View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Journal of Molecular Biochemistry

Research Article

Molecular dynamics simulations through GPU video games technologies

Styliani Loukatou[1,†], Louis Papageorgiou[1,2,†], Paraskevas Fakourelis[1,3], Arianna Filntisi[1,4], Eleftheria Polychronidou[1,6], Ioannis Bassis[1],Vasileios Megalooikonomou[3], Wojciech Makałowski[5], Dimitrios Vlachakis[1,3,7] and Sophia Kossida[1]

[1]Computational Biology & Medicine Group, Biomedical Research Foundation, Academy of Athens, Soranou Efessiou 4, Athens 11527, Greece
[2]Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, University Campus, Athens 15784, Greece
[3]Department of Computer Engineering and Informatics Faculty, University of Patras, Patras 26500, Greece
[4]School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece
[5]Institute of Bioinformatics, University of Münster, Niels-Stensen-Straße 14, Münster 48149, Germany
[6]Department of Informatics, Ionian University, Tsirigoti Square 7, Corfu 49100, Greece
[7] Bionetwork ltd. 15234, Chalandri, Athens, Greece

Received on June 6, 2014; Accepted on June 30, 2014; Published on June 30, 2014

Correspondence should be addressed to Sophia Kossida; Tel: + 30 210 6597 199, Fax: +30 210 6597 545, E-mail: skossida@bioacademy.gr

Abstract

Bioinformatics is the scientific field that focuses on the application of computer technology to the management of biological information. Over the years, bioinformatics applications have been used to store, process and integrate biological and genetic information, using a wide range of methodologies. One of the most *de novo* techniques used to understand the physical movements of atoms and molecules is molecular dynamics (MD).

MD is an *in silico* method to simulate the physical motions of atoms and molecules under certain conditions. This has become a state strategic technique and now plays a key role in many areas of exact sciences, such as chemistry, biology, physics and medicine. Due to their complexity, MD calculations could require enormous amounts of computer memory and time and therefore their execution has been a big problem. Despite the huge computational cost, molecular dynamics have been implemented using traditional computers with a central memory unit (CPU).

A graphics processing unit (GPU) computing technology was first designed with the goal to improve video games, by rapidly creating and displaying images in a frame buffer such as screens. The hybrid GPU-CPU implementation, combined with parallel computing is a novel technology to perform a wide range of calculations. GPUs have been proposed and used to accelerate many scientific computations including MD simulations. Herein, we describe the new methodologies developed initially as video games and how they are now applied in MD simulations.

Introduction

The sharp increase in biological data, the need for its analysis and the increased interest of the scientific community to understand the structural and biological processes of biomolecules were the main reasons for the development of bioinformatics. The first milestone of this field was in 1970 when Paulien Hogewed studed information processes in biotic systems (Hesper & Hogeweg 1970). Later in 1980s, the term "bioinformatics" was used in genome analysis data to refer to *in silico* methods (Fago *et al.* 1992). Nowadays, bioinformatics is an interdisciplinary science incorporating computer science, biology, math and physics that develops novel methods applied in biological data. The biological knowledge that we can extract using *in silico* methods is significant and can be generated rapidly, as opposed to traditional biological methods (Balatsos *et al.* 2012, Palaiomylitou *et al.* 2008, Sellis *et al.* 2009, Vlachakis 2009).

In order to understand the cellular and biomolecular activities, the biological data must be combined to form a comprehensive picture of these activities (Attwood *et al.* 2011). Bioinformaticians have been developing computationally intensive techniques that can process biological data, including nucleotide sequences, amino acid sequences, 3D structures and biological signals and images. Major research efforts in the field include pattern recognition, sequence alignment, protein structure analysis (alignment – prediction), phylogenetic analysis, molecular dynamics, homology modelling, genome analysis, genome wide association studies, drug design and drug discovery. Furthermore, there are many predictive techniques spe-

© The Author(s) 2014. Published by Lorem Ipsum Press.

cific to gene expression and protein-protein interaction. Herein, we focus on MD techniques.

The first step in MD was the study of the macroscopic dynamic spheres by theoretical physicists in the late 1950s (Alder & Wainwright 1959). Later, given the theory that atoms and molecules are allowed to interact, MD simulations were applied to describe their physical movements in the context of N-body simulations, something that would be very difficult before the invention of computers (Mesirov *et al.* 1996). MD simulations can be extracted from two parameters. The first parameter, the potential energy function is estimated from the energy that grows between the bonded and non-bonded atoms in a molecule. The second parameter is the dynamics function that is computed from the forces between the atoms. Additionally, in order to achieve better accuracy, in many *in silico* methods, the kinetic energy of atoms in a specific temperature is also calculated.

