



Article

A Review on Parallel Virtual Screening Softwares for High-Performance Computers

Natarajan Arul Murugan^{1,*}, Artur Podobas¹, Davide Gadioli², Emanuele Vitali², Gianluca Palermo²
and Stefano Markidis^{1,*}

¹ Department of Computer Science, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, SE-10044 Stockholm, Sweden; podobas@kth.se

² Dipartimento di Elettronica, Infomazione e Bioingegneria, Politecnico di Milano, 20133 Milano, Italy; davide.gadioli@polimi.it (D.G.); emanuele.vitali@polimi.it (E.V.); gianluca.palermo@polimi.it (G.P.)

* Correspondence: arul.murugan@iiitd.ac.in or murugan@kth.se (N.A.M.); markidis@kth.se (S.M.)

† Present address: Department of Computational Biology, Indraprastha Institute of Information Technology (IIIT-Delhi), New Delhi 110020, India.

Abstract: Drug discovery is the most expensive, time-demanding, and challenging project in biopharmaceutical companies which aims at the identification and optimization of lead compounds from large-sized chemical libraries. The lead compounds should have high-affinity binding and specificity for a target associated with a disease, and, in addition, they should have favorable pharmacodynamic and pharmacokinetic properties (grouped as ADMET properties). Overall, drug discovery is a multivariable optimization and can be carried out in supercomputers using a reliable scoring function which is a measure of binding affinity or inhibition potential of the drug-like compound. The major problem is that the number of compounds in the chemical spaces is huge, making the computational drug discovery very demanding. However, it is cheaper and less time-consuming when compared to experimental high-throughput screening. As the problem is to find the most stable (global) minima for numerous protein–ligand complexes (on the order of 10^6 to 10^{12}), the parallel implementation of in silico virtual screening can be exploited to ensure drug discovery in affordable time. In this review, we discuss such implementations of parallelization algorithms in virtual screening programs. The nature of different scoring functions and search algorithms are discussed, together with a performance analysis of several docking softwares ported on high-performance computing architectures.

Keywords: computational drug discovery; virtual screening; molecular docking; chemical space; parallelization; high-performance computers and accelerators



Citation: Murugan, N.A.; Podobas, A.; Vitali, E.; Gadioli, D.; Palermo, G.; Markidis, S. A Review on Parallel Virtual Screening Softwares for High-Performance Computers. *Pharmaceuticals* **2022**, *15*, 63.

<https://doi.org/10.3390/ph15010063>

Academic Editor: Osvaldo Andrade Santos-Filho

Received: 21 November 2021

Accepted: 28 December 2021

Published: 4 January 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Drug discovery is one of the most highly challenging, time-consuming and expensive projects in the healthcare sector. The usual time involved in bringing a drug from basic research to market is 12–16 years, and the cost associated is about 2.5 billion dollars [1–4]. To meet one of the EU Sustainable Development Goals [5] aimed at the good health and wellbeing for everyone, drugs should be made available to common people at an affordable price, and the current protocols in drug development need to be redesigned to make the discovery process economically sustainable. One of the most promising techniques to accelerate the drug discovery process, and to make it more cost-effective, is to perform in silico virtual screening, and to exploit the computational power of large high-performance computing (HPC) systems.

One of the major contributing factors to the cost and time associated with the discovery is that it has been reported [6,7] that only one in 10,000 compounds subjected to research and development (R&D) turns out to be successful. The drug discovery involves various steps such as target discovery, lead identification, lead optimization, ADMET (absorption, distribution, metabolism, excretion, toxicity) properties optimization, and clinical trials [8].

Once a valid target is known for a disease, compounds from different chemical libraries are subjected to high-throughput screening against this target. If the number of compounds used for screening can be narrowed down to a few hundred, the cost and time associated with a drug discovery process can be drastically reduced. Using computational approaches, many of the steps involved in the drug discovery projects can be made to be cost-effective and less time-consuming. For example, in the case of protein tyrosine phosphatase-1B [9], the experimental high-throughput screening of a chemical library with 400,000 compounds yielded a success rate of 0.021% in identifying the ligands that can inhibit the enzyme with IC_{50} values less than 100 μ M. However, with the use of a preliminary screening phase using a computational approach, the success rate turned out to be 34.8% starting from a chemical library of 235,000 compounds.

To summarize, the experimental high-throughput screening is not suitable to deal with modern chemical spaces as they are composed of up to billions of molecules. To solve this problem, it is common to use computational approaches on HPC systems. In this review, we highlight various currently available implementations of virtual screening softwares suitable for high-performance computers. Below, we provide general introduction to virtual screening (VS) problems and discuss the possibilities for the parallelization so that it can be effectively implemented for computing facilities offered by HPCs.

The paper is organized as follows. Section 2 introduces the computational VS with details on scoring functions and search algorithms. Section 2.2 presents details on the major breakthroughs obtained in VS and Section 3 presents the main parallelization techniques used in VS and why they target HP systems. In Section 4, we provide an overview about the implementations of different VS softwares. Finally, we discuss the opportunities offered by reconfigurable architectures such as FPGAs.

2. The In Silico Virtual Screening Problem

In general, the computational approaches for molecular docking have two main components: sampling and scoring. Sampling refers to generation of various conformations and orientations for the ligand within a target binding site (defined usually by a grid box). Scoring refers to evaluating the binding/docking energies for various configurations of the ligand within the binding site. The most stable configuration of the ligand is referred to as binding pose. The VS protocol where molecular dockings are carried out for all the ligands from a chemical library includes a third component referred to as ranking, where different ligands are ranked with respect to their binding potential. Overall, the VS identifies the ligand with the topmost binding affinity (which is based on the docking energies of different ligands) for a given biomolecular target. In addition, the most stable binding mode/pose for each of the ligands within the binding site is found (which is based on the relative docking energies of different configurations of the same ligand). Figure 1 shows the general workflow of computer-aided drug discovery where the VS approach is used to identify lead compounds. It shows the steps involved in the binding pose identification of ligands within the binding site, and the ranking of different ligands is subsequently carried out to identify the lead compounds. Most of the VS schemes do not include the flexibility for the target protein, and only the sampling over translational, rotational, and torsional degrees of freedom of ligand is accounted for.

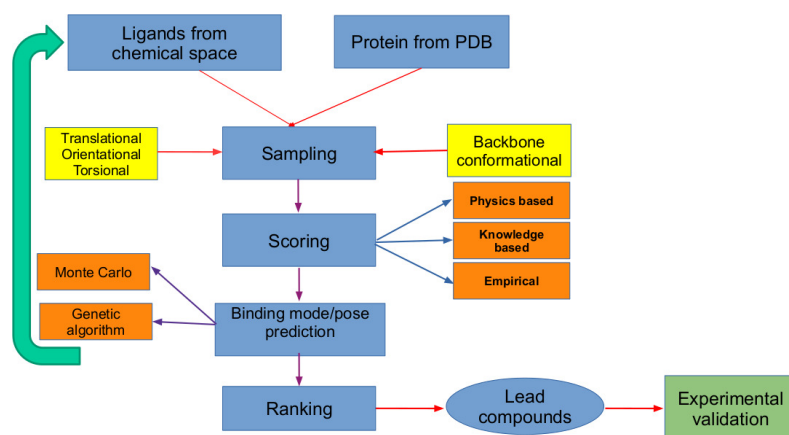


Figure 1. Workflow for computer-aided drug discovery where the lead compounds are identified using VS. It shows that the various configurations of the ligands within binding sites are generated using sampling, which is scored using a scoring function to identify the most stable binding mode/pose. The docking energies of the most stable configurations of all the ligands are used in ranking them to identify a list of lead compounds which are taken for further experimental validation. As the sampling and scoring need to be performed for all the ligands in the chemical library, these steps are shown within a loop.

2.1. Scoring Functions

The reliability and accuracy of the scoring functions used for screening compounds are the most important parameters that dictate the success rate of the computational screening approaches. The scoring functions are mostly defined to be proportional to the binding affinity of the ligand towards a target. The scoring functions are often classified as *physics-based*, *knowledge-based*, and *empirical*.

- (i) *Physics-based* (also referred to as force-field based) scoring functions are based on the binding free energies which are the sum of various interactions between protein–ligand subsystems such as van der Waals, electrostatic, hydrogen bonding, solvation energy, and entropic contributions.
- (ii) The *knowledge-based* scoring functions are based on the available protein–ligand complex structural data from which the distributions of different atom–atom pairwise contacts are estimated. The frequency of appearance of different pairwise contacts are used to compute potential mean force which is used for ranking protein–ligand complexes.
- (iii) Finally, the *empirical scoring* functions, as the name implies, are based on empirical fitting of binding affinity data to potential functions whose weights are computed using a reference test system. Modern scoring functions mainly fall into this class, including the machine learning-based approaches built based on the available information on the protein–ligand 3D structures and inhibition/dissociation constants [10].

As discussed above, there are different scoring functions developed, and this section mainly focuses on implementations available in open-source softwares such as Dock [11], Autodock4.0 [12,13], Autodock Vina [14,15], and Gnina [16]. The docking energy defined to rank protein–ligand complexes (sf) in Autodock4.0 is classified as physics-based and is defined as the sum of van der Waals, electrostatic, hydrogen bonding, and desolvation energy, as shown in Equation (1).

$$sf = W_{vdw} \sum_{i,j} \left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^6} \right) + W_{HB} \sum_{i,j} \left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}} \right) + W_{elec} \sum_{i,j} \left(\frac{q_i q_j}{\epsilon(r_{ij}) r_{ij}} \right) + W_{ds} \sum_{i,j} (S_i V_j + S_j V_i) \left(\exp \frac{-r_{ij}^2}{2\sigma^2} \right) \quad (1)$$

In addition, the entropic contribution which is proportional to number of rotatable bonds is also added to the docking energy. In the equation, r_{ij} refers to the distance between the two atoms, i and j , centered on protein and ligand subsystems. Similarly, q_i and q_j refer to charges on these atoms. A_{ij} and B_{ij} are the coefficients of the potential energy functions describing van der Waals interaction. C_{ij} and D_{ij} are the coefficients of the potential energy functions describing hydrogen bonding interaction. The terms S_i and V_i refer to the solvation parameter and fragmental volume of atom i , respectively.

Because the protein–ligand complexes are considered to be in an aqueous environment, the binding free energies need to account for this, and solvation energy adds the binding free energy differences due to vacuum to aqueous-like environments. In particular, the last term in the equation accounts for this solvation effect (refer to Equation (1)).

