2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing

# FRODRUG: a virtual screening GPU accelerated approach for drug discovery

Santiago García, E. Ramírez-Aportela, J.I. Garzón, Pablo Chacón
Biological Chemical Physics Department
Rocasolano Physical Chemistry Institute (CSIC)
Madrid, Spain
pablo@chaconlab.org

Antonio S. Montemayor, Raúl Cabido
Computer Science Department
Universidad Rey Juan Carlos (URJC)
Móstoles, Madrid, Spain

***Abstract*— The procedure for screening large databases of small chemical compounds to select likely drug candidates by computational means is very time demanding. Here, we present and evaluate a new method for virtual screening (VS) that combines the efficiency of spherical harmonic approximations to accelerate the rotational part of a docking search with multicore and GPU parallelism. To validate these novel parallel algorithms, we used standard benchmark cases. The obtained results are comparable to those generated via state-of-the-art VS docking approximations, but with a considerable gain in efficiency. GPU implementation speedups of more than 30-fold with respect to a single core CPU were achieved, reducing the docking time for a single ligand to only 50 milliseconds. The achieved efficiency and the accuracy on standard blind benchmarks demonstrate the applicability and robustness of this approach.**

***Keywords—Drug discovery; Virtual screening; Docking; GPU;***

## I. INTRODUCTION

The need to develop safe and innovative drugs shifts the focus toward improving the early phases of drug discovery. In this context, the selection of likely drug candidates from extensive compound libraries by computational means represents a fast and an especially cost-effective approach. In particular, receptor-based virtual screening (VS) is becoming a key part of the drug discovery process. VS relies on docking algorithms to perform computational screening. Protein-ligand docking methods attempt to identify the optimal positions, orientations and conformations of a small compound (ligand) with respect to a target protein with a known 3D structure (receptor). Typically, VS approaches are implemented via a two-step process. First, the whole ligand database is filtered using efficient but low discrimination docking power approximations. Then, the top scoring compounds are passed to more accurate and demanding docking screening algorithms. At the end of this process, a limited number of potential binding compounds (from hundreds to a few thousand) are ready to be experimentally tested. There are many docking programs currently available (Glide[1, 2], AutoDock Vina[3], GOLD[4], ICM[5], MS-DOCK[6], VSDMIP[7], VSDocker[8], PhDock[9], FlexX[10], Surflex[11], BINDSURF[12]) for implementing this two-step protocol, but no single program has yet emerged that outperforms all of the others in all cases. Most of these programs are able to employ multithreading capabilities on multicore machines. Besides, BINDSURF can perform the docking process using CUDA devices.

Despite significant progress in docking technologies, the largest challenge to be addressed is to combine highly accurate binding predictions with speed and sampling power. Current sampling and scoring approaches take from seconds to minutes to process a single compound, even at the initial early screening step. Thus, the efficiency and accuracy of the current approaches must be improved to handle the avalanche of new drug targets and the current necessity to perform a fully automated database screening of libraries that contains millions of compounds.

Here, we focus on the first and most computationally challenging stage of docking, which consists of rigid-body orientational sampling of a small ligand with respect to a fixed receptor molecule, while a docking scoring function is maximized. The 6D sampling space of the relative orientations between the ligand and receptor is very large. In fact, more than $10^6$ relative poses can be evaluated. Moreover, the number of molecules in the chemical libraries that are currently used for VS is greater than $10^6$. Thus, a very efficient docking tool is mandatory. To ease this computational bottleneck, we adapted a procedure to the protein-ligand docking problem. Our group previously developed this procedure for addressing related docking problems [13, 14]. This algorithm, which is based on a suitable parameterization of the 3D rotation group with spherical harmonics (SH), significantly speeds up docking by accelerating the rotational part of the search [15, 16]. This adaptation for protein-ligand docking involved deriving new mathematical expressions, and hence, it is a novel docking methodology, which we have designated FRODRUG. Moreover, to boost the efficiency of the VS process even more, we ported this methodology to High Performance Computing (HPC) facilities. In this paper, we report two parallelization strategies for FRODRUG. One version is designed for many core systems/clusters, which is based on MPI (Message Passing Interface), and the other is applied for GPUs and uses NVIDIA CUDA [17]. We discovered that the latter approach provides superior efficiency, with speedups greater than 30-fold being achieved