MD simulation techniques are important methods for discovering the natural 3D structure and function of biological macromolecules. During the last 10 years a great effort has been used to link the structure and the function of biomolecules. Dynamic models provide significant information about the internal motion and conformational changes of a macromolecule, which play a major role in its function. Despite the inflexible structure models constructed to understand the physical shape of macromolecules in 3D space, MD simulations are applied to get more information about their functionality. Furthermore, MD simulations models provide the structural dynamic properties of molecules and the conformational transitions that they undergo (Karplus & McCammon 2002) such as molecular prediction and drug design (Sellis *et al.* 2012, Vangelatos *et al.* 2009, Vlachakis *et al.* 2012).

The number of studies that use MD simulations has increased during the last 10 years due to the growing availability of programs and computer power. However, the design of a dynamic model still requires a large amount of computer power. The duration of a simulation must be relevant to the timescale of the natural process being studied. A simulation of (n) particles has a complexity of $O(n^2)$, if all pairwise electrostatic and van der Waals interactions are accounted for explicitly. Therefore, the simulation of a macromolecule dynamic model in a traditional computer with a CPU requires a total time spanning from days to years. In order to reduce time cost, scientists have applied electrostatic methodologies. A novel idea has been applied in the last years, incorporating video games technology such as parallel programming and GPUS on molecular dynamic simulations (Vlachakis & Kossida 2013, Vlachakis *et al.* 2012).

Many available technologies were invented to serve purposes different to which they are currently used for. Despite the fact that GPU-CPU implementation and parallel computing were developed as video games technologies, it became clear that they are the means to perform a wide range of calculations and therefore have been applied in many scientific fields to solve computational problems. A GPU has more transistors than the average CPU. The application of parallel programming using GPUs in MD simulations has the significant benefit of a time cost significantly reduced by many times, as compared to CPUs. In the present review we describe the general information about the usage of GPU and parallel programming in the MD field.

The advent of the GPU

Assuming that x=GP; y=U; z=(x (√y))², then z equals what? By performing these arithmetic operations on x*y (GPU), the correct answer is z=GPGPU. In the early 80's, Intel created the precursor of graphics processing unit, the iSBX 275 Video Graphics Controller Multimodule Board, designed for industrial systems based on the Multibus standard. After almost twenty years, in 1999, GPU emerged, with NVIDIA marketing the GeForce 256 as "the world's first GPU". It is a circuit similar to CPU that is composed of a processor, a RAM and an I/O system, designed specifically for performing the complex mathematical and geometric calculations that are necessary for graphics performance (Kirk & Hwu 2010). The main target of this creation was the video game industry, which had a need for a rapid processor to produce images and graphics. Initially, computer graphics were in two dimensions (2D) and after several years, 3D computer games were developed. Some of the fastest GPUs have more transistors than the average CPU. Nowadays, every widely used technological innovation has a GPU installed (Alerstam *et al.* 2010, Hoang *et al.* 2013, Sharma *et al.* 2011). One picture is worth a thousand words; that is probably the phrase that Nvidia's people had in mind back in 2007, when they decided to develop CUDA, *i.e.* compute unified device architecture. CUDA is a framework for parallel computing and programming, which allows the developers to use GPUs not solely for graphics processing but for overall use, known as GPGPU, or general-purpose GPU. GPGPU was a simple idea, but the principal purpose of the construction of GPUs made this a difficult conception.

Molecular dynamics simulations: from CPU to GPU

The main apothegm of the comparison between GPU and CPU is that the architecture of the former focuses in executing a number of simultaneous threads, while the CPU executes only one thread in a heartbeat, which gives GPU a head start (Owens *et al.* 2008). As a result, it is feasible for a single GPU to replace several CPU cluster nodes and make parallel and quick calculations without waiting for shared resources. That allows for a better execution of complex algorithms and a more efficient management of big data. Further-

more, a graphics unit has less financial and electrical power requirements, so the creation of a supercomputer using GPUs instead of CPUs indemnifies the need for immense rooms filled with computers. The installation of a second GPU duplicates the parallelism of programming. Thus, it is now possible to fit a GPU supercomputer in a university lab (Stone *et al.* 2010).