In general, the entropic contributions can be due to translational, rotational, and torsional degrees of freedom. However, the docking energies implemented in the aforementioned molecular docking softwares only account for the contributions due to torsional degrees of freedom. The contributions due to other degrees of freedom are assumed to be negligible in ranking different ligands. It is worth noting that certain free energy calculation tools such as MMPBSA.py estimate the entropic contributions due to all degrees of freedom based on normal mode analysis [17]. The entropic contributions due to torsional degrees of freedoms in molecular docking softwares are oversimplified and they are estimated from the number of rotatable bonds (each bond contributes with 0.2–0.5 kcal/mol) [18]. In the case of Autodock Vina, the scoring function can be majorly classified as empirical in nature and it is a sum of various distance-dependent pairwise interactions [14]. It includes terms for describing steric, hydrophobic, and hydrogen bond interactions. The values for different parameters and weights for different terms of potential functions were obtained from nonlinear regression of the PDBbind 2007 dataset. In other words, the empirical scoring functions can have the same mathematical expression as in Equation (1), but the weights/coefficients for different types of interactions are obtained from fitting to experimental binding potentials. The knowledge-based scoring functions [19] (sf_{kb}) have the following form:

$$sf_{kb} = \sum_i^L \sum_j^R -k_B T \ln[g(r_{ij})] \quad (2)$$

Here, the summation runs over the ligand and receptor atoms, and $g(r_{ij})$ refers to the relative probability distribution of distances of a specific types of protein–ligand atom pairs in the docked complex structure when compared to reference experimental complex structure.

Recently, deep learning networks have been proposed to provide scoring functions. For instance, Gnina uses convolutional neural network (CNN)-based scoring function to rank the protein–ligand complexes [16]. The neural networks were trained using three-dimensional protein–ligand complex structures from the PDBbind database. In particular, the dataset contained two sets of poses for the ligands within the binding site. The group of positive poses had root mean square deviation (RMSD) value below 2 Å, when compared to the crystallographic poses, while the rest were considered as a group of negative poses. Here, RMSD is the root mean square deviation in atomic positions in the predicted pose when compared to reference pose, as in the crystal structure. The positive and negative poses were generated using the experimental protein–ligand three-dimensional structures by adopting a random conformation generation algorithm. The CNN model was trained using the 4D grid (which was constructed using the protein–ligand coordinates within the grid box and atom types) to classify the poses.

2.2. Search Algorithms

The search algorithms aim at finding the protein–ligand structure corresponding to the global minimum in a potential energy surface. However, this is a very challenging problem and many search algorithms end up in a local minima. Therefore, molecular docking software uses several techniques, such as deterministic search [20], genetic algorithm, Monte Carlo with simulated annealing [21], particle swarm optimization [22], or Broyden–Fletcher–Goldfarb–Shanno (BFGS) [14]. Deterministic approaches apply techniques such as gradient descent, and they focus on a reproducible sequence of pose evaluations. Monte Carlo algorithms generate numerous poses by using random values for translational, rotational, and torsional degrees of freedom of the ligand. In Monte Carlo-simulated annealing, the heating step allows the system to escape from the local minimum (during which it can sample high-energy regions of the potential energy surface). In genetic algorithms, each pose is defined by a vector of genes that correspond to translational, rotational, and torsional degrees of freedom. By varying these values in the genes, new poses can be generated. The fitness function aims at finding the minimum energy of the pose.

2.3. Validation of Molecular Docking Approaches

As discussed above, molecular docking approaches employ different types of scoring functions, and before implementation they were validated rigorously against available experimental data. In particular, two properties obtained from molecular docking can be considered in general for benchmarking:

- (i) RMSD computed for the predicted binding pose against the crystallographic pose obtained experimentally.
- (ii) Binding free energies/docking energies which are proportional to experimental inhibition/dissociation constants.

The RMSD in the above list is computed from the experimental and predicted protein–ligand complex structures and provides an estimate of how well the molecular docking software is capable of producing the most stable binding mode and binding pose of the ligand within the target biomolecule. An RMSD value of $<2 \text{ \AA}$ is considered as a threshold value for the correct prediction of complex structures [23].

A benchmark study using Autodock4.0 and Autodock Vina on the complex structures (190 in number) from PDBbind showed that the latter could predict structures within the threshold value (i.e., $<2 \text{ \AA}$) for about 78% complexes while the former one achieved 42% [14]. A recent study compared the performance of Autodock4.0 and Autodock Vina in discriminating the active compounds and decoys using a dataset of 102 protein targets, 22,432 active compounds, and 1,380,513 decoy molecules [24]. The study showed that Autodock4.0 was better in discriminating actives and decoys in the targets having hydrophobic binding sites, while the Autodock Vina's performance was better for those targets having binding sites with polar and charged residues [24]. There are other studies reported in the literature which compared the performance of various molecular docking softwares such as AutoDock, DOCK, FlexX, GOLD, and ICM, and the ICM turned out to be the superior performer, with structures predicted for 93% complexes within the acceptable accuracy [23].

The other set of quantities used for benchmarking the molecular docking approaches are the inhibition constants, dissociation constants, IC_{50} and pIC_{50} , which are available from experimental binding assay studies. All these quantities refer to the binding potential or inhibiting potential of ligands to a specific biomolecular target. The dissociation constants and binding free energies are related to each other through the following equation:

$$\Delta G = RT \ln K_d \quad (3)$$

where R is the gas constant (equal to $1.987 \text{ cal K}^{-1} \text{ mol}^{-1}$) and T refers to temperature (set to 298.15 K). Thanks to this equation, the computed docking energies can be directly validated using the experimental binding assay results.

2.4. Computational Cost Associated with Virtual Screening

In a computational drug discovery project, high-affinity lead compounds against a target biomolecule (can be an enzyme, membrane protein, DNA, RNA, Quadruplex, membrane, or fibril aggregates) are identified using a reliable scoring function. One can identify lead compounds for a target from various chemical spaces. The popular chemical spaces are ZINC, Cambridge, ChEMBL, Pubchem, Pubmed, DrugBank, TCM, IMPPAT, and GDB13-17. Refer to Table 1 for a list of different chemical libraries with their properties [25]. The estimate for the size of chemical space for carbon-based compounds with molecular mass <500 daltons is 10^{60} , which clearly indicates that there are limitless possibilities for designing a therapeutic compound. As can be seen, even the use of the top supercomputers with exascale computing speed (which can perform 10^{18} floating point operations per second running the high-performance Linpack benchmark) for screening these compounds will take a universal lifetime. Therefore, it is reasonable to use certain filters to reduce the number of compounds before subjecting to screening. Recently, a filtering procedure based on Bayesian optimization algorithm, referred to as MolPAL, was used to identify the top 50% compounds by developing a model with data of explicit screening of less than 5% compounds of the chemical space [26].

Otherwise, it is reasonable to use other chemical libraries having compounds that are easy to synthesize and having favourable pharmacokinetic (ADMET) properties. The chemical library with the largest number of compounds is GDB17, which contains 166 billion organic molecules made of just 17 atoms of C, O, N, S and halogens [27]. Most of the drug discovery applications use the DrugBank database, Enamine database, ZINC15, [28], and Cambridge, and the number of compounds in these chemical libraries are listed in Table 1. As can be seen, the compounds range from tens of thousands to billions, and the computational cost associated with screening is enormous, which requires use of the HPCs and accelerators.

To demonstrate the computational demand associated with virtual screening, we describe an application below. In the case of the AmpC target, Lyu et al. docked 99 million molecules. For each compound, 4000 orientations on average were generated. Further, for each orientation, about 280 conformations were generated [29]. Therefore, for each ligand, 1.1 M docking energy calculations were carried out, and given that the number of compounds considered were 99 M, a total of 10^{13} of such calculations were carried out. This will be further increased in the flexible receptor docking where the sampling over side-chain conformations of residues needs to be accounted for [29]. As can be seen, the computational demand is really huge with such virtual screening applications, and so it is inevitable to develop parallel algorithms and to use HPCs to accomplish such screenings within affordable time.

Table 1. Top chemical spaces available for VS.

Chemical Library	NO of Compounds	Features
Virtual compounds	10^{60}	Molecular mass ≤ 500 daltons
GDB17 [25]	166 B	17 heavy atoms of type C, O, N, S and halogens
REAL DB (Enamine) [30]	1.95 B	Synthesizeable compounds $M \leq 500$, Slogd ≤ 5 , HB ≤ 10 , HB ≤ 5 rotatable bond ≤ 10 , and TPS ≤ 140
ZINC15 [28]	980 M	Synthesizeable, available in ready-to-dock format
Pubchem [31]	90 M	Literature-derived bioactive compounds
ChEMBL [32]	63 M	Curated database with chemical structure and physicochemical properties
ChEMBL [33]	2 M	Manually-curated drug-like bioactive molecules

3. Milestones in Virtual Screening

The first virtual screening using 3D structures of chemical compounds was carried out in 1990 against the target, dopamine D2 agonists based on which agonist with pKi 6.8 was successfully found [34]. A similar search against other targets, such as alpha-amylase, thermolysin, and HIV-I protease, yielded inhibitors with significant inhibition potential [34,35]. The top compounds obtained from screening were found to be potential inhibitors, and with further optimization of the lead compounds, the inhibition potential increased considerably. With the use of more accurate scoring functions and large-sized chemical libraries and powerful computers, the computer-aided drug discovery can become a potential route to narrow down the search space before subjecting to experimental high-throughput screening. Thanks to the currently available Petaflop/s computing facilities, one can screen a billion compounds in a day. The latest record on the high-speed virtual screening is reported by Jens Claser et al. [36], where the authors screened a billion compounds within 21 h.

The number of compounds screened against various targets keeps increasing with time. As the chance for a compound with better binding affinity increases with size of the chemical library, such screening procedures result in identifying highly potent compounds. We here list a few example cases (refer to Table 2): The screening study by J. Lyu et al. [29] yielded an active compound (339204163), which had 20 times more potency than the known inhibitors for AmpC target. The most potent inhibitor for the same target was designed by the same research group by optimizing the lead compound (the end compound was referred to as 549719643). Similarly, 10-fold more potent agonists (465129598, 270269326, and 464771011) were identified for the D4 receptor [29]. The most potent compound with 180 pM binding affinity, 621433144, for the same target has been identified by the same group. Using the multistage docking workflow, referred to as Virtual Flow for Virtual Screening (VFVS) (where different docking softwares such as Quickvina2, Smina vinardo, and Autodock Vina were used in sequence), the inhibitors for KEAP1 target were recently identified from the chemical space of 1.4 billion compounds (made of Enamine Real database and ZINC15) [37]. The first round of scoring was carried out using Quickvina2, while the rescoring was carried out for the top 3 million compounds using Smina vinardo and Autodock Vina. An inhibitor, iKEAP1 with dissociation constant 114 nM, was identified which is shown to interrupt the interaction between the KEAP1 and transcription factor NRF2 [37].