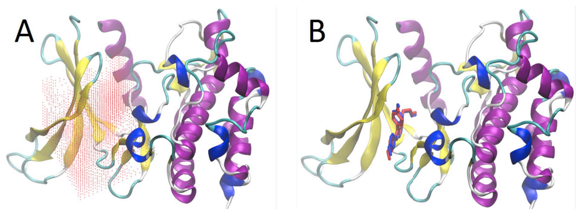

Fig. 1. Illustrative case of the Chk1 kinase (PDB id 2br1). A) The translational search points explored by the docking site are plotted as red points. Such points have been generate by sampling uniformely (every 1Å) the docking search box and removing those in which a potential ligand deeply penetrates into the receptor or in which the ligand molecule is not even in contact with the receptor. B) Superimposition of the ligand's best docking (in blue) pose, as found in a VS search with the experimentally observed bound ligand conformation.

with respect to a single-core CPU. The GPU implementation of FRODRUG allows a ligand to dock in only 50 milliseconds on a NVIDIA GTX 680 graphics processor. This speed makes it possible to conduct a virtual screening campaign for one million compounds using a single GPU card in slightly more than half a day.

## II. Methods

The virtual screening process involves prediction of the binding conformations of a large database of chemical compounds to identify possible new drug candidates. Such computational predictions are performed using a docking algorithm based on the 3D atomic structures of the target receptor and the ligands. Because we know the approximate location of the binding site of our target protein receptor, we can reduce the size of the docking search to a generous box around this site. In this box of ~15 x 15 x 15 Å$^3$ (see the illustrative example in figure 1), we define several translational positions to determine the relative rigid-body orientation of mobile ligands while maximizing a scoring function. Thus, in our docking algorithm, the three translational degrees of freedom (DOF) are simply scanned uniformly in the box search, while the remaining three rotational DOF will be accelerated, as described below. The selection of the sampling of the rotational and translational search represented a practical solution that was a compromise between exhaustiveness and efficiency. This rotational sampling depends on the bandwidth (BW) used in the harmonic representation (360º/(BW·2)). The sampling values employed were 1 Å and 11.25º (BW=16), which corresponds to scanning more than 2,500 translational points and more than 10,000 distinct rotations. Thus, 25 million poses will be evaluated per ligand. Considering that a VS campaign requires the testing of several thousand to a million ligands, the efficiency of the docking method is critical.

### *A. The docking algorithm*

For protein ligand docking, we adapted a successful procedure developed by our group based on spherical harmonics for other bioinformatics problems, such as protein-protein docking [14] and multi-resolution fitting [13]. Briefly, the interaction energy of the receptor and ligand molecules can be expressed on the unit sphere in terms of spherical harmonic functions and their corresponding coefficients, $\hat{f}_{lm}(r)$ and $\hat{g}_{lm}(r)$. Then, the scoring to be maximized during docking can be calculated based on the correlation of such interaction energy terms. In this way, for a given translation, the correlation function for all of the rotations can be calculated very rapidly by following an inverse Fourier transform:

$$E(R) = FT^{-1}_{m,h,m'}\left[\sum_{l} d^{l}_{mh} d^{l}_{hm'} I^{l}_{mm'}\right] \qquad (1)$$

where

$$I^{l}_{mm'} = \int_{0}^{\infty} \hat{f}_{lm}(r) \cdot \overline{\hat{g}_{lm}(r)} \cdot r^2 \cdot dr \qquad (2)$$

The subscripts and superscripts run for the degree and order of the SH and $d^l$ are precomputed coefficients that define the matrix elements of the irreducible representations of the 3D rotational group.

We tested several types of energy potential, but we found optimal results using a simple Lennard-Jones 6-12 potential, followed by rescoring with a knowledge-based scoring function consisting of distance-dependent pair potentials similar to DSX [18]. The details of the energy terms and procedures will be given elsewhere.

### *B. Virtual screening*

The VS procedure loads every ligand of the compound database into FRODRUG, retrieves the results and ranks them. The performance of the VS approach relies on the docking algorithm as well as the application of the most cost-effective data parallel architecture. Splitting an exhaustive search into translational and rotational parts is very convenient for the parallel design. Efficient rotational searches of independent translational scanned points can be computed in parallel.

## III. Parallel Implementations

We explored two parallelization strategies. To this end, an MPI extension was developed to perform docking on either CPUs or CUDA devices over heterogeneous clusters. We employed asynchronous functions of the MPI API to overlap data transferences with computation. This MPI extension follows a master/slave architecture, where there is one master process that distributes the ligands among the other processes and collects the results. The master sends a new ligand while the slave is still docking the previous one to avoid idle situations. At the end of this process, the master sorts the results, and the ligands are ranked according to their docking correlations. The highest-scoring ligands with their corresponding translational and rotational orientations are then saved to disk.

