# The role of High Performance Computing in Bioinformatics

Horacio Peréz-Sánchez, José M. Cecilia, and Ivan Merelli

Bioinformatics and High Performance Computing Research Group (BIO-HPC)
Computer Science Department
Universidad Católica San Antonio de Murcia (UCAM), Guadalupe E30107, Spain
{hperez,jmcecilia}@ucam.edu
Institute for Biomedical Technologies
National Research Council of Italy
Milano, Italy
ivan.merelli@itb.cnr.it

**Abstract.** We are witnessing one of the major revolutions in parallel systems. The consolidation of heterogeneous systems at different levels -from desktop computers to large-scale systems such as supercomputers, clusters or grids, through all kinds of low-power devices- is providing a computational power unimaginable just few years ago, trying to follow the wake of Moore's law. This landscape in the high performance computing arena opens up great opportunities in the simulation of relevant biological systems and for applications in Bioinformatics, Computational Biology and Computational Chemistry. This introductory article shows the last tendencies of this active research field and our perspectives for the forthcoming years.

**Keywords:** HPC, Parallel Computing, Heterogeneous Computing, Bioinformatics, Computational Biology

## 1 Introduction

The integration of the latest breakthroughs in biochemistry and biotechnology from one side and high performance computing and computational modelling from the other, enables remarkable advances in the fields of healthcare, drug discovery, genome research, systems biology and so on. By integrating all these developments together, scientists are creating new exciting personal therapeutic strategies for living longer and having healthier lifestyles that were unfeasible not that long ago.

Those efforts have created new research fields such as *Bioinformatics and Computational Biology*, defined most broadly as informatics in the domains of biology and biomedical research. Bioinformatics spans to many different research areas, such as life sciences, where there are many examples of scientific applications for discovering biological and medical unknown factors that could greatly benefit from increased computational resources.

Indeed, as computing resources available on current systems are limited, this limitation becomes a serious constraint, then hindering it from successfully taking the next step forward. For instance, applications such programs of Molecular Dynamics (MD) [1], employed to analyse the dynamical properties of macromolecules such as folding and allosteric regulations, or software used for solving atom-to-atom interactions for drug discovery, such as AutoDock [2] and FlexScreen [3], could clearly benefit from enhanced computing capabilities and also from some novel algorithms approaches like those inspired by nature such as Genetic algorithms [4] or Ant Colony Optimization techniques [5].

There are other applications, which are actually working well, but their execution is too slow for providing feedback in real-time to the users, and thus limiting the effectiveness and comfort of such applications. In this group, we may cite biomedical image processing applications, such as X-ray computed tomography or mammography for breast cancer detection. The low-performance of these applications can drastically affect the patient's health. For instance, imagine a patient who is waiting for mammography results. She is actually waiting for a breast cancer diagnostic, thus the acceleration of the diagnosis time becomes paramount for the patient's health.

High Performance Computing technologies are at the forefront of those revolutions, making it possible to carry out and accelerate radical biological and medical breakthroughs that would directly translate into real benefits for the society and the environment. In this regard, Graphics Processing Units (GPUs) are providing a unique opportunity to tremendously increase the effective computational capability of the commodity PCs, allowing desktop supercomputing at very low prices. Moreover, large clusters are adopting the use of these relatively inexpensive and powerful devices as a way of accelerating parts of the applications they are running. Since June 2011, when the fastest supercomputer was the Tianhe-1A, placed in the National Supercomputing Centre at Tianjin (China)[1], including up to 7.168 NVIDIA® Tesla™ M2050 general purpose GPUs, several supercomputers have followed this trend.

However, the inclusion of those accelerators in the system has a great impact on the power consumption of the system, as a high-end GPU may increase the power consumption of a cluster node up to 30% which is actually a big issue. This is a critical concern especially for very large data centres, where the cost dedicated to supply power to such computers represents an important fraction of the Total Cost of Ownership (TCO) [6]. The research community is also aware of this and it is making efforts in striving to develop reduced-power installations. Thus, the GREEN500 list[2] shows the 500 most power efficient computers in the world. In this way, we can see a clear shift from the traditional metric FLOPS (FLoating point Operations Per Second) to FLOPS per watt.