Thanks to parallel implementations of molecular docking software, not only has the number of compounds used for screening increased drastically, but the time required to accomplish the high-throughput screening has also reduced considerably. The inhibitors for the targets Purine Nucleoside Phosphorylase and Heat Shock Protein 90 were identified from the REAL enamine database (1.4 B compounds) using CPU-enabled Orion software [38]. The recent screening of compounds from Enamine database against the enzyme targets from SARS-CoV-2 was carried out in 21 h instead of 43 days with the use of parallel implementation of Autodock4.0 (referred to as Autodock-GPU) on the Summit supercomputer [39,40]. Currently, the largest experiment ever run was achieved using the EXSCALATE platform based on LIGEN software. It virtual-screened a library of 71.6 B compounds against 15 docking sites of 12 viral proteins of SARS-CoV-2. The experiment was carried out on the CINECA-Marconi100 and ENI-HPC5 supercomputers, and it lasted 60 h [41], performing more than 1 trillion docking evaluations overall. A list of megadocking and gigadocking screening calculations on large-sized chemical libraries is presented in Table 2.

Table 2. Topmost mega or gigadocking applications reported in the literature.

NO	Year	Target	No of Compounds	Docking Tool
1	2019	enzyme AmpC	99 M	Dock3.7
2	2019	D4 dopamine receptor	138 M	Dock3.7
3	2019	Purine Nucleoside Phosphorylase	1.43 B	Orion
4	2019	Heat Shock Protein 90	1.43 B	Orion
5	2020	KEAP1	1.4 B	Quickvina2
6	2021	Mpro	1.37 B	Autodock-GPU
7	2021	12 SARS-CoV-2 Proteins	71.6 B	LiGen

4. High-Performance Computing

Molecular docking for a single chemical compound involves sampling and scoring. This refers to sampling over the configurational phase space for the compound in the target binding site and computing the scoring functions for each of these configurations generated. In the case of virtual screening, compounds from a chemical library are ranked against a single target with respect to their binding affinities, and this additional component is referred to as ranking (refer to Figure 1). As discussed in the previous section, this is computationally very demanding and can highly benefit from the use of the parallel algorithms which can run on high-performance computers with different (shared memory and/or distributed) architectures [42,43]. Nowadays, the massively parallel computing units, referred as graphical processing units (GPUs) and field programmable gate arrays (FPGAs), are accessible to research groups or even individuals, and with the development of parallel virtual screening software, drug discovery projects can be offloaded to such groups, making the drug discovery economically sustainable. Below, we highlight the opportunities for the parallelization of VS protocol.

4.1. Parallelization Strategies of Virtual Screening for High-Performance Computers

As we discussed above, the virtual screening involves three key steps:

- (i) Sampling over configurational phase space of ligands within the binding site.
- (ii) Estimating the scoring function for each of the configurations of the chemical compound within the target binding site to identify the most stable binding mode/pose.
- (iii) Ranking of compounds with respect to their relative binding potentials.

Each of these key steps can be parallelized, as the computations can be carried out independently.

The estimate of scoring function for each binding mode/pose within the protein binding site involves the computation of docking energies or binding free energies or any other empirical potentials. We need to find the most stable binding mode for each of the ligands in the target binding site (which corresponds to a global minimum in the protein–ligand potential energy surface). Therefore, for each ligand, millions of configurations (each configuration is a point in the ligand configurational phase space) are generated and the scoring functions are estimated for these structures. As discussed above, the configurations can be generated by changing the translational, rotational, or torsional degrees of freedom. These changes can be performed with a deterministic methodology or by using random-driven approaches such as Monte Carlo simulations or genetic algorithms. The estimate of energies for each of these configurations can be carried out independently; these can be distributed to different computing units.

In the virtual screening, the aforementioned procedure is repeated for all the ligands in the chemical library, and even this can be carried out independently, and so can be distributed to different computing units. Finally, even the calculation of the energy of a single conformation can be parallelized, as it needs to estimate the interactions of all the ligand–protein atoms couples. Overall, the parallelization of the virtual screening approach can be implemented in the following three steps, as we also show in Figure 2.

- (i) **Low-level parallelization (LLP):** Parallelizing the energy calculation.
- (ii) **Mid-level parallelization (MLP):** Parallelizing the conformer evaluations and scoring.

- (iii) **High-level parallelization (HLP):** Parallelizing the ligands evaluations on different computing units.

These three different levels have different efficiencies, as the amount of data transfer between the computing units is very different and may become a strong overhead. Moreover, it is also possible to combine more of these strategies in a single application to address different level of parallelism available (such as, for example, HLP for multi-node and MLP for multi-core architectures). The low-level parallelism approach is, however, usually not appealing, since the computed docking energies for different docking poses need to be compared, and this involves frequent data transfer between different computing units. The docking energy calculation involves summation over the pairwise interactions of the atoms centered on protein and ligand subsystems. In case of flexible targets, the intramolecular energies also need to be computed, which is obtained from a double summation over the number of atoms in the target. The non-bonded interaction calculation (sum of electrostatic and van der Waals) is the most time-consuming part of the energy calculation (by 80%) [42]. Stone et al. showed a 100-times speed-up for the non-bonded energy calculation using GPUs, while Harvey et al. showed 200-times increased performance [44,45]. A more sophisticated algorithm which effectively distributed the tasks to CPUs and GPUs for computing non-bonded interactions was developed by Gine's D. Guerrero et al. [46]. In this case, each atom of the receptor was assigned to a single thread in GPUs, which handle computing its interaction with all the ligand atoms [46]. Each thread was provided with necessary ligand and receptor atom coordinates and charges. The speed-up due to this algorithm was up to 280 times.

The parallelization of scoring function calculation is the most problematic approach, as it requires to sum all the atom contributions that are evaluated by different compute units (see Figure 2). This introduces frequent small data transfers that may be limiting the scaling behavior. For this reason, the other two techniques are the most used in parallelizing VS software. Indeed, computing the energy for different configurations of the ligand can be carried out independently. Similarly, finding the global minimum structure for each of the ligands as well can be carried out independently. Each molecular docking step involves generation of millions of configurations, and the scoring functions for all the conformers can be calculated independently by assigning the tasks to different computing units. The energies for conformers can be gathered and checked for the least energy configuration corresponding to global minimum in the protein–ligand free energy surface.

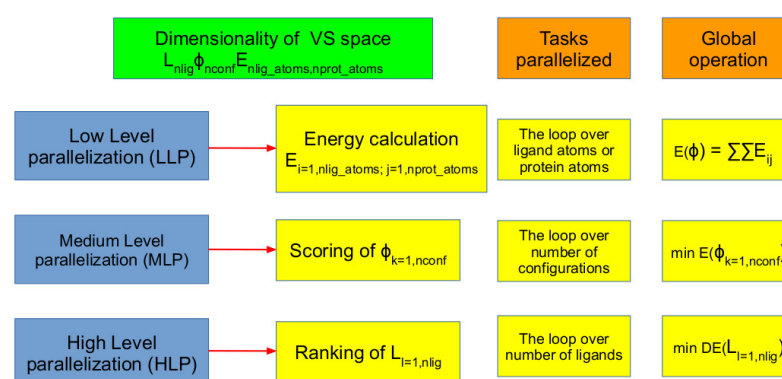


Figure 2. Various parallelization opportunities in virtual screening. The terms *nlig*, *nconf*, *nlig_atoms*, and *nprot_atoms* refer to number of ligands in a chemical library, number of configurations (specific to each ligand and dictated by number of rotatable bonds), number of ligand atoms, and number of protein atoms, respectively.

4.2. HPCs and Accelerator Technology as Problem Solvers

Parallel computing architectures can be classified based on the memory availability to computing units:

- (i) In the shared memory architectures, a set of processors use the same memory segment.

(ii) In the case of distributed memory architectures, each computing unit has its own memory.

In general, a high-performance computing environment is usually made of both architectures where the shared-memory multi-core CPUs are connected through high-performance network connections. The memory in different nodes are local and are available to those cores within the specific node. The parallel execution of tasks in a shared memory architecture can be achieved by compiler directives such as OpenMP pragmas or libraries such as pthreads.

On the other hand, the parallel implementation of tasks in distributed memory architectures requires to handle the communication manually, using libraries such as Message Passing Interface (MPI) between different nodes of the machine.

As most of the supercomputers have both shared memory behavior (on a single node) and distributed memory behavior, as they are multi-node machines, it is easy to see the usage of both programming libraries for parallelizing the VS. In addition to multiple CPUs, recent HPCs also are integrated with accelerators such as graphical processing units, which have their own memory unit. The programming languages for CPU+GPU supercomputers are CUDA, OpenACC, and OpenCL. To use the accelerators, it is mandatory to move the data between the host and the device memory, which means that the developer needs to copy the data on the device before performing the computation and the result from the device after the computation is done. In addition, the tasks distribution to GPU cores and synchronization of the task execution in CPU and GPUs are achieved.

In virtual screening, the docking of different ligands can be carried out independently from each other. The procedure can be effectively parallelized on supercomputers with thousands of multiprocessor nodes, as well as in multi-CPU+GPU computers. As discussed above, there are different possibilities for running the virtual screening in HPCs. The molecular docking procedure for each ligand can be parallelized by distributing the docking energy calculations for different conformers of the ligand in a target binding site. In this case, the receptor coordinates and ligand coordinates and their charges need to be provided to all computing units (i.e., server nodes or GPU cards). In the second scenario, the molecular docking workload for different ligands is distributed to different computing units. In this case, the receptor coordinates and charges should be made available to all the computing units, while the ligand coordinates can be made available to the specific computing unit handling this specific ligand. The main disadvantage in this way of distributing tasks is that the ligands can have different sizes and so a rank assigned with the smaller sized ligand completes the task and waits until the molecular docking procedure is completed for all the ligands in different ranks to accept the tasks of second-round screening. This way, the computing time in this specific node is wasted. Thus, it is usually recommended to sort the ligands in terms of their size, and then the distribution of ligands can be carried out from the list. In this way, all the ranks will have more-or-less equivalent-sized tasks, and the wall time in different ranks is efficiently used. The next section provides an overview of the different software targeting HPC systems, also highlighting the type of parallelization employed.

5. Current Implementation of VS Available for Workstations, Accelerators, and HPCs

Currently, there are many parallel implementations for performing virtual screening in multicore machines, clusters, and accelerators. As discussed above, different strategies were adopted for parallel virtual screening. Only a few virtual screening softwares, such as Flexscreen, use the parallel implementation of docking energy calculation. The remaining perform the parallelization by distributing the conformational sampling/scoring or the ligands ranking segments over different computing units.

Table 3 shows a comparison of the analyzed software under some key features. As we can notice, most of the available softwares have the possibility to scale on multiple nodes, thus are able to exploit the whole HPC machine. All of them are able to fully utilize a node, while only a few have the support for the GPU acceleration. Moreover, as we already anticipated, all of them use an MLP (parallel conformational search) or HLP (parallel ligand

evaluations) or both strategies to parallelize the computation, while none of them uses a low-level parallelism (parallel energy evaluations).

We now analyze, one by one, more in detail, these softwares: we focus on their performance in massively parallel architectures and accelerators when compared to CPU implementation. Wherever possible, the accuracy of the docking results (in terms of reproducing the experimental protein–ligand complex structure and experimental inhibition constants) will be discussed.

Table 3. Different parallel virtual screening softwares and important features.