The GPU implementation is divided into four different steps.

- *Integral Kernel:* Computes the integral defined as eq. 2.
- *Correlation Kernel*: Computes correlation volumes, i.e., the expression inside within the brackets of eq. 1.
- *FFT Computation*: Computes the inverse Fourier transform from the correlation volumes. To this end,

we employed the cuFFT library [19] provided by NVIDIA.

- *Search Kernel*: Searches the ligand orientation with the highest docking score in the correlation volume obtained in the previous inverse Fourier transform step.

Some of the optimizations of our kernels are based on the warp size. A warp is a group of threads that are executed physically at the same time. Its size is a constant that is fixed at 32 threads per warp for every CUDA specification. Our implementation requires the use of CUDA devices that have a computational capability of 2.0 or above (Fermi architecture and later) due to the number of registers and threads per block that this specification provides. Since the Fermi architecture was released, CUDA devices have included a small L1 cache on their chips. The size of these caches can be configured per kernel at run time such that there are 16 KB or 48 KB on the Fermi or 16 KB, 32 KB or 48 KB on the Kepler architecture (the default is 48 KB of shared memory). Below, we summarize the CUDA implementation of the kernels.

## A. *Integral kernel*

This kernel computes the integral (eq. 2) for every translational point within the binding site of a given ligand. The length of the integrals depends on the bandwidth used (BW·(4·BW·BW-1)/3). The integral complex data is separated into two arrays, splitting the real and imaginary parts to speed up the read/write operations. The storage of the integral data is done in a transpose way to optimize the input read for the next kernel.

The process of computing the integral is illustrated in figure 2. The left part of the figure shows a block of threads in grey that compute data from the first 32 integrals. Each thread reads its corresponding part from $\hat{f}_{lm}(r)$ and $\hat{g}_{lm}(r)$ and stores the computed integral in shared memory. A 32 x 32 (+1) block of shared memory is allocated for this purpose. An extra column is added to the shared memory block to avoid bank conflicts (see figure 3 for details). We choose a 32 x 16 thread block matching the X-axis with the warp size. The set of 32 integrals was computed in two steps because the Y-axis only contained 16 units. This arrangement was chosen for reasons of optimization because the hardware performed better with smaller thread blocks, which results from the amount of register per thread reducing the occupancy. On the right side of the figure, the process of writing the data to the global memory is shown. The warps read the data from the shared memory in a column-wise manner and store the data linearly in the global memory. This process is again performed in two iterations.

To cover all of the integral blocks, we extended the X- and Y-axes to conform to the grid and to process all of the translational points.

## B. *Correlation kernel*

This kernel performs the operation shown within brackets in the main equation (eq. 1). In principle, we will require a matrix of 32 x 32 x 32 values. However, because it will be used as the input of a complex-to-real Fourier transform, half of the volume can be omitted due to "Hermitian" redundancy. Moreover, only two quadrants of the volume must be computed because the other two quadrants can be mirrored based on the symmetry of the harmonic coefficients.

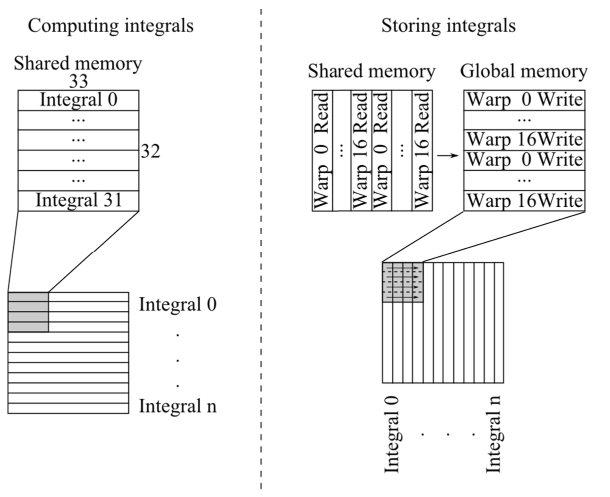

Fig. 2. The figure illustrates the process of generating the integral for every translational point. The data for each integral is computed in two steps in a row-wise way in shared memory, then it's transposed before it's written to global memory, also in two steps.