Virtualization techniques may provide significant energy savings, as they enable a larger resource usage by sharing a given hardware among several users, thus reducing the required amount of instances of that particular device. As a

[1] http://www.top500.org/
[2] http://www.green500.org/

result, virtualization is being increasingly adopted in data centres. In particular, cloud computing is an inherently energy-efficient virtualization technique [7], in which services run remotely in a ubiquitous computing cloud that provides scalable and virtualized resources. Thus peak loads can be moved to other parts of the cloud and the aggregation of a cloud's resources can provide higher hardware utilization [8]. Public cloud providers offer their services in a *pay as you go* fashion, and provide an alternative to physical infrastructures. However, this alternative only becomes real for a specific amount of data and target execution time.

The rest of the paper is organized as follows. We briefly introduce some HPC architectures we think are at the forefront of this revolution in Section 2. In Section 3 we present some relevant Bioinformatics applications that are using HPC alternatives to deal with their computational issues before we summarize our findings and conclude with suggestions for future work.

## 2 HPC Architectures

Traditionally, high performance computing has been employed for addressing bioinformatics problems that would otherwise be impossible to solve. A first example was the preliminary assembly of the human genome, a huge effort in which the Human Genome Project was challenged by Celera Genomics with a different approach, consisting in a bioinformatics post analysis of whole sets of shotgun sequencing runs instead of the long-standing vector cloning technique, arriving very close to an unpredictable victory [9].

In this Section, we briefly summarize the high performance systems that has been commonly used in Bioinformatics. Among them, we highlight throughput-oriented architectures such as GPUs, large-scale heterogeneous clusters and clouds or distributed computing systems, with a discussion about how the latest breakthroughs in these architectures are being used in this field.

### 2.1 GPU computing

Driven by the demand of the game industry, graphics processing units (GPUs) have completed a steady transition from mainframes to workstations to PC cards, where they emerge nowadays like a solid and compelling alternative to traditional computing platforms. GPUs deliver extremely high floating point performance and massively parallelism at a very low cost, thus promoting a new concept of the high performance computing (HPC) market; i.e. *heterogeneous computing* where processors with different characteristics work together to enhance the application performance taking care of the power budget. This fact has attracted many researchers and encouraged the use of GPUs in a broader range of applications, particularly in the field of Bioinformatics, where developers are required to leverage this new landscape of computation with new programming models which ease the developers task of writing programs to run efficiently on such platforms altogether [10].

The most popular microprocessor companies such as NVIDIA, ATI/AMD or Intel, have developed hardware products aimed specifically at the heterogeneous or massively parallel computing market: Tesla products are from NVIDIA, Firestream is AMDs product line and Intel Xeon Phi comes from Intel. They have also released software components, which provide simpler access to this computing power. CUDA (Compute Unified Device Architecture) is NVIDIAs solution as a simple block-based API for programming; AMDs alternative was called Stream Computing and Intel relies on X86-based programming. More recently (in 2008), the OpenCL[3] emerged as an attempt to unify all of those models with a superset of features, being the best broadly supported multi-platform data-parallel programming interface for heterogeneous computing, including GPUs, accelerators and similar devices.

Although these efforts in developing programming models have made great contributions to leverage the capabilities of these platforms, developers have to deal with a massively parallel and high throughput-oriented architecture[11], which is quite different than traditional computing architectures. Moreover, GPUs are being connected with CPUs through PCI Express bus to build heterogeneous parallel computers, presenting multiple independent memory spaces, a wide spectrum of high speed processing functions, and communication latency between them. These issues drastically increase scaling to a GPU-cluster, bringing additional sources of latency. Therefore, programmability on these platforms is still a challenge, and thus many research efforts have provided abstraction layers avoiding to deal with the hardware particularities of these accelerators and also extracting transparently high level of performance, providing portability across operating systems, host CPUs and accelerators. For example, libraries interfaces for programming with popular programming languages like OMPSs for OpenMP [4] or OpenACC[5] API, which describes a collection of compiler directives to specify loops and regions of code in standard programming language such as C, C++ or Fortran.