Software	Parallelization Segment	Programming Language	Scoring Function	Minimization	Multi Thread	Multi Node	GPU
Dock 5,6	Conformational search	C++, C, Fortran77, MPI	Physics-based and hybrid	Monte Carlo	Yes	Yes	No
DOVIS2.0	Ligand screening	C++, Perl, Python	Physics-based	Monte Carlo&GA	Yes	Yes	No
Autodock Vina	Conformational search	C++, OpenMP	Hybrid	Monte Carlo	Yes	No	No
VSDocker	Ligand screening	C++, Perl, Python	Physics-based	Monte Carlo&GA	Yes	Yes	No
MPAD4	Ligand screening, Conformational search	C++, MPI, OpenMP	Physics-based	Lamarckian GA	Yes	Yes	No
VinaLC	Ligand screening, Conformational search	C++, MPI, OpenMP	Hybrid	Monte Carlo	Yes	Yes	No
VinaMPI	Ligand screening, Conformational search	C++, MPI, OpenMP	Hybrid	Monte Carlo	Yes	Yes	No
Ligen Docker-HT	Ligand screening, Conformational search	C++, MPI, CUDA	Empirical	Deterministic	Yes	Yes	Yes
GeauxDock	Ligand screening, Conformational search	C++, OpenMP, CUDA	Physics- and knowledge-based	Monte Carlo	Yes	Yes	Yes
POAP	Ligand screening	bash	Same as parent docking software	Same as parent Docking software	Yes	Yes	No
GNINA	Conformational search	C++	Empirical and CNN ML	Monte Carlo	Yes	Yes	Yes
Autodock-GPU	Ligand screening	C++ and OpenCL	Physics-based	MC/ LGA	Yes	No	Yes

5.1. Dock5,6

Dock5 and 6 [47,48] are the parallel implementations of virtual screening program written in C++ with MPI libraries. They are based on the original version referred to as Dock1, which was developed by Kuntz and coworkers [11]. The Dock1 used a geometric shape-matching approach for identifying the lead compounds for a given protein target. The subsequent versions adopted physics-based scoring functions for ranking and had improvement over the thoroughness of sampling and accounted for the ligand flexibility. The recent version allowed using multiple scoring functions where the solvation energies are computed using different implicit models, such as Zou GB/SA [49], Hawkins GB/SA [50], PB/SA [51], and generalized solvent models as implemented in Amber16. The benchmarking study to evaluate the performance of recent Dock6 (V6.7) used SB2012 dataset [52] which is a collection of crystallographic structures for about 1043 receptor–ligand complexes. It was able to reproduce the crystallographic poses in this dataset to an extent of 73.3% (i.e., RMSD between the experimental and predicted binding poses was <2 Å), which is due to reduction in the sampling failure of previous Dock versions. There is also a report in the literature on the offloading of Dock6 to CPU+GPU architectures using CUDA [53]. Only the ranking using amber scoring was offloaded to GPU architectures. In this offloading, the coordinates, gradients, and velocities are copied to host (CPU) memory to device (GPU) memory and the results are copied back from GPU to CPU memory. As the GPU could only handle single-precision numbers, the original data in double-precision were converted to single-precision numbers before transfer. Overall, the study reported about a 6.5-speedup for the amber scoring in Dock6 in GPU (NVIDIA GeForce 9800 GT) when compared to AMD dual-core CPU [53].

5.2. DOVIS2.0 VSDocker2.0

Both these parallel VS softwares use Autodock4.0 as docking engine. The former one has been developed for the multi-CPU systems with Linux operating system (OS), while the latter one is for parallel computer system running with MS Windows OS. DOVIS2.0 [54] uses multithreading, SSH, and batch system for parallelization, while C++ library, Perl, and Python are used for ligand format conversion, virtual screening workflow, and receptor grid files preparation, respectively. Instead of preparing the receptor grid files for each ligand, the program reuses them, and so the I/O file operation is reduced drastically. As the program employs dynamic job controlling, the load balancing due to different size of the ligands is effectively handled. Once the docking of block of ligands assigned to a CPU is completed, the results are written to project directory. Subsequently, the CPU

requests for newer assignment for screening. This way, the CPUs are continuously in action until all the ligands are ranked and the results are written to shared directory. During a virtual screening for 2.3 M ligands from ZINC database on 256 CPUs, the DOVIS2.0 achieved a ranking speed of 670 ligands/CPU/day. The VSDocker program is developed to carry parallel virtual screening in both multiprocessor computing clusters and workstations running MS windows OS. The program is written in C++ and parallel library `mpi.h` [55]. In this VS implementation, the receptor map file generation and analysis of docking results for different ligands are carried out sequentially while the docking of different ligands are carried out in parallel. The performance of VSDocker has been tested using a chemical library of 86,775 ligands from ZINC database against a target made of 145 amino acids. The speed-up was comparable to that of DOVIS2.0, which is similar to VSDocker in implementation, but is suited for Linux running computer clusters.

5.3. Autodock Vina

Autodock4.0 is the most widely used molecular docking software based on physics-based scored function, but the original version is sequential in nature. However, the Autodock Vina, which is also from the same developers at Scripps Institute, can use multiple cores simultaneously for carrying out the docking. The calculations can be executed with single-threaded or multithreading options. The implementation uses C++ with Boost thread libraries. In the multithreading version, multiple Monte Carlo simulations are initiated with different random number seed to explore different areas of conformational space of the ligand within the binding site. In a benchmarking study, for the same protein-ligand complex, the Autodock Vina [14] with single thread ran 62 times faster than the Autodock4.0. Further, running Autodock Vina with multithreading option in eight CPU machines yielded a 7.3-times-faster (when compared to single-threaded option) completion of molecular docking. The performance of single-threaded Autodock Vina compared to Autodock should be attributed to the difference in the computational cost of scoring functions. The more than 7-times speed-up with multithreading option in an eight-core machine shows that the molecular docking calculation scales well with the number of cores. Autodock Vina relies on OpenMP for distributing tasks to different threads and so is suitable for workstations with multiple cores. However, for the supercomputers with distributed memory, this version of Autodock Vina is not suitable, and rather more robust programs that allow the data transfer and communication between different nodes need to be used. It is also worth mentioning that there are updated versions of Autodock Vina, namely, Qvina 1 and Qvina 2, which showed some speedup due to the improvement in local search algorithm. SMINA [56] is a fork of Autodock Vina with a number of additional features such as user-specified scoring function, creating grid box for docking based on the coordinates of ligand bound to target, improved minimization, feasibility to include residues for flexible docking, and possibility to print more than 20 poses. In terms of speed-up in HPCs, this did not contribute to any improvement.

5.4. MPAD4

MPAD4 [57] is a parallel implementation of Autodock4.0, and the important features when compared to parent code are listed as follows: (i) It uses MPI to distribute docking jobs across the cluster; (ii) The grid maps generated for receptor are reused for all the docking calculations while in the default version, and these files are generated for each ligand docking with the target receptor and loaded into memory and released at the completion of docking. In MPAD4, The maps are loaded into memory of the node at the beginning of tasks and are used for all remaining docking calculations. This greatly reduces I/O usage, contributing to speed-up in the performance; (iii) The OpenMP is used for the node-level parallelization in executing the LGA for finding the global minimum. Overall, it allows system-level and node-level parallelization. The performance analysis of MPI version and MPI+OpenMP version can be studied by setting OMP to 1 and 4, respectively. The computers used for the initial performance analysis were IBM Blue gene/P and shared-memory 32-core POWER7 p755 server. The dataset used for performance analysis was

HIV protease target and 9000 compounds from a druglike subset of ZINC8 database (which has 34,481 ligands in total). The performance analysis in Blue gene/P with 2048 (8192) node (core) showed that grid map reuse has reduced single-threaded execution time by 17.5%. Multithreaded execution of the code yielded 25% improvement in the overall performance. The execution of the code on nodes ranging from 512–4096 showed near-linear scaling behavior for symmetric multiprocessing (SMP) mode (OMP = 4). In particular, for 16,384 core system, the speed gained was 92% compared to that of the ideal case. The virtual node (VN) mode (OMP = 1) with the grid map “reuse” option showed, however, 72% speed-up when compared to the ideal case. The gain in the performance of SMP mode should be attributed to multithreading. The node utilization can be further improved with the use of preordering ligands with decreasing number of torsional angles.

5.5. VinaLC

VinaLC [58] is an Autodock Vina extension to use MPI library for parallelization in large supercomputers. This implementation is very similar to VinaMPI (which is described below) and uses MPI and multithreading hybrid scheme for parallelization across nodes and within node, respectively. The computational cost of each docking calculation depends on the size of the ligand, receptors, and the grid box. If the calculations are distributed on all MPI processes due to this uneven size of the input systems, there can be MPI load imbalance where the processes are waiting for other processes to complete the task. This load imbalance is tackled effectively by the master–slave MPI scheme where the master process handles the inputs, and outputs job allocation to the slave processes. The tasks in the master slaves are handled by three loops: (i) the first loop is over each combination of receptor–target ligand which assigns docking task to a free slave; (ii) The second loop collects the docking results from slave processes and the new tasks are assigned in case of unfinished calculations; (iii) The third loop frees the slave processes. In the slave processes, an infinite whole loop is initiated which ends when the “job finished” signal is received from the master. The ideal slave processes are identified from the MPI_ANY_source tag and docking tasks are assigned upon the completion of previous task. In this way, the computing resources available in different processes are utilized efficiently. To make the communication effective, all the inputs needed for a single docking calculation are sent by single MPI_Send call. Therefore, coordinates of the receptor and ligand and grids (which are computed on the fly in the master process) are sent to the slave process. The outputs from the slave processes are collected by the master process into a few files instead of generating file for each ligands (which will generate a million or billion files depending upon the size of the chemical library). The benchmarking study was carried out using the two datasets, namely ZINC and DUD (directory of useful decoys). The target was chosen as *Thermus thermophilus* gyrase B ATP-binding domain. The benchmarking calculations were carried out on HPC machines at Lawrence Livermore National Laboratory, and the number of cores used were in the range 600–15,408. The study showed that the average CPU time per docking was closer to ideal average CPU time. The VinaLC was shown to scale well up to 15 K cores. The percentage of I/O activity was reported to be negligible when compared to the total computing time. For aforementioned target using 15 K cores, VinaLC could screen about one million compounds from Zinc15 database in 1.4 h. This can be extrapolated to 17 million compounds per day which suggests the suitability of VinaLC for the most time-taking mega- or gigadocking screening applications.