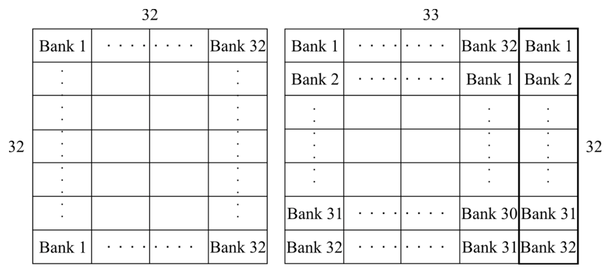

Fig. 3. In the left side we have a shared memory block of 32x32. Threads can read data horizontally in parallel as they access 32 different banks. Reading a column is done secuentially, as every thread access the same bank. Adding an extra column solves the problem, as shown in the right side. Threads can read a row or a column in parallel as they access always different banks.

In figure 4, we can see the configuration selected for the thread blocks. As above, we have set a length of 32 threads in the X-axis, matching warp size, and 16 threads in the Y-axis. Each block includes a shared memory section of 32 x 33 floats for storing the correlation data. Again, this extra column avoids bank conflicts. The zone indicated in grey corresponds to threads of the same warp, which shows that the data is written to the shared memory in a column-wise manner. Each warp computes the complex value for the same X-index of 32 different correlation volumes, and this index is determined by the Y index of the thread block. Both parts of the complex number are computed by the same thread.

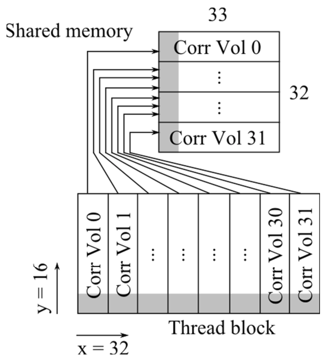

Fig. 4. Every warp of the thread block computes data from 32 different correlation volumes and each warp stores its data in two columns of the shared memory block.

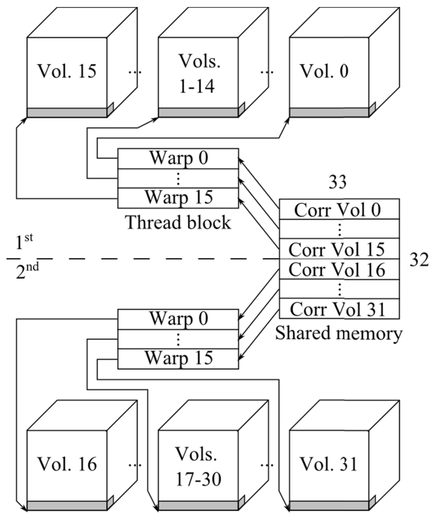

Fig. 5. Each thread block stores its data in two steps, storing half of the data in each one. The first one is in the top side and the second in the bottom.

The Z and Y indexes of the thread block within the grid determine which X row of the correlation volumes is computed. Figure 5 shows a thread block that fills the X row and corresponds to the indexes Z=0 and Y=0 of 32 different volumes. This process is performed in two steps. The top side of the figure shows the first step, where the thread block stores the data for the first 16 volumes. The remaining values are stored in the second step, as shown in the bottom of the figure. The remaining correlation volumes covered the grid along the X-axis.

### C. *Search kernel*

The inverse FFT of the complex matrix calculated in the previous kernel yields a correlation volume of 32 x 32 x 32. This volume stores all of the docking scores, sampling the rotational space twice over the bandwidth used, which amounts to 11.25º (360º/(16·2)). Within this volume, the kernel searches slice-by-slice for the highest correlation values, as illustrated in figure 6. We set the thread block size to 32 x 32 to cover a single slice of the volume. Interpolating the correlation values between the closest neighbors enhances the accuracy of each of the high correlation peaks. Moreover, only half of the correlation volume has to be scanned due to the redundancy of the rotational space, using Euler angles (for full coverage, it is sufficient to sample the three Euler angles that define a given rotation, from 0-360, 0-180 and 0-360 degrees, respectively).

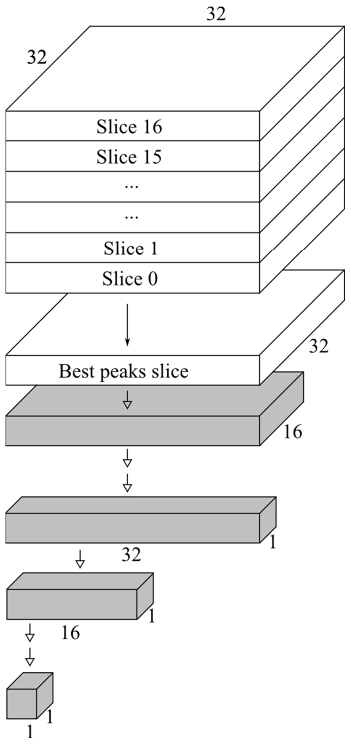

Fig. 6. In first place the slices are scanned one by one to get a 32 x 32 slice with the best peaks, then a parallel search is done over the slice to get the best value.