### 2.2 Supercomputers

Many of the supercomputers in the TOP500 list are heavily involved in computational biology research. For example Titan, the fastest system in the TOP500 in the list of November 2013, works for providing a molecular description of membrane fusion, one of the most common ways for molecules to enter or exit from living cells. Looking at the top ten supercomputers of this latest TOP500 list, the SuperMUC cluster, installed at the Leibniz Supercomputer Centre in Monaco, is often employed in bioinformatics, for example in analysis of linkage disequilibrium in genotyping and Piz Daint, installed at the CSCS/Swiss Bioinformatics Institute in Lugano, has been successfully employed for a challenge of evolutionary genomics, for calculating selection events in genes many times more quickly.

[3] http://www.khronos.org/opencl/
[4] http://openmp.org/wp/
[5] http://www.openacc-standard.org/

Looking at the November 2013 list, it is very interesting to see that the two top supercomputers in the word make use of co-processors to improve their performance. In particular, Tianhe-2 has more then 16.000 nodes composed by two Processor Intel Xeon E5 and three Coprocessor Intel Xeon Phi 31S1, while Titan uses NVDIA K20 cards to improve its performance. Notably, in the top ten supercomputers, four make use of co-processors to enhance their performance. This choice should also be analysed in the view of the power consumption of these supercomputers: Tianhe-2 has a declared power consumption of 17GW for a total of 33 PetaFLOPS, while Titan has a power consumption of 8GW for a total of 17 PetaFLOPS. For example, the fourth supercomputer, K, installed at the Riken Institute of Japan, has a power consumption of 12GW for 10.5 PetaFLOPS. The energy saving using co-processors is therefore clear.

Concerning performance, *virtual clusters* should be also considered as very reliable in this cloud era: for example a virtual infrastructure of 17024 cores built using a set of Amazon Elastic Cloud Computing virtual machines was able to achieve 240.09 TeraFLOPS for the High Performance Linpack benchmark, placing the cluster at position 102 in the November 2011 Top500 list. A similar example was performed on Windows Azure, bringing together 8064 cores for a total of 151.3 TeraFLOPS, a virtual cluster that reached position 165 in in the November 2011 Top500 list.

### 2.3 Grid and Cloud computing

In current bioinformatics, the cost of buying and maintaining an in-house cluster is very important and this explains why the grid computing paradigm gained a great success in the mid-1990s [12]. The problem is that even if the low-level details of the grid infrastructures are hidden via middlewares, often the application of bioinformatics methods on HPC facilities requires specialized knowledge. A suitable solution to offer easy-to-use and intuitive access to applications are science gateways, which offer specific services tailored to the users needs. Nonetheless, ten years later, cloud computing was presented as a more flexible solution with respect to grid [13]. Cloud computing overcomes the idea of volunteer computing for resource sharing by proposing an *on-demand* paradigm in which users pay for what they use. Cloud computing providers offer their services according to several fundamental models: infrastructure as a service (IaaS), platform as a service (PaaS), software as a service (SaaS), and Data as a service (DaaS), where IaaS is the most basic model while the other provide higher level of abstraction [14].

By instantiating many virtual resources a parallel cluster can be deployed on demand, where common libraries such as the Message Passing Interface (MPI) can be exploited. Also batch-processing systems can be used to manage the different computations in a queue. The flexibility and the cost-effectiveness provided by cloud computing is extremely appealing for computational biology, in particular for small-medium biotechnology laboratories which need to perform bioinformatics analysis without addressing all the issues of having an in-house ICT infrastructure [15]. An intermediate solution is represented by Hybrid Clouds

that couple the scalability offered by general-purpose public clouds with the greater control and ad-hoc customizations supplied by the private ones [16].

Moreover, frameworks for distributed access to files such as Hadoop can be adapted to distributed programming paradigms such as MapReduce [17]. For example, the Crossbow [18] genotyping program leverages the Hadoop implementation of MapReduce to launch many copies of the short-read aligner Bowtie [19] in parallel. After Bowtie has aligned the reads (which can be billions for a human re-sequencing project) to the reference genome, Hadoop automatically sorts and aggregates the alignments by chromosomal region. It then launches many parallel instances of the Bayesian single-nucleotide polymorphism (SNP) caller SOAPsnp [20] to accurately call SNPs from the alignments.