5.6. VINAMPI

VinaMPI is another implementation of Autodock Vina for distributed computing architectures [59]. It is written in C and for communication between the nodes, it uses MPI libraries. In order to avoid poor scaling behavior of the parent-child (or master-slave) distribution scheme in massively parallel supercomputers, this implementation uses all-worker scheme. It is worth recalling that rather VinaLC used Master-slave scheme for distributing tasks. In this code, each worker (or each MPI rank) deals with its own protein-ligand complex and within each rank the computation (related to search of global

minimum) is carried out using multithreading. Due to this reason it is also suitable for the virtual screening for more than one targets. Further the computations are sorted out in terms of complexity so that the work loads at a given round of screening can be distributed equivalently. The computational complexity is measured based on the number of rotatable bonds and size of the ligands which is used to sort the tasks in the beginning of the screening. As a benchmark, the dockings were carried out for targets namely ACE (angiotensin-converting enzyme), ER AGONIST (estrogen receptor agonist), VEGFR2 (vascular endothelial growth factor receptor kinase), and PARP (poly(ADP-ribose) polymerase) with a chemical library of 98,164 compounds (comprised of ligands and decoys). Running this screening in a 516 cores supercomputer costed about 103 s [59]. It is expected that the same implementation in a supercomputer with 0.3 M cores can be used to screen 250 M compounds in 24 h.

5.7. LiGen Docker-HT

LiGen [60] is a VS software that leverages CPU and GPU to perform the required computation. Several versions of the tool have been developed, starting from a CPU-only application [20,60,61], then the main kernels have been ported to the GPU by [62] using OpenACC. Finally, it has been optimized using CUDA for the GPU kernels and this last version has been used to perform a large VS experiment in the search of a therapeutic cure for COVID-19 screening 71.6 B compounds against 15 binding sites from 12 Sars-Cov2 viral proteins on two supercomputers, accounting for 81PFlops [41]. LiGen uses deterministic algorithms to generate the different conformers of a ligand, and an empirical scoring function to select the best molecule. The Docker-HT application is the version of LiGen that is designed targeting a large VS campaign, and it is able to leverage multi-node, multi-core, and heterogeneous systems. In particular, it uses MPI to perform the multi-node communication, which is limited as much as possible by the algorithm to avoid large communication overheads [63]. Indeed, the amount of data that needs to be processed by every node is divided beforehand, and it may create load-balancing issues as there is no mechanism to rebalance it during the execution of the application. On the single node, it leverages the C++ thread library. Finally, it uses CUDA to support GPU acceleration. Within each node, the program uses pipeline parallelism and work-stealing to process the ligands.

5.8. Geauxdock

This is a parallel implementation of virtual screening available for multicore CPU, GPU, and Xeon Phi-computers. The software uses a common code for front-end computations in all these computers [64]. However, the back-end codes have one version for CPU and Xeon Phi architectures, and another for GPU. The code for CPU and Xeon Phi architectures written in C++, OpenMP and IntelSIMD pragmas. The GPU version is written in C++ and CUDA. The program uses the Monte Carlo approach for conformational search and for identifying the global minimum of the protein–ligand complex. The scoring function is based on physics-based energy terms combined with statistical and knowledge-based potentials. The performance of the code has been tested in multi-core CPUs and massively parallel architectures, namely Xeon Phi and NVIDIA GPUs. The testing using CCDC/Astex dataset showed a 1.9-times increase in performance for Xeon Phi when compared to 10-core Xeon CPU. Further, on the GeForce GTX 980 GPU accelerator, the performance was 3.5 times higher when compared to the CPU version.

5.9. POAP

Poap is a GNU parallel-based multithreaded pipeline for preprocessing ligands, for virtual screening, and for postprocessing the docking results [65]. It also allows the minimal use of memory through optimized dynamic file-handling protocol. It has also been optimized in a way that erroneous ligand input does not affect the workflow. It can be integrated with any of other sequential or multithreaded molecular docking softwares, such as Autodock4.0, Autodock Vina, or AutodockZn. In the case of Autodock-based virtual

screening, the map files are generated for each ligand in the datasets. In the case of POAP, the map files are directed to a common hub directory and so occupancy of space due to redundant atom types in ligands is overcome. The number of threads to be used should be mentioned when the Autodock Vina is used as a docking software. In the case of Autodock and AutodockZn, the number of parallel jobs to be executed (which can be equal to the number of CPU threads) should be defined by the user. The performance of POAP has been tested using the virtual screening for the targets namely Human ROCK I, HTH-type transcriptional regulator, Polyketide synthase, and PqsA (Anthranilate-coenzyme A ligase) using the ligands from the chemical library, DrugBank. Since POAP does not have any serial code, the speed-up (theoretical estimate) is directly proportional to the number of processors used. The performance analysis of Autodock in a T5510 DELL workstation with Intel Xeon(R) CPU E5-2620V2, 2.10 GHz clock speed (12 Cores, 24 threads) with 62.9 GB RAM showed a 12.4-times speed-up when compared to serial mode (the number of parallel jobs specified is 24). Similarly, the Autodock Vina showed 2.4-times speed when compared to default mode (which is already multithreaded) and here, the number of jobs was set to three.

5.10. GNINA

GNINA is a fork of SMINA [56] and Autodock Vina [16]. When compared to the hybrid scoring function employed in Autdock Vina, it provides options to use various built-in scoring functions (such as Vina, Vinardo) along with user-customized scoring functions. More importantly, it provides a machine learning-based scoring function to rank the complexes. The default scoring function (called “none” option) is the same as used in Vina or Smina, while the rescoring (called “rescoring” option) allows the topmost ligand poses to be ranked using machine learning-based scoring functions.

In particular, this scoring function is based on convolutional neural networks trained using 3D protein–ligand complex structures (as reported in PDBBind or BindingDB) and corresponding experimental inhibition constants. They were trained to reproduce binding pose and the binding affinity. There are multiple machine learning functions (namely, *crossdock_default2018*, *dense*, *general_default2018*, *redock_default2018*, and *default2017*) provided by GNINA, and these have been developed using different datasets. The CNN-based scoring function outperforms the scoring function implemented in Vina in reproducing the binding poses. The RMSDs for the predicted poses in the unseen examples are below 2 Å in as many cases as 56%. Further, the binding prediction of poses within this cutoff improves to 79% if the redocking is performed.

When compared to Autodock Vina, the grid box center can be provided with the help of a ligand file. For the conformational search, GNINA uses Monte Carlo sampling scheme. The sampling is carried out over the ligand translational, rotational, and torsional degrees of freedom. In the case of flexible docking, the sampling is also carried out over the residue side-chain conformations. Unlike Smina, GNINA performs computing in single-precision (32 bit) which allows the possibility of offloading the CNN scoring tasks to GPUs. Even though GNINA can be used in massively parallel supercomputers and HPCs with accelerators, there is no performance analysis or profiling when compared to other docking softwares reported in the literature.

5.11. AUTODOCK-GPU

Autodock4.0 is one of the most widely used molecular docking softwares, but it is a serial code which runs on a single thread so cannot be effectively used in high-performance computing environments with multiple CPUs and GPUs. Autodock-GPU [66] is the version of Autodock developed for multiple node parallel computers with GPU accelerators. It is worth recalling that the above discussed MPAD4 was developed for multi-CPU architectures. This program has been developed using the application programming interface, OpenCL, as it allows portability to hybrid platforms with CPUs and GPUs. When compared to Autodock4.0, the local search algorithm uses derivatives of energies with respect to translations, rotations, and torsions (this implementation of gradient-based local

search is referred to as ADADELTA). In the case of CPU+GPU architectures, the workflow consists of a sequence of host and device functions. In analogy to biological gene, the state of the protein–ligand complex is represented by a sequence of variables. In the case of a rigid docking (where the protein framework is treated as a rigid body), the variables represent positions, orientations, and conformation of the ligand. The number of variables are $6+N_{rot}$, where N_{rot} is the number of rotatable bonds. The docking is aimed at finding the genotype which corresponds to the global minimum in the protein–ligand potential energy surface and the ranking is dictated by the scoring function.

The performance of Autodock-GPU has been tested using the diverse data set of 140 protein–ligand complexes from Astex Diversity Set [67], CASF-2013 [68], and protein databank. The reference docking calculations were performed using the single-threaded Autodock4.2.6. The speedup performance was dependent on the minimization algorithm used for the local search (whether Solis–Wets or ADADELTA), GPU type, and the type of protein used in the docking. With the use of Solis–Wets local search algorithm, the speed-up was 30 to 350 times in GPUs, with the M2000 showing the least performance and with TITAN V showing the best. However, with the use of ADADELTA local search, the speed-up was only 2 to 80 times improved, which has to be attributed to the computationally expensive calculation of gradients and difficulties associated with the parallelization of this local minimization step. In general, TITAN V cards showed 10 times higher speed-up when compared to M2000 versions. The performance analysis in multiple-core CPUs showed a similar trend where for the Solis–Wets search, the speed-up was in the range 5 to 33 times (the number of cores employed 8–36), while for the ADADELTA local search the speed-up was 2–20 times better.

5.12. Other VS Tools

The focus of this review was mostly about the open-source parallel VS softwares which are summarized in Table 3 along with some important features. The details about the year they were introduced and source URL pages are listed in Table 4. Many of these softwares such as GeauxDock, Autodock-GPU, and GNINA, were introduced in recent years and so their capacity in the lead compounds identification from huge chemical libraries needs to be validated extensively. Meanwhile, many already existing virtual screening softwares have contributed to successful lead compound identification and lead optimization over the years. In particular, the softwares such as FlexX, DOCK (the sequential version of above discussed Dock6), SLIDE, Fred (OpenEye), GOLD, LigandFit, PRO_LEADS, ICM, GLIDE, LUDI, and QXP are worth mentioning [69]. Among these, LigandFit and QXP employ Monte Carlo for sampling, while SLIDE and Fred employ conformational ensembles approach. GOLD, ICM, and GLIDE, respectively, adopt genetic algorithm, pseudo-Brownian sampling/local minimization, and exhaustive search for sampling. Finally, Dock and FlexX use incremental build approach for identifying the most stable binding mode/pose for the ligand. There are other VS softwares, such as RosettaDock [70], Surfex [71], and LIDAEUS [72], which are not discussed here as the review focuses on those with parallelism capability. It is also worth mentioning Spark-VS [73] software, which uses Google’s MapReduce to run parallel virtual screening in distributed cloud resources. The parallel efficiency of Spark-VS against a chemical library of 2.2 M compounds is reported to be 87% when compared to a public cloud environment [73]. This opens up another possibility of using cloud computing resources for parallel virtual screening without a need to buy our own HPCs and accelerators.

Table 4. Timeline for different parallel virtual screening software and source URLs (with accessed dates in bracket).