The second step of the search is illustrated at the bottom of the figure. Once we obtain all of the high-correlation peaks for a slice, a parallel step-wise reduction is performed to find the best peak. At the end of this process, a single best docking pose will be obtained for every translational point, copied to RAM and ranked by the CPU.

## IV. RESULTS

Blind docking experiments were conducted for the set of protein-ligand test cases obtained from the Astex Diverse Set [20], which is a published collection of 85 protein-ligand crystal structures extracted from the Protein Data Bank and specifically selected to evaluate the performance of the docking algorithms. Based on the rotated and translated versions of these ligands, we determined whether the docking results identified the correct binding pocket, as defined by the crystal structure of the protein/ligand complex.

Our results indicated that FRODRUG correctly identifies 93% (79 out of 85) of the native poses. These results are comparable to those obtained with the most widely used docking programs. For example, in the original study of the Astex benchmark using GOLD [20], a performance of 80% of complexes was reported, with an RMSD $\leq 2$ Å. A more recent version of this program increased the percentage to 93% [21].

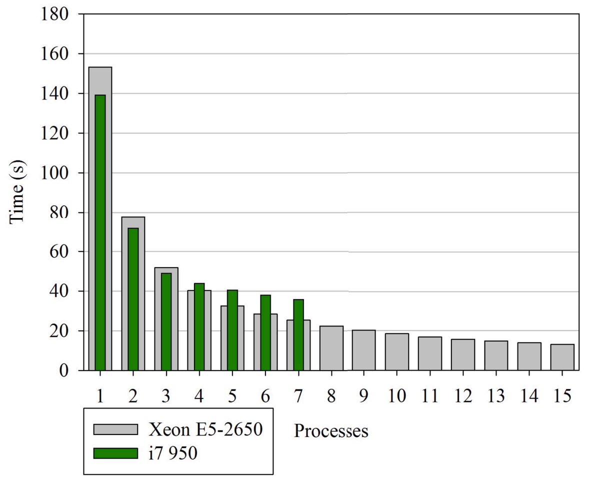

Fig. 7. Execution times for the Intel i7 from 1 to 7 processes and for the Intel Xeon from 1 to 15 processes.

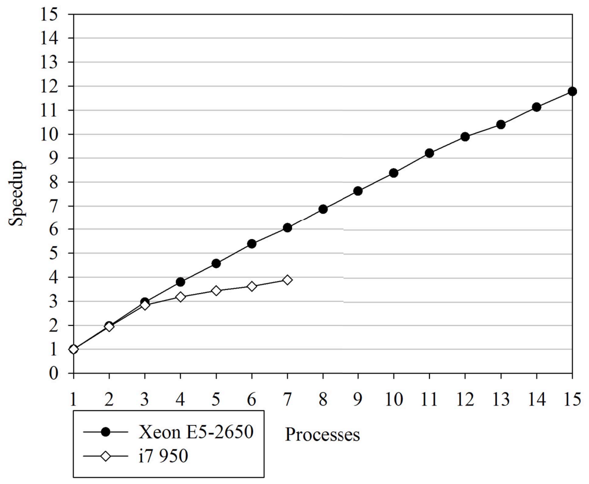

Fig. 8. Speedup chart of for the Intel i7 from 1 to 7 processes and for the Intel Xeon from 1 to 15 processes.

A comparison of the performance obtained with the Glide[1, 2], ICM[5], PhDock[9], FlexX[10] and Surflex[11] docking tools revealed rates of successfully docked poses of 70–90% [22].

To further test the parallel implementations, several standard tests were conducted using different machines with the hardware described in TABLE I. A fair comparison of such architectures is really hard, however in terms of cost, they are around 1,000$, 400$ and 300$ for Xeon, GTX 680 and i7, respectively.

The CUDA code of the software was compiled for computational capabilities of 2.0 and 3.0 using the NVIDIA CUDA 5.0 compiler. The remaining code was compiled to take maximum advantage of the capabilities of each processor. The i7 uses the SSE4.2 instruction set extension, and the Xeon uses the AVX instruction set extension. The library used for testing the MPI implementation was MPICH2 1.4.1p1.

