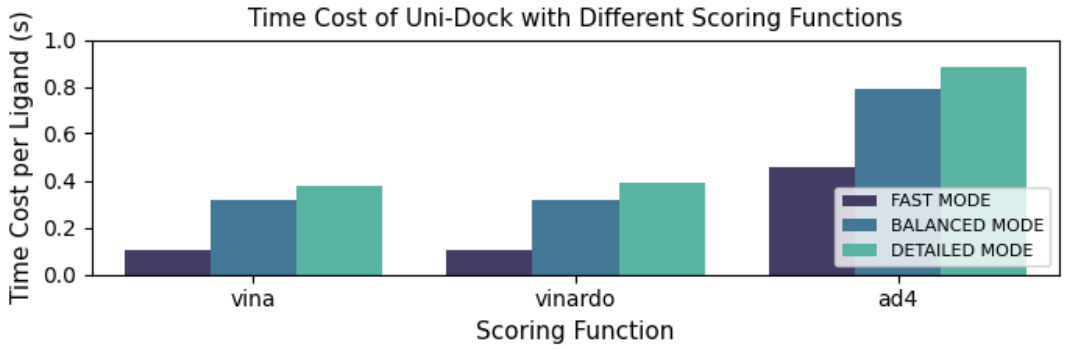

Figure 5: Performance of Uni-Dock using different scoring functions tested on DUD-E dataset. The vina and vinardo scoring function are similar, resulting in analogous running time, while ad4 scoring has more complex energy terms and costs more time.

Different hyperparameter combinations also mean different calculation speeds. As shown in fig. 5, with vina scoring function, the average time cost for docking each ligand is around 0.10 seconds for Uni-Dock $fast$ mode, around 0.32 seconds for Uni-Dock $balanced$ mode, and around 0.38 seconds for Uni-Dock $detailed$ mode. Different scoring functions also have an impact on speeds. Since vina and vinardo scoring functions share similar energy terms, the performance of calculation speed is very similar. And ad4 scoring function has more energy terms, so the average time cost for each ligand is slightly larger, around 0.46 seconds for Uni-Dock $fast$ mode, around 0.79 seconds for Uni-Dock $balanced$ mode, and 0.89 seconds for Uni-Dock $detailed$ mode. While using AutoDock Vina 1.2 with exhaustiveness equal to 8, the average time cost for docking each ligand is 124.56 seconds using vina scoring function, 149.99 seconds using vinardo scoring function, and 335.89 seconds for ad4 scoring function, respectively.

Above on, for high-throughput virtual screening tasks, we recommend using Uni-Dock $fast$ mode, since it can perform well on most of the targets that Uni-Dock $detailed$ mode performs good, while the speed is over 2 multiple $fast$er than Uni-Dock $detailed$ mode. And for the binding pose prediction task, we recommend using Uni-Dock $detailed$ mode, since the Uni-Dock $detailed$ performs best. The Uni-Dock $balanced$ mode has moderate performance in both precision and speed, which is suitable for virtual filtering on small datasets.

3.2 Virtual screening on ultra-large dataset using Uni-Dock

To demonstrate the advantages of Uni-Dock in high-throughput screening, we conducted a high-throughput virtual screening of the recently popular drug target, KRAS G12D (PDBID: 7RPZ), using Enamine Diverse REAL drug-like set [9] containing 38.2 million ligands.

In order to take into account both speed and precision, we adopt a hierarchical virtual screening approach. We first applied Uni-Dock $fast$ mode to dock the 38.2 million ligands of Enamine Diverse REAL drug-like set. Then, take out the top 10% of the ligands (3.82 million ligands) obtained in the previous step, and apply the Uni-Dock $balanced$ mode for docking. Finally, take out the molecules with a score of 10% ($\sim 382,000$ ligands) obtained in the previous step, and apply Uni-Dock $detailed$ mode for docking to obtain the final results.

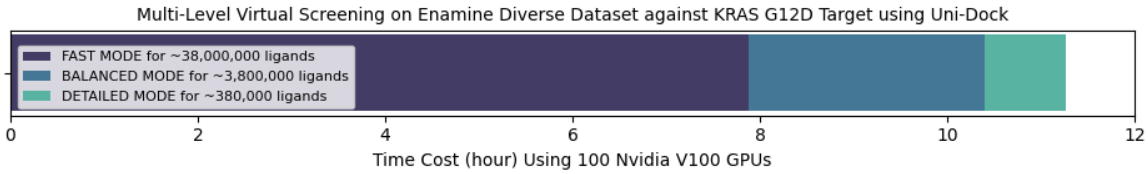

Figure 6: Time cost of hierarchical virtual screening approach on Enamine Diverse REAL drug-like set against KRAS G12D target with Uni-Dock using 100 NVIDIA V100 GPUs.