## 3 Applications

This section describes some bioinformatics applications from the High Performance computing point of view.

### 3.1 Virtual Screening

In this Section, we summarize the main technical contributions for the parallelization of Virtual Screening (VS) methods on GPUs available on the bibliography. Concretely, we pay special attention to the parallelization of docking methods on GPUs.

In terms of implementations, the trend seems to be reusing available libraries when possible and implement the achievements into existing simulation packages for VS. Among the most-used strategies are either implementing the most time-consuming parts of previously designed codes for serial computers, or redesigning the whole code from scratch. When porting VS methods to GPUs, we should realize that not all methods are equally amenable for optimization. Programmers should check carefully how the code works and whether it is suited for the target architecture. Irrespective of CUDA, most authors maintain that the application will be more accessible in the future thanks to new and promising programming paradigms which are still in the experimental stage or are not yet broadly used. Among them, we may highlight OpenCL or DirectCompute.

**Dock6.2** In the work of Yang et al. [21] a GPU accelerated amber score in Dock6.2 is presented. They report up to 6.5x speedup factor with respect to 3,000 cycles during MD simulation compared to a dual core CPU. [6].

The lack of the single-precision floating point operations in the targeted GPU (NVIDIA GeForce 9800GT) produces small precision losses compared to the CPU, which the authors assume as acceptable. They highlight the thread management utilizing multiple blocks and single transferring of the molecule grids as the main factor that dominates the performance improvements on GPU.

[6] http://dock.compbio.ucsf.edu/DOCK_6/

They use another optimization technique, such as dealing with the latency attributed to thread synchronization, divergence hidden and shared memory through tiling, that authors state may double the speedup of the simulation. We miss a deeper analysis on the device memory bandwidth utilization. It is not clear whether the pattern accesses to device memory in the different versions of the designs presented here are coalesced or not, which may drastically affect the overall performance.

They finally conclude that the speedup of Amber scoring is limited by the Amdahl's law for two main reasons: (1) the rest of the Amber scoring takes a higher percentage of the run time than the portion parallelized on the GPU, and (2) partitioning the work among SMs will eventually decrease the individual job size to a point where the overhead of initializing an SP dominates the application execution time. However, we do not see any clear evaluation that supports these conclusions.

**Autodock** In the paper of Kannan et al. [22] the migration to NVIDIA GPUs of part of the molecular docking application *Autodock* is presented. Concretely, they only focus on the Genetic Algorithm (GA) which is used to find the optimal docking conformation of a ligand with respect to a protein. They use single-precision floating point operation arguing that, "GA depends on relative goodness among individual energies and single precision may not affect the accuracy of GA path significantly". All the data relative to the GA state is maintained on the GPU memory, avoiding data movement through the PCI Express bus.

The GA algorithms need random numbers for the selection process. They decide to generate the random numbers on the CPU instead of doing it on the GPU. The explanation of that is two-fold according to the authors: (1) it enables one-to-one comparisons of CPU and GPU results, and (2) it reduces the design, coding and validation effort of generating random numbers on GPU.

A very nice decision is what the authors call *CGPU Memory Manager* that enables alignment for individual memory request, support for pinned memory and join memory transfer to do all of them in just one transfer. Regarding the fitness function of the GA, authors decide to evaluate all the individuals in a population regardless of modifications. This avoids warp divergences although it makes some redundant work.

Three different parallel design alternatives are discussed in this regard. Two of them only differ in the way they calculate the fitness function, assigning the calculation of the fitness of an individual either to a GPU thread or GPU block. A good comparison between them is provided. The last one includes an extra management of the memory to avoid atomic operations which drastically penalizes the performance.

All of these implementations are rewarded with up to 50x in the fitness calculation, but they do not mention anything about global speedup of the Autodock program.