## A. *Scalability test*

We used the two CPUs to test the scalability of FRODRUG. The Xeon E5-2650 has 8 cores and can run up to 16 threads simultaneously due to HTT (Hyper Threading Technology). The i7 950 processor is less powerful, running up to 8 threads with HTT. Both figures 7 and 8 show only the number of MPI processes that compute ligands; the master process is omitted. These figures provide the docking execution time and the corresponding speedup, respectively. The time indicated is the average time for docking a given protein against all 85 ligands.

Processing a single docking test of the Astex benchmark takes 153.2 seconds on average on the Xeon employing a single process (figure 7). The required time is reduced to 13.0 seconds when using 15 processes. On the other hand, the i7 takes 139.1 seconds on average, using only 1 process, and the time is reduced to 35.7 seconds with 7 processes.

Figure 8 shows that the algorithm scaling is nearly linear when using 4 processes or fewer in the Xeon E5-2650. We reach a speedup of 11.8 times using 15 processes compared to the time required with only 1. The algorithm scales linearly with the i7 950 when using 3 processes or fewer for computing. The scalability is fairly poor with additional processes. The response of this processor is adequate because it has only 4 physical cores. Globally, this algorithm scales very well in many core systems when hardware limits are not exceeded.

## B. *Efficiency tests with GPUs*

The first efficiency test is to run the Astex benchmark on the NVIDIA GTX 680 card and to compare the execution times with previous CPU results. In this test, the GPU version of the algorithm requires an average of 4.6 seconds to dock 85 ligands in a single protein. Figure 9 shows the speedup obtained using the Intel i7 950. The GPU performs 30.5 times faster compared to executing the algorithm using 1 process with this processor. We also obtained a speedup of 33.6x against the CPU version using 1 process with the Intel Xeon E5-2650 (figure 10). The efficiency on heterogeneous systems increase as the number of MPI processes grows; it is limited mainly by the available hardware and the amount of molecules to dock.

We conducted another test to measure the efficiency of the parallel docking algorithm. In this test, we checked the performance of the docking of the Cyclooxygenase-2 protein

TABLE I HARDWARE USED FOR TESTING PARALLEL FRODRUG.

| Hardware | Cores | Threads | Cache | Clock |
|---|---|---|---|---|
| Intel Xeon E5-2650 | 8 | 16 | 20 MB | 2 GHz |
| Intel i7 950 | 4 | 8 | 8 MB | 3.06 GHz |
| NVIDIA GTX 680 | 1,536 | - | * | 1.06 GHz |

*THE CACHE MEMORY FOR GPUS WITH A KEPLER (GTX 680) ARCHITECTURE CAN BE SELECTED AT RUN TIME FOR EACH KERNEL TO BE 16, 32 OR 48 KB.

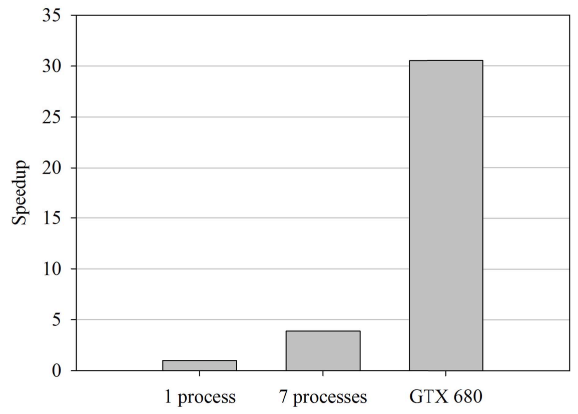

Fig. 9. The i7 using 1 process is taken as base at 1x speedup. Running the CPU algorithm with the same processor with 7 processes has a speedup of 3.89x, and we reach a speedup of 30.52x with the GTX 680.

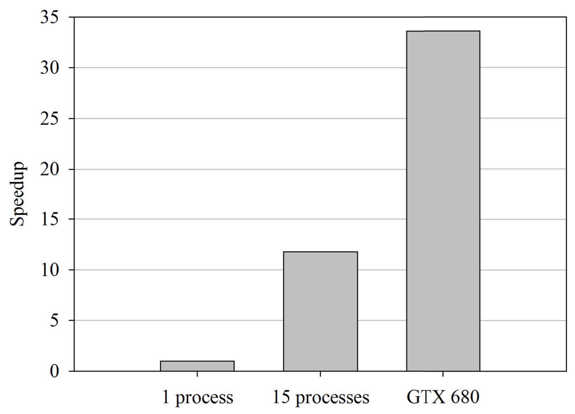

Fig. 10. The Xeon using 1 process is taken as base at 1x speedup. Running the CPU algorithm with the same processor with 15 processes has a speedup of 11.78x, and we reach a speedup of 33.61x with the GTX 680.