We use a GPU cluster with 100 NVIDIA V100 GPUs for the hierarchical virtual screening approach described above. As shown in fig.6, the docking step using Uni-Dock $fast$ mode on 38 million molecules took 7.88 hours, accounting for 69.92%, the docking step using Uni-Dock $balanced$ mode on 3.8 million molecules took 2.52 hours, accounting for 22.36%, and the docking step using Uni-Dock $detailed$ mode on 0.38 million molecules took 0.87 hours, accounting for 7.72%.

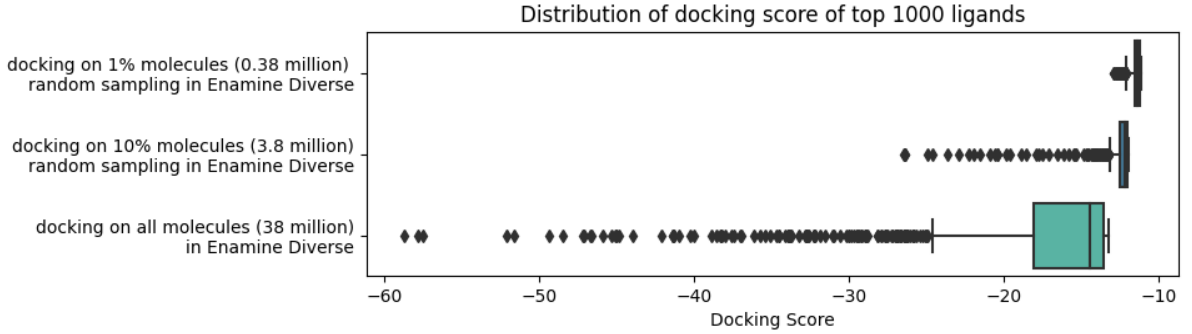

Figure 7: Distribution of docking score of top 1000 ligands for subsets from 38 million molecules dataset.

To demonstrate the advantage of virtual screening on ultra-large molecule dataset, we randomly sampled 1% and 10% of molecules in the 38 million dataset and calculated the docking score distribution of the top 1000 molecules after they were sorted by docking score. We can see that as the number of molecules increases, the docking score of the head molecule is lower, which means that the head molecule we find has a stronger binding ability to the target.

Above all, it took in total 11.27 hours to complete hierarchical virtual screening approach on 38 million molecules with Uni-Dock using a GPU cluster with 100 NVIDIA V100 GPUs, which means that rapid high-throughput virtual screening of extremely large molecular libraries is possible.

3.3 Influence of hyperparameters and ligand properties on speed

Based on the experiments above, the regression equation of running time with respect to the settings of hyperparameters and properties of input ligands is obtained. The equation is first proposed on the basis of our algorithm's time complexity analysis and then refined with huge amount of real data. We finally achieve an accurate prediction of running time ($R^2 = 0.946$).

We first realize that the running time of MC search on GPU roughly goes linear with the product of $nthreads$ and $nsteps$ if other settings remain the same. Moreover, complex and large ligands take more time to mutate, move, and calculate energy in MC searches. We also need a storage on GPU in $O(n^2)$ to save precalculated interactions between each atoms, in which n is the number of atoms of ligand. In each iteration of MC search, operation complexity roughly goes linear with the square of number of atoms of the ligand, since we calculate the interaction between each pair of atoms in the ligand. Furthermore, there are operations outside MC search such as initialization of GPU kernel, result collection and memory copy between host and device. Last but not least, IO on host and constant time in the workflow takes approximately fixed time.