**Genetic algorithms based docking** In literature is also available [23] an enhanced version of the PLANTS [24] approach for protein-ligand docking using GPUs. They report speedup factors of up to 50x in their GPU implementation compared to an optimized CPU based implementation for the evaluation of interaction potentials in the context of rigid protein. The GPU implementation was carried out using OpenGL to access the GPU's pipeline and Nvidia's Cg language for implementing the shaders programs (i.e. Cg kernels to compute on the GPU). Using this way of programming GPUs, the developing effort is too high, and also some peculiarities of the GPU architecture may be limited. For instance, the authors say that some of the spatial data structures used in the CPU implementation can not directly be mapped to the GPU programming model because of missing support for shared memory operations [23].

The speedup factors observed, especially for small ligands, are limited by several factors. First, only the generation of the ligand-protein conformation and the scoring function evaluation are carried out on the GPU, whereas the optimization algorithm is run on the CPU. This algorithmic decomposition implies time-consuming data transfers through PCI Express bus. The optimization algorithm used in PLANTS is the Ant Colony Optimization (ACO) algorithm [25]. Concretely, authors propose a parallel scheme for this algorithm on a CPU cluster, which use multiple ant colonies in parallel, exchanging information occasionally between them [26]. Developing the ACO algorithm on the GPU as it has been shown in [27] can drastically reduce the communications overhead between CPU and GPU.

### 3.2 Next Generation Sequencing

Next Generation Sequencing data analysis is one of the most demanding application in bioinformatics. Starting from routinely procedure like alignments and variant calling to more complex challenges like genome wide annotations and biomarkers correlation to diseases, NGS analyses are time-consuming and high-performance computing can provide great advantages in this field of genomics, in particular if applied to medicine and healthcare.

Global aligners are very fast, usually tanks to the use of particular data representation approaches, such as the Burrows-Wheeler transform (BWT). Nonetheless, using the option to achieve the optimal result, through the backtracking approach, they are quite slow, despite the use of these reliable representations of data. The problems are even more complex while local alignments are needed. GPU solutions are available in this sense, such as CUSHAW, which is a CUDA compatible short read alignment algorithm for multiple GPUs sharing a single host. This aligner only provides support for ungapped alignment, but in this context has results comparable with BWT-based aligners such as Bowtie and SOAP2. Another example is BarraCUDA, which is directly based on BWA, and delivers a high level of alignment fidelity and is comparable to other mainstream alignment programs. It can perform alignments with gap extensions, in order to minimize the number of false variant calls in re-sequencing studies.

Nonetheless, sometime BLAST is the only option to achieve reasonable results (for example while working on clustering of sequences, for the creation of domain profiles, or for the analysis of chimera sequences). In this case the computation can be extremely time consuming. Although CUDA based solutions exist also for BLAST, the algorithm is complex to implement on GPGPU and the scalabilities achieved are limited. Therefore, also depending on the dimension of the reference database, grid and cloud approaches can still be the only solutions to achieve results in reasonable amount of time. A number of solutions are available, but it is worthy to cite the results obtained on the EGEE/EGI grid platform for sequence clustering [28].

Other issues can be found in genome assembly projects, where the bottlenecks are represented by the huge amount of memory required for the representation of data. Reference assembly is faster and less demanding than de novo, but the problem is the creation of the indexes: in particular some assembly algorithms use read-indexing, while others (the newer ones) usually index the whole genome. While the formers use significantly more memory if the number of reads is higher, the latters use significantly more memory if the genome size is larger. Concerning de novo assembly, the memory consumption is less demanding while using a suffix tree based approach, but really critical while using approaches relying on graphs. This can be solved using a pure parallel approach, for example through the MPI library, such as the one proposed by Abyss (for genome analysis) or Trinity (for transcripts reconstruction). In this case High Performance Computing is more oriented at sharing the memory resources than the CPU power, which is also very important.