No	Year	Parallel VS	Source
1	2006	Dock5&6	http://dock.docking.org/ (1 August 2021)
2	2008	DOVIS2.0	http://www.bioanalysis.org/downloads/DOVIS-2.0.1-installer.tar.gz (15 December 2021)
3	2009	Autodock Vina	http://vina.scripps.edu/ (1 August 2021)
4	2010	VSDocker	http://www.bio.nnov.ru/projects/vsdocker2/ (15 December 2021)
5	2011	MPAD4	http://autodock.scripps.edu/downloads/multilevel-parallel-autodock4.2 (15 August 2021)
6	2013	vinaMPI	https://github.com/mokarrom/mpi-vina (1 June 2021)
7	2013	vinaLC	https://github.com/XiaohuaZhangLLNL/VinaLC (10 June 2021)
8	2016	GeauxDock	http://www.brylinski.org/geauxdock (20 June 2021)
9	2018	POAP	https://github.com/inpacdb/POAP (21 June 2021)
10	2021	Autodock-GPU	https://github.com/ccsb-scripps/AutoDock-GPU (10 September 2021)
11	2021	GNINA	https://github.com/gnina/gnina (1 November 2021)

6. Emerging Reconfigurable Architectures for Molecular Docking

In prior sections, we have focused exclusively on reviewing methods of virtual screening and molecular docking that target modern central processing unit (CPU) and graphics processing unit (GPU) solutions (refer to Figure 3). At the same time, we know that Moore's law (transistor scaling) is terminating, which could motivate (or even necessitate) the search for alternative computing platforms that can continue the performance trend that molecular docking has come to rely upon. Among the many (so-called) post-Moore technologies [74], reconfigurable architectures are perhaps the most noticeable, partially because they are readily available today. A reconfigurable architecture, such as a field-programmable gate array (FPGA) or coarse-grained reconfigurable array (CGRAs) [75] is a system which aspires to retain some of the silicon plasticity that is lost when manufacturing an application-specific integrated circuit (ASIC). In turn, users can leverage reconfigurable systems to perfectly match the hardware to the application, which in turn can lead to improvement in performance and reduction in energy costs. For example, the expensive von Neumann bottleneck associated with the decoding of instructions in CPUs can be virtually eliminated. Traditionally, reconfigurable architectures such as FPGAs have been programmed using complex, low-level hardware description languages (HDLs) such as VHDL or Verilog. This, in turn, has limited exposure of using these devices to specialized hardware and is thus out of reach for typical HPC users. However, with the increase in maturity of high-level synthesis (HLS) [76] tools in the past decade, today, programmers can describe their hardware in abstract languages such as C/C++ and directive-driven models (e.g., OpenCL [77] or OpenMP [78]) and automatically translate the code down to specialized hardware. Modern HLS has, in turn, facilitated the accelerated use and research of FPGAs in other HPC applications such as computational fluid dynamics, neuroscience, and molecular docking.

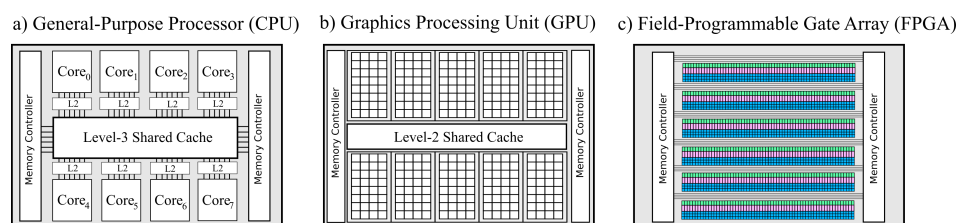


Figure 3. A conceptual picture of different processors and accelerators, showing (a) CPUs, which are latency-focused architectures with few processing units and large (and deep) memory hierarchies, (b) a GPU, which is a throughput-focused architecture with more processing units (contra CPUs) and a shallower memory hierarchy, and (c) FPGAs, which offer much more parallelism compared to both CPUs and GPUs, with finer control over individual unit types (here shown in different controls), but is harder to use.

Pechan et al. [79] evaluated and compared the use of FPGAs against both GPU and CPU solutions of the popular Autodock software. They created a custom RTL-based three-stage FPGA accelerator that computes the performance-critical sections of the Autodock

algorithm. More specifically, the custom accelerator has four modules capable of exploiting MLP (see Section 4.1), while LLP is exploited inside each module (through pipelining); the accelerator relies on other methods to exploit HLP. They compared their solution against a custom (CUDA-based) GPU solution (GT220 and GTX260) and a CPU (Intel Xeon 3.2 GHz) version on the 1hvr and 2cpp protein pairs (from the Protein Data Bank). The overall results showed that FPGAs outperformed the CPUs for both use-cases independent of the number of dockings that were used. The GPU, however, had a clear advantage when a large number of dockings were executed, and the FPGA was only preferable when a few number of docking runs were executed. Recent work by Solis-Vasquez et al. [80,81] focused particularly on using OpenCL HLS tools to create custom accelerators that target FPGAs. Aside from disseminating their design process, they also vary several different architectural properties in their accelerator. For example, they consider both floating-point and fixed-point representation for various phases of the computation, which demonstrates an advantage that FPGAs can provide over more general-purpose approaches. The accelerator runs at a fairly high frequency (between 172 MHz and 215 MHz) on a Intel Arria 10, and consumes a varying amount of resources (subject to their design-space exploration). They compare their accelerator against the single-threaded Autodock software on five protein targets, and show that they reach between $1.73\times$ and $2.77\times$ speed up.

Today, there is a remarkably small number of published work that leverages FPGAs in the Autodock software (for surveys using FPGAs on other molecular algorithms, see [82,83]). Even more surprising is that (to the authors' knowledge) CGRAs have been largely unexplored in this domain. With both FPGAs and CGRAs emerging as performance (and, more importantly, *greener*) alternatives to traditional CPUs and GPUs, we believe that these systems will come to play a much larger role in molecular docking and virtual screening in the future than they have so far.

7. The Advent of Quantum Computing for Molecular Docking

With the advent of publicly available quantum computers via cloud computing, such as the IBM, Righetti, Google, and D-Wave quantum systems [84], quantum computing is becoming a promising approach to support and accelerate molecular docking computations.

A study of a molecular docking implementation on a photonic quantum computer was presented in [85]. The authors used a Gaussian boson sampler (GBS) which is a special model of photonic quantum computer where the computation is realized via the interference of identical photons that are passing through a circuit or a network of beam splitters and phase shifters. In this work, the binding interaction graph between ligands and receptor is used to generate the ligand orientations within the protein pocket. A simplified pharmacophore representation is used, limiting the graph size from all-molecular model of the ligand and receptor to a set of points having large influence on the interactions, i.e., negative/positive-charged atoms, hydrogen bond donor/acceptor atoms, hydrophobic characteristics, and aromatic ring positions. The docking problem has been formalized by mapping it to the identification of large clusters in a weighted graph. The GBS device was used to search for the largest cliques while considering the graph weights. The method shows very good results compared to solving the same problem in a classical way; however, it cannot be used alone in a virtual screening process unless to pair it with classical data postprocessing techniques (scoring) thus generating a hybrid-quantum approach.

The usage of quantum annealers to understand the capabilities of these devices to improve the quality and the throughput of molecular docking methods is presented in [86]. In particular, the paper focused on a specific phase of the molecular docking, consisting of ligand manipulation in terms of its rotatable bonds. The authors propose a quantum annealing approach to molecular docking by formulating it as a high-order unconstrained binary optimization (HUBO), which was possible to solve on the latest D-Wave annealing hardware (2000Q and Advantage). The work demonstrated how a lot of simplifications have to be taken into consideration during the problem formulation and embedding phase, even with small molecules. The results show that despite that the current hardware is

not yet mature to solve the molecular docking problem on real-life scales, there is a clear positive trend in that direction.

8. Conclusions

The parallel implementation virtual screening algorithms in massively parallel computers with multiple CPUs and/or GPUs have the high potential to speed up the exploration of gigantic chemical spaces (having compounds in the range 10^9 to 10^{12}) in real time. In a serial version of virtual screening software, it may take many years of CPU hours for such tasks. The current regard for gigantic docking is the screening of billions of compounds from ZINC15 and Enamine databases with the use of Autodock GPU in Summit HPC computers in less than a day. The parallel implementations and reliable scoring functions will increase the success rates in the lead compounds identification for drug discovery. This makes the drug discovery less time-consuming and economically sustainable. Further, as the chemical spaces are really huge, the drugs with entirely different scaffold geometry can be identified. The speed-up of the virtual screening software is found to be dependent on the number of factors: energy minimization algorithm, scoring function, biomolecular target, and computer architecture. More elaborate studies will allow us to develop highly optimized virtual screening software in the future. The implementation of VS for FPGAs and quantum computing is still in its infancy, and a dedicated research is needed for adopting such architectures for drug discovery projects.

Author Contributions: Conceptualization, N.A.M. and S.M.; writing—original draft preparation, N.A.M., A.P. and S.M.; writing—review and editing, N.A.M., A.P., D.G., E.V., G.P. and S.M.; funding acquisition, S.M. All authors have read and agreed to the published version of the manuscript.

Funding: Funding for the work is received from the European Commission H2020 program, Grant Agreement No. 801039 (EPiGRAM-HS). We also acknowledge funding from EuroHPC-JU under grant agreement No 956137 (LIGATE) and European Commission H2020 program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data has been presented in main text.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADMET	Absorption, Distribution, Metabolism, Excretion, Toxicity
CUDA	Compute Unified Device Architecture
CGRA	Coarse-Grained Reconfigurable Arrays
GPU	Graphical processing unit
FPGA	Field programmable gate array
HPC	High-Performance Computing
OS	Operating system
PDB	Protein data bank
RMSD	Root Mean Square Deviation
VS	Virtual screening

References

1. Clark, D.E. What has virtual screening ever done for drug discovery? *Expert Opin. Drug Discov.* **2008**, *3*, 841–851. [[CrossRef](#)] [[PubMed](#)]
2. Morgan, S.; Grootendorst, P.; Lexchin, J.; Cunningham, C.; Greyson, D. The cost of drug development: A systematic review. *Health Policy* **2011**, *100*, 4–17. [[CrossRef](#)] [[PubMed](#)]
3. Dickson, M.; Gagnon, J.P. The cost of new drug discovery and development. *Discov. Med.* **2009**, *4*, 172–179.
4. Mullin, R. Drug development costs about \$1.7 billion. *Chem. Eng. News* **2003**, *81*, 8. [[CrossRef](#)]