Based on above analysis, we collect running time data from screening experiments using Vina scoring function. The raw data is aggregated for about every 10,000 ligand docking sample since we put them in one Uni-Dock job. We then perform OLS regression and successfully fit running time with $R^2 = 0.946$:

$$time = 7.23 \times 10^{-8} * nthreads * natoms^2 + 1.86 \times 10^{-8} * nthreads * nsteps * natoms^2 - 0.007984 \quad (3)$$

where $time$ stands for the average end-to-end running time of docking one ligand in a batch, $nthreads$ is the number of threads, $natoms$ is the number of atoms of ligand, and $nsteps$ is the number of steps. Note that we have to put enough ligands in one batch to fully utilize the power of GPU and we recommend putting no less than 10,000 ligands in the command line and let Uni-Dock divide them into appropriate batches. Since our dataset mainly consist of ligands with less than 50 atoms, our function might be inaccurate in the case of much heavier ligands.

4 Procedure

This is a short user guide of Uni-Dock which demonstrates a typical docking procedure from setup to result collection.

First we install Uni-Dock and set up the environment. Either building from source or using docker works. Uni-Dock requires an NVIDIA GPU supporting CUDA version $\geq$ 11.2. We have made a good adaptation to common GPUs such as V100 32G and T4 16G and we recommend using machine with more than 10 GB global memory to achieve better performance.

Next, we prepare the input files and pocket information. Uni-Dock can take either *pdbqt* file or grid maps generated by AutoGrid or Vina as protein receptor. The ligands should be in *pdbqt* format as well. Therefore, if input files are in other formats such as SMILES, *sdf*, *pdb* or *mol2*, they should be converted to *pdbqt* format before. The size and position of pocket are provided by the user. The recommended box size is around $22.5\text{Å} \times 22.5\text{Å} \times 22.5\text{Å}$. So far, we support rigid and semi-flexible docking, which means ligands can have rotatable bonds but the receptor must be rigid.

To launch a Uni-Dock job using vina or vinardo scoring function, the most important parameters are as follows:

`--receptor`: filepath of the receptor (*pdbqt* file)

`--gpu_batch`: filepath of the ligands to dock with GPU (*pdbqt* files), process multiple at a time, separated with space

`--search_mode`: computational complexity, choice in `fast`, `balanced`, or `detailed` as described in sec.3.

If there are too many ligands to be put into one command line, it might exceed the maximum command line length of linux. We recommend saving your ligands path in a file (separated by spaces) and use `--ligand_index <ligands path file>` in place of `--gpu_batch` or save your command in a shell script like `run.sh`, and run the command by `bash run.sh`. Here is an example of Uni-Dock command line:

```
unidock --receptor <receptor.pdbqt> \
     --gpu_batch <lig1.pdbqt> <lig2.pdbqt> ... <ligN.pdbqt> \
     --search_mode balanced \
     --scoring vina \
     --center_x <center_x> \
     --center_y <center_y> \
     --center_z <center_z> \
     --size_x <size_x> \
     --size_y <size_y> \
     --size_z <size_z> \
     --dir <save dir>
```

In the case of ad4 scoring function, Uni-Dock only accept AutoGrid map as receptor. Thus we should generate map using AutoGrid4 or similar tools (*e.g.* [28, 29]),and then change command line option `--receptor <receptor.pdbqt>` to `--map <receptor>`.

5 Conclusion

In this work, we proposed Uni-Dock, a GPU-accelerated molecular docking program supporting vina, vinardo and ad4 scoring functions. The significant high efficiency greatly reduced the time and computation resource cost of the high-throughput virtual screening on ultra-large molecular databases.

We implemented parallel MC search threads in CUDA to cover search space, and took the strategy of increasing the number of MC search threads and reducing the number of search steps of each MC search thread. At the same time, we implemented the parallel docking of multiple ligands with dynamically batch processing based on accurate memory space prediction, thereby making full use of the graphical memory space and computing power. In addition, we reduced the amount of data transfer from the CPU to GPU through the migration of the process. We also replaced double-precision float with single-precision float, reducing the memory space of data to achieve more ligands to calculate in parallel.

Studies on large benchmarks demonstrate that Uni-Dock keeps comparable docking accuracy to AutoDock Vina 1.2, and the average time cost of docking each molecule is reduced to 0.1 seconds for vina and vinardo scoring function and to 0.5 seconds for ad4 scoring function on one NVIDIA V100 GPU. We conducted a hierarchical virtual screening against a recently popular drug target, KRAS G12D, on Enamine Diverse REAL drug-like set containing 38.2 million ligands using a GPU cluster of 100 NVIDIA V100 GPUs with Uni-Dock. All calculations completed within 12 hours, which showed Uni-Dock making the high-throughput virtual screening on ultra-large molecular databases more user-friendly.