TABLE II. DOCKING TIMES FOR 13,716 LIGANDS WITH THE CYCLOOXYGENASE-2 PROTEIN

| Method | Total time | Time per point |
|---|---|---|
| i7 950 (4 processes) | 2 h 1 m 14 s | 221.8 µs |
| GTX 680 (1 card) | 11 m 16 s | 20.6 µs |
| GTX 680 (2 cards) | 5 m 53 s | 10.8 µs |

(PDB id 1cx2) with a database of 13,715 ligands. These ligands were taken from the Directory of Useful Decoys (DUD) [23]. The docking algorithm selects 2,391 points within the binding site. For this test, we employed three configurations: the i7 950 CPU using 4 processes (1 master, 3 slaves), a single NVIDIA GTX 680 card and a two GTX 680 card setup with the MPI extension over two machines. A compilation of the results of this test is shown in TABLE II. This table shows both the total time for docking the whole set of ligands over the entire binding site and the average time that it takes to compute every rotation for any ligand at any translational point. This test yields a speedup of 10.7x with a single card and 20.5x with two cards compared to the CPU version employing 4 processes with the i7 950 processor.

### C. *Accuracy test*

We ran all of the test cases of the Astex to assess the accuracy of the CPU (Intel i7) and GPU (GTX 680) implementations. The result of each blind test is a scored list of solutions with their corresponding locations and rotations. This list is evaluated by measuring the root mean square deviation (RMSD) and the rank of the solutions. The RMSD is expressed in Å and measures the distance error between the ligand atoms of the docked solutions and the correct solution provided by the benchmark. The rank is the position of the first correct solution found in the list. A correct solution is defined as a ligand with an RMSD that is less than or equal to 2 Å.

The correct ligand was found to be within the top 10 solutions in 94.1% of the cases when using the CPU implementation. This number decreased to 92.9% for the GPU implementation. This minimal difference resulted because the CPU version takes the three best solutions per translational point, whereas the GPU version takes only the best solution for optimization reasons. The consideration of more peaks yields to a more sequential and inefficient GPU search.

When we examined the RMSD in more detail, the difference between platforms was more apparent. Using the CPU, 82.8% of the best solutions showed an RMSD below 1 Å, while the percentage of highly accurate solutions dropped to 74.5% with the GPU. The GPU computes the location and translation of the ligands slightly less accurately than the CPU.

## V. CONCLUSIONS

The process of docking millions of ligands against a single target macromolecule represents a very demanding computational problem. In this paper, we have presented a parallel version of a docking algorithm for CUDA devices and an MPI extension for exploiting multicore architectures. We showed using the MPI version that the scalability of the algorithm is very good in this kind of many-core system because of the way that it is implemented. We also obtained very satisfying results for the GPU version of the algorithm, reaching a speedup of up to 10.7x using an NVIDIA GTX 680 compared to a 4-core Intel i7 950 CPU. This GPU implementation allows the evaluation of $10^6$ distinct docking poses for a single ligand at the binding site in only 50 milliseconds. Extrapolating this efficiency into a complete VS campaign, our implementation computes the screening of one million compounds using a single GPU card in slightly more than half a day, while this process takes several days using the current state-of-the-art methods on a CPU cluster.

## REFERENCES

[1] R. A. Friesner, J. L. Banks, R. B. Murphy, T. A. Halgren, J. J. Klicic, D. T. Mainz*, et al.*, "Glide: A New Approach for Rapid, Accurate Docking and Scoring. 1. Method and Assessment of Docking Accuracy," *Journal of Medicinal Chemistry,* vol. 47, pp. 1739-1749, 2004.

[2] T. A. Halgren, R. B. Murphy, R. A. Friesner, H. S. Beard, L. L. Frye, W. T. Pollard, *et al.*, "Glide: A New Approach for Rapid, Accurate Docking and Scoring. 2. Enrichment Factors in Database Screening," *Journal of Medicinal Chemistry,* vol. 47, pp. 1750-1759, 2004.

[3] O. Trott and A. J. Olson, "AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading," *Journal of computational chemistry,* vol. 31, pp. 455-461, 2010.