5. Rickels, W.; Dovern, J.; Hoffmann, J.; Quaas, M.F.; Schmidt, J.O.; Visbeck, M. Indicators for monitoring sustainable development goals: An application to oceanic development in the European Union. *Earths Future* **2016**, *4*, 252–267. [[CrossRef](#)]
6. Petrova, E. Innovation in the pharmaceutical industry: The process of drug discovery and development. In *Innovation and Marketing in the Pharmaceutical Industry*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 19–81.
7. Kiriiri, G.K.; Njogu, P.M.; Mwangi, A.N. Exploring different approaches to improve the success of drug discovery and development projects: A review. *Future J. Pharm. Sci.* **2020**, *6*, 27. [[CrossRef](#)]
8. Frazier, K. *Biopharmaceutical Research & Development: The Process Behind New Medicines*; PhRMA: Washington, DC, USA, 2015.
9. Doman, T.N.; McGovern, S.L.; Witherbee, B.J.; Kasten, T.P.; Kurumbail, R.; Stallings, W.C.; Connolly, D.T.; Shoichet, B.K. Molecular docking and high-throughput screening for novel inhibitors of protein tyrosine phosphatase-1B. *J. Med. Chem.* **2002**, *45*, 2213–2221. [[CrossRef](#)]
10. Li, H.; Sze, K.H.; Lu, G.; Ballester, P.J. Machine-learning scoring functions for structure-based drug lead optimization. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2020**, *10*, e1465. [[CrossRef](#)]
11. DesJarlais, R.L.; Sheridan, R.P.; Seibel, G.L.; Dixon, J.S.; Kuntz, I.D.; Venkataraghavan, R. Using shape complementarity as an initial screen in designing ligands for a receptor binding site of known three-dimensional structure. *J. Med. Chem.* **1988**, *31*, 722–729. [[CrossRef](#)]
12. Morris, G.M.; Goodsell, D.S.; Halliday, R.S.; Huey, R.; Hart, W.E.; Belew, R.K.; Olson, A.J. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. *J. Comput. Chem.* **1998**, *19*, 1639–1662. [[CrossRef](#)]
13. Cosconati, S.; Forli, S.; Perryman, A.L.; Harris, R.; Goodsell, D.S.; Olson, A.J. Virtual screening with AutoDock: Theory and practice. *Expert Opin. Drug Discov.* **2010**, *5*, 597–607. [[CrossRef](#)]
14. Trott, O.; Olson, A.J. AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J. Comput. Chem.* **2010**, *31*, 455–461. [[CrossRef](#)]
15. Jaghoori, M.M.; Bleijlevens, B.; Olabariaga, S.D. 1001 Ways to run AutoDock Vina for virtual screening. *J. Comput.-Aided Mol. Des.* **2016**, *30*, 237–249. [[CrossRef](#)]
16. McNutt, A.T.; Francoeur, P.; Aggarwal, R.; Masuda, T.; Meli, R.; Ragoza, M.; Sunseri, J.; Koes, D.R. GNINA 1.0: Molecular docking with deep learning. *J. Cheminform.* **2021**, *13*, 1–20. [[CrossRef](#)] [[PubMed](#)]
17. Miller, B.R., III; McGee, T.D., Jr.; Swails, J.M.; Homeyer, N.; Gohlke, H.; Roitberg, A.E. MMPBSA.py: An efficient program for end-state free energy calculations. *J. Chem. Theory Comput.* **2012**, *8*, 3314–3321. [[CrossRef](#)] [[PubMed](#)]
18. Poongavanam, V.; Namasivayam, V.; Vanangamudi, M.; Al Shamaileh, H.; Veedu, R.N.; Kihlberg, J.; Murugan, N.A. Integrative approaches in HIV-1 non-nucleoside reverse transcriptase inhibitor design. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2018**, *8*, e1328. [[CrossRef](#)]
19. Dittrich, J.; Schmidt, D.; Pflieger, C.; Gohlke, H. Converging a knowledge-based scoring function: DrugScore2018. *J. Chem. Inf. Model.* **2018**, *59*, 509–521. [[CrossRef](#)] [[PubMed](#)]
20. Beccari, A.R.; Cavazzoni, C.; Beato, C.; Costantino, G. *LiGen: A High Performance Workflow for Chemistry Driven De Novo Design*; ACS Publications: Washington, DC, USA, 2013.
21. Nayeem, A.; Vila, J.; Scheraga, H.A. A comparative study of the simulated-annealing and Monte Carlo-with-minimization approaches to the minimum-energy structures of polypeptides: [Met]-enkephalin. *J. Comput. Chem.* **1991**, *12*, 594–605. [[CrossRef](#)]
22. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
23. Bursulaya, B.D.; Totrov, M.; Abagyan, R.; Brooks, C.L. Comparative study of several algorithms for flexible ligand docking. *J. Comput.-Aided Mol. Des.* **2003**, *17*, 755–763. [[CrossRef](#)]
24. Vieira, T.F.; Sousa, S.F. Comparing AutoDock and Vina in ligand/decoy discrimination for virtual screening. *Appl. Sci.* **2019**, *9*, 4538. [[CrossRef](#)]
25. Walters, W.P. Virtual chemical libraries: Miniperspective. *J. Med. Chem.* **2018**, *62*, 1116–1124. [[CrossRef](#)]
26. Graff, D.E.; Shakhnovich, E.I.; Coley, C.W. Accelerating high-throughput virtual screening through molecular pool-based active learning. *Chem. Sci.* **2021**, *12*, 7866–7881. [[CrossRef](#)] [[PubMed](#)]
27. Ruddigkeit, L.; Van Deursen, R.; Blum, L.C.; Reymond, J.L. Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. *J. Chem. Inf. Model.* **2012**, *52*, 2864–2875. [[CrossRef](#)] [[PubMed](#)]
28. Sterling, T.; Irwin, J.J. ZINC 15—Ligand discovery for everyone. *J. Chem. Inf. Model.* **2015**, *55*, 2324–2337. [[CrossRef](#)] [[PubMed](#)]
29. Lyu, J.; Wang, S.; Balius, T.E.; Singh, I.; Levit, A.; Moroz, Y.S.; O'Meara, M.J.; Che, T.; Alga, E.; Tolmachova, K.; et al. Ultra-large library docking for discovering new chemotypes. *Nature* **2019**, *566*, 224–229. [[CrossRef](#)]
30. Shivanyuk, A.; Ryabukhin, S.; Tolmachev, A.; Bogolyubsky, A.; Mykytenko, D.; Chupryna, A.; Heilman, W.; Kostyuk, A. Enamine real database: Making chemical diversity real. *Chem. Today* **2007**, *25*, 58–59.
31. Kim, S.; Thiessen, P.A.; Bolton, E.E.; Chen, J.; Fu, G.; Gindulyte, A.; Han, L.; He, J.; He, S.; Shoemaker, B.A.; et al. PubChem substance and compound databases. *Nucleic Acids Res.* **2016**, *44*, D1202–D1213. [[CrossRef](#)]
32. Williams, A.J.; Tkachenko, V.; Golotvin, S.; Kidd, R.; McCann, G. ChemSpider-building a foundation for the semantic web by hosting a crowd sourced databasing platform for chemistry. *J. Cheminform.* **2010**, *2*, O16. [[CrossRef](#)]
33. Gaulton, A.; Bellis, L.J.; Bento, A.P.; Chambers, J.; Davies, M.; Hersey, A.; Light, Y.; McGlinchey, S.; Michalovich, D.; Al-Lazikani, B.; et al. ChEMBL: A large-scale bioactivity database for drug discovery. *Nucleic Acids Res.* **2012**, *40*, D1100–D1107. [[CrossRef](#)]
34. Martin, Y.C. Accomplishments and challenges in integrating software for computer-aided ligand design in drug discovery. *Perspect. Drug Discov. Des.* **1995**, *3*, 139–150. [[CrossRef](#)]

35. Martin, Y.C. 3D database searching in drug design. *J. Med. Chem.* **1992**, *35*, 2145–2154. [[CrossRef](#)]
36. Glaser, J.; Vermaas, J.V.; Rogers, D.M.; Larkin, J.; LeGrand, S.; Boehm, S.; Baker, M.B.; Scheinberg, A.; Tillack, A.F.; Thavappiragasam, M.; et al. High-throughput virtual laboratory for drug discovery using massive datasets. *Int. J. High Perform. Comput. Appl.* **2021**, *35*, 452–468. [[CrossRef](#)]
37. Gorgulla, C.; Boeszoermenyi, A.; Wang, Z.F.; Fischer, P.D.; Coote, P.W.; Das, K.M.P.; Malets, Y.S.; Radchenko, D.S.; Moroz, Y.S.; Scott, D.A.; et al. An open-source drug discovery platform enables ultra-large virtual screens. *Nature* **2020**, *580*, 663–668. [[CrossRef](#)] [[PubMed](#)]
38. LeGrand, S.; Scheinberg, A.; Tillack, A.F.; Thavappiragasam, M.; Vermaas, J.V.; Agarwal, R.; Larkin, J.; Poole, D.; Santos-Martins, D.; Solis-Vasquez, L.; et al. GPU-accelerated drug discovery with docking on the summit supercomputer: Porting, optimization, and application to COVID-19 research. In Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics, Virtual, 21–24 September 2020; pp. 1–10.
39. OpenEye Scientific, GigaDocking™—Structure Based Virtual Screening of over 1 Billion Molecules Webinar. Available online: <https://www.eyesopen.com/> (accessed on 1 August 2020).
40. Acharya, A.; Agarwal, R.; Baker, M.B.; Baudry, J.; Bhowmik, D.; Boehm, S.; Byler, K.G.; Chen, S.; Coates, L.; Cooper, C.J.; et al. Supercomputer-based ensemble docking drug discovery pipeline with application to COVID-19. *J. Chem. Inf. Model.* **2020**, *60*, 5832–5852. [[CrossRef](#)] [[PubMed](#)]
41. Gadioli, D.; Vitali, E.; Ficarelli, F.; Latini, C.; Manelfi, C.; Talarico, C.; Silvano, C.; Cavazzoni, C.; Palermo, G.; Beccari, A.R. EXSCALATE: An extreme-scale in-silico virtual screening platform to evaluate 1 trillion compounds in 60 h on 81 PFLOPS supercomputers. *arXiv* **2021**, arXiv:2110.11644.
42. Dong, D.; Xu, Z.; Zhong, W.; Peng, S. Parallelization of molecular docking: A review. *Curr. Top. Med. Chem.* **2018**, *18*, 1015–1028. [[CrossRef](#)] [[PubMed](#)]
43. Perez-Sanchez, H.; Wenzel, W. Optimization methods for virtual screening on novel computational architectures. *Curr. Comput.-Aided Drug Des.* **2011**, *7*, 44–52. [[CrossRef](#)]
44. Stone, J.E.; Phillips, J.C.; Freddolino, P.L.; Hardy, D.J.; Trabuco, L.G.; Schulten, K. Accelerating molecular modeling applications with graphics processors. *J. Comput. Chem.* **2007**, *28*, 2618–2640. [[CrossRef](#)]
45. Harvey, M.; De Fabritiis, G. An implementation of the smooth particle mesh Ewald method on GPU hardware. *J. Chem. Theory Comput.* **2009**, *5*, 2371–2377. [[CrossRef](#)]
46. Guerrero, G.D.; Pérez-Sánchez, H.; Wenzel, W.; Cecilia, J.M.; García, J.M. Effective parallelization of non-bonded interactions kernel for virtual screening on gpus. In Proceedings of the 5th International Conference on Practical Applications of Computational Biology & Bioinformatics (PACBB 2011), Salamanca, Spain, 6–8 April 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 63–69.
47. Moustakas, D.T.; Lang, P.T.; Pegg, S.; Pettersen, E.; Kuntz, I.D.; Brooijmans, N.; Rizzo, R.C. Development and validation of a modular, extensible docking program: DOCK 5. *J. Comput.-Aided Mol. Des.* **2006**, *20*, 601–619. [[CrossRef](#)]
48. Allen, W.J.; Balius, T.E.; Mukherjee, S.; Brozell, S.R.; Moustakas, D.T.; Lang, P.T.; Case, D.A.; Kuntz, I.D.; Rizzo, R.C. DOCK 6: Impact of new features and current docking performance. *J. Comput. Chem.* **2015**, *36*, 1132–1156. [[CrossRef](#)] [[PubMed](#)]
49. Liu, H.Y.; Kuntz, I.D.; Zou, X. Pairwise GB/SA scoring function for structure-based drug design. *J. Phys. Chem. B* **2004**, *108*, 5453–5462. [[CrossRef](#)]
50. Hawkins, G.D.; Cramer, C.J.; Truhlar, D.G. Parametrized models of aqueous free energies of solvation based on pairwise descreening of solute atomic charges from a dielectric medium. *J. Phys. Chem.* **1996**, *100*, 19824–19839. [[CrossRef](#)]
51. Grant, J.A.; Pickup, B.T.; Nicholls, A. A smooth permittivity function for Poisson–Boltzmann solvation methods. *J. Comput. Chem.* **2001**, *22*, 608–640. [[CrossRef](#)]
52. Mukherjee, S.; Balius, T.E.; Rizzo, R.C. Docking validation resources: Protein family and ligand flexibility experiments. *J. Chem. Inf. Model.* **2010**, *50*, 1986–2000. [[CrossRef](#)] [[PubMed](#)]
53. Yang, H.; Zhou, Q.; Li, B.; Wang, Y.; Luan, Z.; Qian, D.; Li, H. GPU acceleration of Dock6’s Amber scoring computation. In *Advances in Computational Biology*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 497–511.
54. Jiang, X.; Kumar, K.; Hu, X.; Wallqvist, A.; Reifman, J. DOVIS 2.0: An efficient and easy to use parallel virtual screening tool based on AutoDock 4.0. *Chem. Cent. J.* **2008**, *2*, 18. [[CrossRef](#)]
55. Prakhov, N.D.; Chernorudskiy, A.L.; Gainullin, M.R. VSDocker: A tool for parallel high-throughput virtual screening using AutoDock on Windows-based computer clusters. *Bioinformatics* **2010**, *26*, 1374–1375. [[CrossRef](#)]
56. Koes, D.R.; Baumgartner, M.P.; Camacho, C.J. Lessons learned in empirical scoring with smina from the CSAR 2011 benchmarking exercise. *J. Chem. Inf. Model.* **2013**, *53*, 1893–1904. [[CrossRef](#)]
57. Norgan, A.P.; Coffman, P.K.; Kocher, J.P.A.; Katzmann, D.J.; Sosa, C.P. Multilevel parallelization of AutoDock 4.2. *J. Cheminform.* **2011**, *3*, 1–9. [[CrossRef](#)]
58. Zhang, X.; Wong, S.E.; Lightstone, F.C. Message passing interface and multithreading hybrid for parallel molecular docking of large databases on petascale high performance computing machines. *J. Comput. Chem.* **2013**, *34*, 915–927. [[CrossRef](#)]
59. Ellingson, S.R.; Smith, J.C.; Baudry, J. VinaMPI: Facilitating multiple receptor high-throughput virtual docking on high-performance computers. *J. Comput. Chem.* **2013**, *34*, 2212–2221. [[CrossRef](#)] [[PubMed](#)]
60. Beato, C.; Beccari, A.R.; Cavazzoni, C.; Lorenzi, S.; Costantino, G. Use of experimental design to optimize docking performance: The case of ligendock, the docking module of ligen, a new de novo design program. *J. Chem. Inf. Model.* **2013**, *53*, 1503–1517. [[CrossRef](#)] [[PubMed](#)]