We will continue to promote the development of Uni-Dock in the following aspects: 1) optimize the performance of Uni-Dock on different kinds of GPUs. 2) optimize the data structure and data stream during the Uni-Dock calculation process, to further reduce the time and memory consumption. 3) optimize the interface of the scoring function, assuring the efficient search of Uni-Dock can be compatible with more scoring functions, such as machine learning-based scoring functions.

6 Acknowledgements

We thank Linfeng Zhang, Yuqing Deng, Denghui Lu, Zhaohan Ding, Yannan Yuan, Xinyu Li, Dongdong Wang, Tao Xu, Junhan Chang, Xiaokuang Bai and many colleagues in DP Technology for their great help in this project.

All experiments are carried out on the cloud platform Bohrium™(`https://bohrium.dp.tech`).

References

[1] W Patrick Walters, Matthew T Stahl, and Mark A Murcko. Virtual screening—an overview. *Drug discovery today*, 3(4):160–178, 1998.

[2] Ashutosh Kumar and Kam YJ Zhang. Hierarchical virtual screening approaches in small molecule drug discovery. *Methods*, 71:26–37, 2015.

[3] Aleix Gimeno, María José Ojeda-Montes, Sarah Tomás-Hernández, Adrià Cereto-Massagué, Raúl Beltrán-Debón, Miquel Mulero, Gerard Pujadas, and Santiago Garcia-Vallvé. The light and dark sides of virtual screening: what is there to know? *International journal of molecular sciences*, 20(6):1375, 2019.

[4] Tiejun Cheng, Xun Li, Yan Li, Zhihai Liu, and Renxiao Wang. Comparative assessment of scoring functions on a diverse test set. *Journal of chemical information and modeling*, 49(4):1079–1093, 2009.

[5] Minyi Su, Qifan Yang, Yu Du, Guoqin Feng, Zhihai Liu, Yan Li, and Renxiao Wang. Comparative assessment of scoring functions: the casf-2016 update. *Journal of chemical information and modeling*, 59(2):895–913, 2018.

[6] Renxiao Wang, Xueliang Fang, Yipin Lu, and Shaomeng Wang. The pdbbind database: Collection of binding affinities for protein- ligand complexes with known three-dimensional structures. *Journal of medicinal chemistry*, 47(12):2977–2980, 2004.

[7] Michael M Mysinger, Michael Carchia, John J Irwin, and Brian K Shoichet. Directory of useful decoys, enhanced (dud-e): better ligands and decoys for better benchmarking. *Journal of medicinal chemistry*, 55(14):6582–6594, 2012.

[8] Jiankun Lyu, Sheng Wang, Trent E Balius, Isha Singh, Anat Levit, Yurii S Moroz, Matthew J O'Meara, Tao Che, Enkhjargal Algaa, Kateryna Tolmachova, et al. Ultra-large library docking for discovering new chemotypes. *Nature*, 566(7743):224–229, 2019.

[9] AN Shivanyuk, SV Ryabukhin, A Tolmachev, AV Bogolyubsky, DM Mykytenko, AA Chupryna, W Heilman, and AN Kostyuk. Enamine real database: Making chemical diversity real. *Chemistry today*, 25(6):58–59, 2007.

[10] Francesco Gentile, Vibudh Agrawal, Michael Hsing, Anh-Tien Ton, Fuqiang Ban, Ulf Norinder, Martin E Gleave, and Artem Cherkasov. Deep docking: a deep learning platform for augmentation of structure based drug discovery. *ACS central science*, 6(6):939–949, 2020.

[11] Andreas Luttens, Hjalmar Gullberg, Eldar Abdurakhmanov, Duy Duc Vo, Dario Akaberi, Vladimir O Talibov, Natalia Nekhotiaeva, Laura Vangeel, Steven De Jonghe, Dirk Jochmans, et al. Ultralarge virtual screening identifies sars-cov-2 main protease inhibitors with broad-spectrum activity against coronaviruses. *Journal of the American Chemical Society*, 144(7):2905–2920, 2022.

[12] Garrett M Morris, Ruth Huey, William Lindstrom, Michel F Sanner, Richard K Belew, David S Goodsell, and Arthur J Olson. Autodock4 and autodocktools4: Automated docking with selective receptor flexibility. *Journal of computational chemistry*, 30(16):2785–2791, 2009.

[13] 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.