But the real problems with NGS is probably the huge amount of data that are produced, in particular while this technique is entering in the clinical practice for analysing personal variations, tumour profiles and gene-therapy safety. Big data represents a huge concern in NGS, because managing terabytes of data requires specific technologies and capabilities (High Availability clusters), redundant facilities (RAID, Backups), shared and distributed file systems (LUSTRE, GPFS), clustered databases (Mysql cluster, Oracle), indexing and searching process (Lucene, HBase), and dedicated network configuration (bridging, bonding). Also security becomes a primarily concern and virtualization should be carefully considered, because, although extremely powerful and flexible, should be certified for its privacy if employed for managing healthcare data. The diverse methods and technologies often require also the re-engineering of applications in the field of bioinformatics. Solutions such as Hadoop and MapReduce for example are nowadays largely employed to treat omics data at large scale.

### 3.3 Molecular Dynamics

In bioinformatics, one of the most successful application of GPU concerns Molecular Dynamics simulations. Molecular Dynamics is certainly the most CPU-demanding application in computational biology, because it consists in solving time step after time step the Newtons equations of motion for all the atoms of a protein, taking as boundary conditions the initial protein structure and a set

of velocity taken from a Gaussian distribution. Molecular Dynamics is often employed in combination to docking screenings, because while VSs are very useful for discarding compounds that clearly do not fit with the identified target, the identification of lead compounds is usually more challenging [29]. The reason is that docking software have errors in computing binding energy in the range of few kcal. Therefore, best compounds achieved through the virtual screening process usually undergone to a protocol of energy refinement implemented using Molecular Dynamics [30].

Indeed, by employing specific simulations schemas and energy decomposition algorithm in the post analysis, Molecular Dynamics allows to achieve more precise quantification of the binding energy [31]. Common techniques for energy estimation are MM-PBSA and MM-GBSA, which consist in the evaluation of the different terms that compose the binding energy taking into account different time point. In particular, it is possible to account the binding energy to the sum of molecular mechanical energies in the gas phase, solvation contribute, evaluated using an implicit solvent model like Generalized Born, or solving the Poisson-Boltzman equation, and the entropic contribute, estimated with normal mode analysis approximation, for example.

Moreover, Molecular Dynamics can be used to predict protein structures (abinitio or refining models computed by homology) or to analyse protein stability (for example verifying what happens in case of mutations). The simulation of proteins can be also very useful to verify the interactions of residues within the macromolecule, for example to clarify why the binding of certain molecule (such as ATP) can change the structure of a particular binding site, a phenomenon that is usually referred as allostery.

The possibility of using NVIDIA cards prompted the use of Molecular Dynamics techniques in computational chemistry and biology researches to new boundaries of discovery, enabling their application in wider range of situations. Compared to CPUs, GPUs run common molecular dynamics, quantum chemistry and visualization applications more than 5x faster. In particular, the team of AMBER has worked very hard to improve the performance of their simulator on GPUs, which is now extremely fast, between 5x and 10x, depending on the number of atoms, the composition of the system and the type of simulation desired [32]. Also GROMACS[7] has been ported on GPUs [33], with very good performance when the implicit solvent is used and performance that are less brilliant in case of explicit solvent.

There are also a lot of tests concerning MD simulations on Intel Xeon Phi, which in theory has the great advantage to run normal x86 code, while for GPUs the software should be re-implemented in a considerable portion. Although this is true, without optimization, the scalability of the Intel Xeon Phi is not as powerful as for the NVIDIA cards [34]. However, by improving the code to obtain an optimal and a balanced vectorization, the achieved performance are consistently faster, up to 10x for the Intel Xeon Phi coprocessor.

[7] http://www.gromacs.org

## 4 Conclusions and outlook

Applications with a real impact on the society, such as those in the field of Bioinformatics and Computational Biology, can take advantage from improvement in high performance computing to overcome computational limitations they have by definition. Those applications are developed to give the opportunity to create new exciting personal therapeutic strategies for living longer and having healthier lifestyles that were unfeasible not that long ago.

This work summarizes the main trends in HPC applied to Bioinformatics. We show several successful stories and application fields, in which relevant biological problems have been solved (or are being targeted) thanks to the computational power available in current processors. We have also pointed out the main drawbacks in the HPC arena, which may limit this good alliance between Bioinformatics and HPC systems. Among them we may highlight power consumption, high learning-curve in emerging programming models to leverage their computational power and the total cost of ownership. We think that the investigations on improvement Bioinformatics application's performance on HPC systems will be also of high technological interest as those applications have novel computational patterns that can lead the next generation of heterogeneous computing systems.