[4] G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor, "Development and validation of a genetic algorithm for flexible docking," *Journal of molecular biology,* vol. 267, pp. 727-748, 1997.

[5] R. Abagyan, M. Totrov, and D. Kuznetsov, "ICM—a new method for protein modeling and design: applications to docking and structure prediction from the distorted native conformation," *Journal of computational chemistry,* vol. 15, pp. 488-506, 1994.

[6] N. Sauton, D. Lagorce, B. O. Villoutreix, and M. A. Miteva, "MS-DOCK: Accurate multiple conformation generator and rigid docking protocol for multi-step virtual ligand screening," *BMC bioinformatics,* vol. 9, p. 184, 2008.

[7] R. Gil-Redondo, J. Estrada, A. Morreale, F. Herranz, J. Sancho, and A. R. Ortiz, "VSDMIP: virtual screening data management on an integrated platform," *Journal of computer-aided molecular design,* vol. 23, pp. 171-184, 2009.

[8] N. D. Prakhov, A. L. Chernorudskiy, and M. R. Gainullin, "VSDocker: a tool for parallel high-throughput virtual screening using AutoDock on Windows-based computer clusters," *Bioinformatics,* vol. 26, pp. 1374-1375, 2010.

[9] D. Joseph-McCarthy, B. E. Thomas, M. Belmarsh, D. Moustakas, and J. C. Alvarez, "Pharmacophore-based molecular docking to account for ligand flexibility," *Proteins: Structure, Function, and Bioinformatics,* vol. 51, pp. 172-188, 2003.

[10] M. Rarey, B. Kramer, T. Lengauer, and G. Klebe, "A fast flexible docking method using an incremental construction algorithm," *Journal of molecular biology,* vol. 261, pp. 470-489, 1996.

[11] A. N. Jain, "Surflex: Fully Automatic Flexible Molecular Docking Using a Molecular Similarity-Based Search Engine," *Journal of Medicinal Chemistry,* vol. 46, pp. 499-511, 2003.

[12] 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, p. S13, 2012.

[13] J. I. Garzon, J. Kovacs, R. Abagyan, and P. Chacon, "ADP_EM: fast exhaustive multi-resolution docking for high-throughput coverage," *Bioinformatics,* vol. 23, pp. 427-33, Feb 15 2007.

[14] J. I. Garzon, J. R. Lopez-Blanco, C. Pons, J. Kovacs, R. Abagyan, J. Fernandez-Recio, *et al.*, "FRODOCK: a new approach for fast rotational protein-protein docking," *Bioinformatics,* vol. 25, pp. 2544-51, Oct 1 2009.

[15] J. A. Kovacs, P. Chacon, Y. Cong, E. Metwally, and W. Wriggers, "Fast rotational matching of rigid bodies by fast Fourier transform acceleration of five degrees of freedom," *Acta Crystallogr D Biol Crystallogr,* vol. 59, pp. 1371-6, Aug 2003.

[16] J. A. Kovacs and W. Wriggers, "Fast rotational matching," *Acta Crystallogr D Biol Crystallogr,* vol. 58, pp. 1282-6, Aug 2002.

[17] *CUDA 5.0 C Programming guide*. Available: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[18] G. Neudert and G. Klebe, "DSX: a knowledge-based scoring function for the assessment of protein-ligand complexes," *J Chem Inf Model,* vol. 51, pp. 2731-45, Oct 24 2011.

[19] *NVIDIA cuFFT*. Available: https://developer.nvidia.com/cufft

[20] M. J. Hartshorn, M. L. Verdonk, G. Chessari, S. C. Brewerton, W. T. Mooij, P. N. Mortenson, *et al.*, "Diverse, high-quality test set for the validation of protein-ligand docking performance," *J Med Chem,* vol. 50, pp. 726-41, Feb 22 2007.

[21] J. W. Liebeschuetz, J. C. Cole, and O. Korb, "Pose prediction and virtual screening performance of GOLD scoring functions in a standardized test," *J Comput Aided Mol Des,* vol. 26, pp. 737-48, Jun 2012.

[22] J. B. Cross, D. C. Thompson, B. K. Rai, J. C. Baber, K. Y. Fan, Y. Hu, *et al.*, "Comparison of several molecular docking programs: pose prediction and virtual screening accuracy," *J Chem Inf Model,* vol. 49, pp. 1455-74, Jun 2009.

[23] N. Huang, B. K. Shoichet, and J. J. Irwin, "Benchmarking Sets for Molecular Docking," *J. Med. Chem.,* vol. 49, pp. 6789-6801, 2006.