[14] Richard A Friesner, Jay L Banks, Robert B Murphy, Thomas A Halgren, Jasna J Klicic, Daniel T Mainz, Matthew P Repasky, Eric H Knoll, Mee Shelley, Jason K Perry, et al. Glide: a new approach for rapid, accurate docking and scoring. 1. method and assessment of docking accuracy. *Journal of medicinal chemistry*, 47(7):1739–1749, 2004.

[15] Thomas A Halgren, Robert B Murphy, Richard A Friesner, Hege S Beard, Leah L Frye, W Thomas Pollard, and Jay L Banks. Glide: a new approach for rapid, accurate docking and scoring. 2. enrichment factors in database screening. *Journal of medicinal chemistry*, 47(7):1750–1759, 2004.

[16] Christoph Gorgulla, Andras Boeszoermenyi, Zi-Fu Wang, Patrick D Fischer, Paul W Coote, Krishna M Padmanabha Das, Yehor S Malets, Dmytro S Radchenko, Yurii S Moroz, David A Scott, et al. An open-source drug discovery platform enables ultra-large virtual screens. *Nature*, 580(7805):663–668, 2020.

[17] Jerome Eberhardt, Diogo Santos-Martins, Andreas F Tillack, and Stefano Forli. Autodock vina 1.2. 0: New docking methods, expanded force field, and python bindings. *Journal of Chemical Information and Modeling*, 61(8):3891–3898, 2021.

[18] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[19] Amr Alhossary, Stephanus Daniel Handoko, Yuguang Mu, and Chee-Keong Kwoh. Fast, accurate, and reliable molecular docking with quickvina 2. *Bioinformatics*, 31(13):2214–2216, 2015.

[20] Nafisa M Hassan, Amr A Alhossary, Yuguang Mu, and Chee-Keong Kwoh. Protein-ligand blind docking using quickvina-w with inter-process spatio-temporal integration. *Scientific reports*, 7(1):1–13, 2017.

[21] Ying Yang, Kun Yao, Matthew P Repasky, Karl Leswing, Robert Abel, Brian K Shoichet, and Steven V Jerome. Efficient exploration of chemical space with docking and deep learning. *Journal of Chemical Theory and Computation*, 17(11):7106–7119, 2021.

[22] Diogo Santos-Martins, Leonardo Solis-Vasquez, Andreas F Tillack, Michel F Sanner, Andreas Koch, and Stefano Forli. Accelerating autodock4 with gpus and gradient-based local search. *Journal of chemical theory and computation*, 17(2):1060–1073, 2021.

[23] Shidi Tang, Ruiqi Chen, Mengru Lin, Qingde Lin, Yanxiang Zhu, Ji Ding, Haifeng Hu, Ming Ling, and Jiansheng Wu. Accelerating autodock vina with gpus. *Molecules*, 27(9):3041, 2022.

[24] David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, 46(D1):D1074–D1082, 2018.

[25] Xiaolun Wang, Shelley Allen, James F Blake, Vickie Bowcut, David M Briere, Andrew Calinisan, Joshua R Dahlke, Jay B Fell, John P Fischer, Robin J Gunn, et al. Identification of mrtx1133, a noncovalent, potent, and selective krasg12d inhibitor. *Journal of Medicinal Chemistry*, 65(4):3123–3133, 2021.

[26] Samuel Genheden and Ulf Ryde. The mm/pbsa and mm/gbsa methods to estimate ligand-binding affinities. *Expert opinion on drug discovery*, 10(5):449–461, 2015.

[27] Nandou Lu and David A Kofke. Accuracy of free-energy perturbation calculations in molecular simulation. i. modeling. *The Journal of Chemical Physics*, 114(17):7303–7311, 2001.

[28] Juan Pablo Arcon, Carlos P Modenutti, Demian Avendaño, Elias D Lopez, Lucas A Defelipe, Francesca Alessandra Ambrosio, Adrian G Turjanski, Stefano Forli, and Marcelo A Marti. Autodock bias: improving binding mode prediction and virtual screening using known protein–ligand interactions. *Bioinformatics*, 35(19):3836–3838, 2019.

[29] Pradeep Anand Ravindranath, Stefano Forli, David S Goodsell, Arthur J Olson, and Michel F Sanner. Autodockfr: advances in protein-ligand docking with explicitly specified binding site flexibility. *PLoS computational biology*, 11(12):e1004586, 2015.