61. Gadioli, D.; Palermo, G.; Cherubin, S.; Vitali, E.; Agosta, G.; Manelfi, C.; Beccari, A.R.; Cavazzoni, C.; Sanna, N.; Silvano, C. Tunable approximations to control time-to-solution in an HPC molecular docking Mini-App. *J. Supercomput.* **2021**, *77*, 841–869. [[CrossRef](#)]
62. Vitali, E.; Gadioli, D.; Palermo, G.; Beccari, A.; Silvano, C. Accelerating a geometric approach to molecular docking with OpenACC. In Proceedings of the 6th International Workshop on Parallelism in Bioinformatics, Barcelona, Spain, 23 September 2018; pp. 45–51.
63. Markidis, S.; Gadioli, D.; Vitali, E.; Palermo, G. Understanding the I/O Impact on the Performance of High-Throughput Molecular Docking. In Proceedings of the IEEE/ACM Sixth International Parallel Data Systems Workshop (PDSW), St. Louis, MO, USA, 15 November 2021.
64. Fang, Y.; Ding, Y.; Feinstein, W.P.; Koppelman, D.M.; Moreno, J.; Jarrell, M.; Ramanujam, J.; Brylinski, M. GeauxDock: Accelerating structure-based virtual screening with heterogeneous computing. *PLoS ONE* **2016**, *11*, e0158898. [[CrossRef](#)]
65. Samdani, A.; Vetrivel, U. POAP: A GNU parallel based multithreaded pipeline of open babel and AutoDock suite for boosted high throughput virtual screening. *Comput. Biol. Chem.* **2018**, *74*, 39–48. [[CrossRef](#)]
66. Santos-Martins, D.; Solis-Vasquez, L.; Tillack, A.F.; Sanner, M.F.; Koch, A.; Forli, S. Accelerating AutoDock4 with GPUs and gradient-based local search. *J. Chem. Theory Comput.* **2021**, *17*, 1060–1073. [[CrossRef](#)] [[PubMed](#)]
67. Hartshorn, M.J.; Verdonk, M.L.; Chessari, G.; Brewerton, S.C.; Mooij, W.T.; Mortenson, P.N.; Murray, C.W. Diverse, high-quality test set for the validation of protein-ligand docking performance. *J. Med. Chem.* **2007**, *50*, 726–741. [[CrossRef](#)] [[PubMed](#)]
68. Li, Y.; Han, L.; Liu, Z.; Wang, R. Comparative assessment of scoring functions on an updated benchmark: 2. Evaluation methods and general results. *J. Chem. Inf. Model.* **2014**, *54*, 1717–1736. [[CrossRef](#)]
69. Lyne, P.D. Structure-based virtual screening: An overview. *Drug Discov. Today* **2002**, *7*, 1047–1055. [[CrossRef](#)]
70. Marze, N.A.; Roy Burman, S.S.; Sheffler, W.; Gray, J.J. Efficient flexible backbone protein–protein docking for challenging targets. *Bioinformatics* **2018**, *34*, 3461–3469. [[CrossRef](#)]
71. Jain, A.N. Surflex: Fully automatic flexible molecular docking using a molecular similarity-based search engine. *J. Med. Chem.* **2003**, *46*, 499–511. [[CrossRef](#)] [[PubMed](#)]
72. Taylor, P.; Blackburn, E.; Sheng, Y.; Harding, S.; Hsin, K.Y.; Kan, D.; Shave, S.; Walkinshaw, M. Ligand discovery and virtual screening using the program LIDAEUS. *Br. J. Pharmacol.* **2008**, *153*, S55–S67. [[CrossRef](#)]
73. Capuccini, M.; Ahmed, L.; Schaal, W.; Laure, E.; Spjuth, O. Large-scale virtual screening on public cloud resources with Apache Spark. *J. Cheminform.* **2017**, *9*, 15. [[CrossRef](#)] [[PubMed](#)]
74. Vetter, J.S.; DeBenedictis, E.P.; Conte, T.M. Architectures for the post-Moore era. *IEEE Micro* **2017**, *37*, 6–8. [[CrossRef](#)]
75. Podobas, A.; Sano, K.; Matsuo, S. A survey on coarse-grained reconfigurable architectures from a performance perspective. *IEEE Access* **2020**, *8*, 146719–146743. [[CrossRef](#)]
76. Nane, R.; Sima, V.M.; Pilato, C.; Choi, J.; Fort, B.; Canis, A.; Chen, Y.T.; Hsiao, H.; Brown, S.; Ferrandi, F.; et al. A survey and evaluation of FPGA high-level synthesis tools. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2015**, *35*, 1591–1604. [[CrossRef](#)]
77. Czajkowski, T.S.; Aydonat, U.; Denisenko, D.; Freeman, J.; Kinsner, M.; Neto, D.; Wong, J.; Yiannacouras, P.; Singh, D.P. From OpenCL to high-performance hardware on FPGAs. In Proceedings of the 22nd international conference on field programmable logic and applications (FPL), Oslo, Norway, 29–31 August 2012; pp. 531–534.
78. Podobas, A. Accelerating parallel computations with openmp-driven system-on-chip generation for fpgas. In Proceedings of the IEEE 8th International Symposium on Embedded Multicore/Manycore SoCs, Aizu-Wakamatsu, Japan, 23–25 September 2014; pp. 149–156.
79. Pechan, I.; Fehér, B.; Bérces, A. FPGA-based acceleration of the AutoDock molecular docking software. In Proceedings of the 6th Conference on Ph.D. Research in Microelectronics & Electronics, Berlin, Germany, 18–21 July 2010; pp. 1–4.
80. Solis-Vasquez, L.; Koch, A. A case study in using opencl on fpgas: Creating an open-source accelerator of the autodock molecular docking software. In Proceedings of the FSP Workshop 2018; Fifth International Workshop on FPGAs for Software Programmers, Dublin, Ireland, 31 August 2018; pp. 1–10.
81. Solis Vasquez, L. Accelerating Molecular Docking by Parallelized Heterogeneous Computing—A Case Study of Performance, Quality of Results, and Energy-Efficiency Using CPUs, GPUs, and FPGAs. Ph.D. Thesis, Technische Universität, Darmstadt, Germany, 2019. [[CrossRef](#)]
82. Majumder, T.; Pande, P.P.; Kalyanaraman, A. Hardware accelerators in computational biology: Application, potential, and challenges. *IEEE Des. Test* **2013**, *31*, 8–18. [[CrossRef](#)]
83. Pechan, I.; Fehér, B. Hardware Accelerated Molecular Docking: A Survey. *Bioinformatics*, Horacio Pérez-Sánchez, IntechOpen, 2012, Volume 133. Available online: <https://www.intechopen.com/chapters/41236> (accessed on 1 August 2020). [[CrossRef](#)]
84. Castelvecchi, D. Quantum computers ready to leap out of the lab in 2017. *Nat. News* **2017**, *541*, 9. [[CrossRef](#)]
85. Banchi, L.; Fingerhuth, M.; Babej, T.; Ing, C.; Arrazola, J.M. Molecular docking with Gaussian boson sampling. *Sci. Adv.* **2020**, *6*, eaax1950. [[CrossRef](#)]
86. Mato, K.; Mengoni, R.; Ottaviani, D.; Palermo, G. Quantum Molecular Unfolding. *arXiv* **2021**, arXiv:2107.13607.