## Acknowledgements

This work was partially supported by the computing facilities of Extremadura Research Centre for Advanced Technologies (CETA−CIEMAT), funded by the European Regional Development Fund (ERDF). CETA−CIEMAT belongs to CIEMAT and the Government of Spain. The authors also thankfully acknowledge the computer resources and the technical support provided by the Plataforma Andaluza de Bioinformática of the University of Málaga. We also thank NVIDIA for hardware donation under CUDA Teaching Center 2014

## References

1. J. L. Klepeis, K. Lindorff-Larsen, R. O. Dror, and D. E. Shaw, "Long-timescale molecular dynamics simulations of protein structure and function," *Current Opinion in Structural Biology*, vol. 19, no. 2, pp. 120–127, 2009.
2. G. M. Morris, D. S. Goodsell, R. Huey, and A. J. Olson, "Distributed automated docking of flexible ligands to proteins: Parallel applications of AutoDock 2.4," *Journal of Computer-Aided Molecular Design*, vol. 10, no. 4, pp. 293–304, Aug. 1996.
3. B. Fischer, H. Merlitz, and W. Wenzel, "Increasing Diversity in In-Silico Screening with Target Flexibility," in *Proceedings of Computational Life Sciences, First International Symposium, CompLife*, ser. CompLife 2005. Springer-Verlag, 2005, pp. 186–197.
4. I. Halperin, B. Ma, H. Wolfson, and R. Nussinov, "Principles of docking: An overview of search algorithms and a guide to scoring functions." *Proteins*, vol. 47, no. 4, pp. 409–443, 2002.

5. O. Korb, T. Stützle, and T. E. Exner, "Accelerating Molecular Docking Calculations Using Graphics Processing Units," *Journal of Chemical Information and Modeling*, vol. 51, pp. 865–876, 2011.
6. X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a Warehouse-Sized Computer," in *Proceedings of the 34th annual International Symposium on Computer Architecture*, ser. ISCA 07. ACM, 2007, pp. 13–23.
7. C. Hewitt, "ORGs for Scalable, Robust, Privacy-Friendly Client Cloud Computing," *IEEE Internet Computing*, vol. 12, no. 5, pp. 96–99, Sep. 2008.
8. A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis, "Energy-efficient cloud computing," *The Computer Journal*, vol. 53, no. 7, pp. 1045–1051, 2010.
9. J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, and R. A. Holt, "The Sequence of the Human Genome," *Science*, vol. 291, no. 5507, pp. 1304–1351, Feb. 2001. [Online]. Available: http://dx.doi.org/10.1126/science.1058040
10. M. Garland, S. Le Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang, and V. Volkov, "Parallel Computing Experiences with CUDA," *IEEE Micro*, vol. 28, pp. 13–27, July 2008.
11. M. Garland and D. B. Kirk, "Understanding throughput-oriented Architectures," *Communications of the ACM*, vol. 53, pp. 58–66, 2010.
12. I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Lecture Notes in Computer Science*, vol. 2150, 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.9069
13. A. Bateman and M. Wood, "Cloud computing," *Bioinformatics*, vol. 25, no. 12, p. 1475, Jun. 2009. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btp274
14. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010. [Online]. Available: http://dx.doi.org/10.1145/1721654.1721672
15. D. D'Agostino, A. Clematis, A. Quarati, D. Cesini, F. Chiappori, L. Milanesi, and I. Merelli, "Cloud Infrastructures for In Silico Drug Discovery: Economic and Practical Aspects," *BioMed Research International*, vol. 2013, pp. 1–19, 2013. [Online]. Available: http://dx.doi.org/10.1155/2013/138012
16. D. D?Agostino, A. Galizia, A. Clematis, M. Mangini, I. Porro, and A. Quarati, "A qos-aware broker for hybrid clouds," *Computing*, vol. 95, no. 1, pp. 89–109, 2013. [Online]. Available: http://dx.doi.org/10.1007/s00607-012-0254-4
17. (2014, Jan.) Apache hadoop project. [Online]. Available: http://hadoop.apache.org
18. B. Langmead, M. C. Schatz, J. Lin, M. Pop, and S. L. Salzberg, "Searching for SNPs with cloud computing." *Genome biology*, vol. 10, no. 11, pp. R134+, Nov. 2009. [Online]. Available: http://dx.doi.org/10.1186/gb-2009-10-11-r134
19. B. Langmead, C. Trapnell, M. Pop, and S. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biology*, vol. 10, no. 3, pp. R25–10, Mar. 2009. [Online]. Available: http://dx.doi.org/10.1186/gb-2009-10-3-r25
20. R. Li, Y. Li, X. Fang, H. Yang, J. Wang, K. Kristiansen, and J. Wang, "Snp detection for massively parallel whole-genome resequencing," *Genome research*, vol. 19, no. 6, pp. 1124–1132, 2009.

21. H. Yang, Q. Zhou, B. Li, Y. Wang, Z. Luan, D. Qian, and H. Li, "GPU Acceleration of Dock6's Amber Scoring Computation," *Advances in Computational Biology*, vol. 680, pp. 497–511, 2010.
22. S. Kannan and R. Ganji, "Porting Autodock to CUDA," *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pp. 1–8, 2010.
23. O. Korb, T. Stützle, and T. E. Exner, "Accelerating molecular docking calculations using graphics processing units." *Journal of chemical information and modeling*, vol. 51, no. 4, pp. 865–876, Apr. 2011.
24. O. Korb, T. Stützle, and T. Exner, "PLANTS: Application of Ant Colony Optimization to Structure-Based Drug Design," in *Ant Colony Optimization and Swarm Intelligence*, ser. Lecture Notes in Computer Science, M. Dorigo, L. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle, Eds. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2006, vol. 4150, ch. 22, pp. 247–258.
25. M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation, Politecnico di Milano, Italy, 1992.
26. M. Manfrin, M. Birattari, T. Stützle, and M. Dorigo, "Parallel ant colony optimization for the traveling salesman problem," in *Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS˜2006*, ser. LNCS, M. Dorigo, L. M. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle, Eds., vol. 4150. Berlin, Germany: Springer Verlag, 2006, pp. 224–234.
27. J. M. Cecilia, J. M. García, M. Ujaldón, A. Nisbet, and M. Amos, "Parallelization strategies for ant colony optimisation on gpus," in *NIDISC '2011: 14th International Workshop on Nature Inspired Distributed Computing. Proc. 25th International Parallel and Distributed Processing Symposium (IPDPS 2011)*, Anchorage (Alaska), USA, May 2011.
28. G. A. Trombetti, I. Merelli, A. Orro, and L. Milanesi, "Bgblast: A blast grid implementation with database self-updating and adaptive replication," *Studies in health technology and informatics*, vol. 126, p. 23, 2007.
29. J. J. Irwin and B. K. Shoichet, "ZINC - a free database of commercially available compounds for virtual screening." *J. Chem. Inf. Model.*, vol. 45, no. 1, pp. 177–182, Dec. 2004. [Online]. Available: http://dx.doi.org/10.1021/ci049714+
30. H. Alonso, A. A. Bliznyuk, and J. E. Gready, "Combining docking and molecular dynamic simulations in drug design." *Medicinal research reviews*, vol. 26, no. 5, pp. 531–568, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1002/med.20067
31. R. Huey, G. M. Morris, A. J. Olson, and D. S. Goodsell, "A semiempirical free energy force field with charge-based desolvation," *J. Comput. Chem.*, vol. 28, no. 6, pp. 1145–1152, Apr. 2007. [Online]. Available: http://dx.doi.org/10.1002/jcc.20634
32. (2014, Jan.) Amber on gpus. [Online]. Available: http://ambermd.org/gpus/benchmarks.htm
33. "The GROMACS website," http://www.gromacs.org/.
34. S. Pennycook, C. Hughes, M. Smelyanskiy, and S. Jarvis, "Exploring simd for molecular dynamics, using intel xeon processors and intel xeon phi coprocessors," 2013.