Additionally, the GPU has impressive floating-point capabilities and high memory bandwidth. It gives the opportunity of optimized memory access, controlled execution configuration and direct management of the cores through a few lines of code. Towards this end, the direct connection between GPUs and motherboards leads to a more trustworthy performance (Ueng *et al.* 2008). The intelligent, innovative technology of GPUs was extended beyond the game industry so as to be utilized by the sciences. This gave computational chemistry and biology researchers the opportunity to expand the horizons of scientific exploration. Specifically, MD, quantum chemistry, visualization and docking applications run even 5 times faster on a GPU's CUDA cores in comparison to CPU's cores (Murakami *et al.* 2010).

MD simulations require a lot of complex arithmetic operations that sometimes lead to numerical errors, result bias, wrong reports and even non-polyonimic (NP) complete problems (Browna *et al.* 2012). Before the discovery of CUDA and GPGPU, GPUs have been used only for the imaging procedure of the molecular structures and the execution of MD simulations could take from hours to days to finish. The solution was provided by the GPGPU, as the GPUs have a lot of arithmetic units which can work in parallel, each doing an arithmetic operation while collaborating with each other. From the beginning of GPU coding to present, the programming rush continues to grow dramatically. The number of GPU based applications has almost reached 200, which is a 60 per cent increase in two years. The top ranked GPU applicable programs have been designed for MD, drug design, quantum chemistry, climate, weather and physics (NVIDIA Corporation, 2014).

In the field of MD, programmers and scientists have taken several steps towards this direction. Many programs based on GPGPU have been developed, supporting even multi-GPU simulations. This innovation unfolds many opportunities for the future, especially for smaller research units. It reduces the time required for the procedures and the necessary funds for research, promoting the development of new applications and scientific progress. Recently, a group from Universitat Pompeu Fabra in Barcelona developed GPUGRID, a project based on distributed computing systems, made of many GPUs all over the world, connected on a network for the implementation of biomedical research (Buch *et al.* 2010)**.** GPUGRID is a volunteer network, running on BOINC, an open source software platform developed at the University of California, Berkeley. GPUGRID executes MD simulations designed to run on NVIDIA's CUDA suitable GPUs.

CUDA and OpenCL share the main central idea but also have several differences. CUDA consists of a proprietary hardware architecture, an internal security assessor (ISA), a programming language, an application programming interface (API), a software development kit (SDK) and different tools. CUDA supports several operating systems, but a lot of its functionalities depend on NVidia. On the other hand, OpenCL is an OpenAPI and a language specification, which is supports multiple GPU companies such as Apple, NVidia, AMD and IBM. Most of the OpenCL capacities depend on the implemented vendor (Scarpino 2012).

Current trends in molecular dynamics

The growing amount of programs related to MD in combination with the available computing power has led to an increase in the number of MD studies. During the last 10 years the publications using MD have increased from six thousand to twenty six thousand (Figure 1). In total, the published studies that use MD have reached seventeen thousand. An extensive analysis of MD studies in a short review is therefore not achievable; however the significant current trends in this field will be reported.

MD simulations using GPU can now rapidly track processes. Many of those processes occur in less than a millisecond and can provide useful information concerning relevant biological systems in atomic resolution. Dynamic models have a significant impact on drug discovery, resolving many problems of the pharmaceutical industry. Molecular dynamics have been used in a numerous drug design studies, providing efficient methods and simulations (Papageorgiou *et al.* 2013, Vlachakis *et al.* 2014, Vlachakis & Kossida 2013, Vlachakis *et al.* 2013). Furthermore, MD simulations have increased the knowledge of molecular biology and have led to more successful clinical trials.

Dynamic models can provide significant molecular information concerning individual particle motions in time. In comparison to *in vitro* experiments, an *in silico* MD simulation is less expensive, more accurate and less time consuming. MD has many applications in a variety of studies. Recently, MD simulations were carried out to analyze the base of ligand selectivity of estrogen receptors (Hu & Wang 2014). Moreover, MD methods in combination with other methods are used to investigate the conformational behavior of peptides (Kulkarni & Ojha 2014). Additionally, the mechanical behavior of osteopontin-hydrixapatite interfaces is investigated using MD methods in order to understand the nanomechanics of bones and develop new artificial biological materials (Lai *et al.* 2014). Similarly, MD were used to simulate the cardiac troponin core domain complex, in order to understand the structural details of troponin switching (Jayasundar *et al.* 2014). Another significant aspect of such simula-

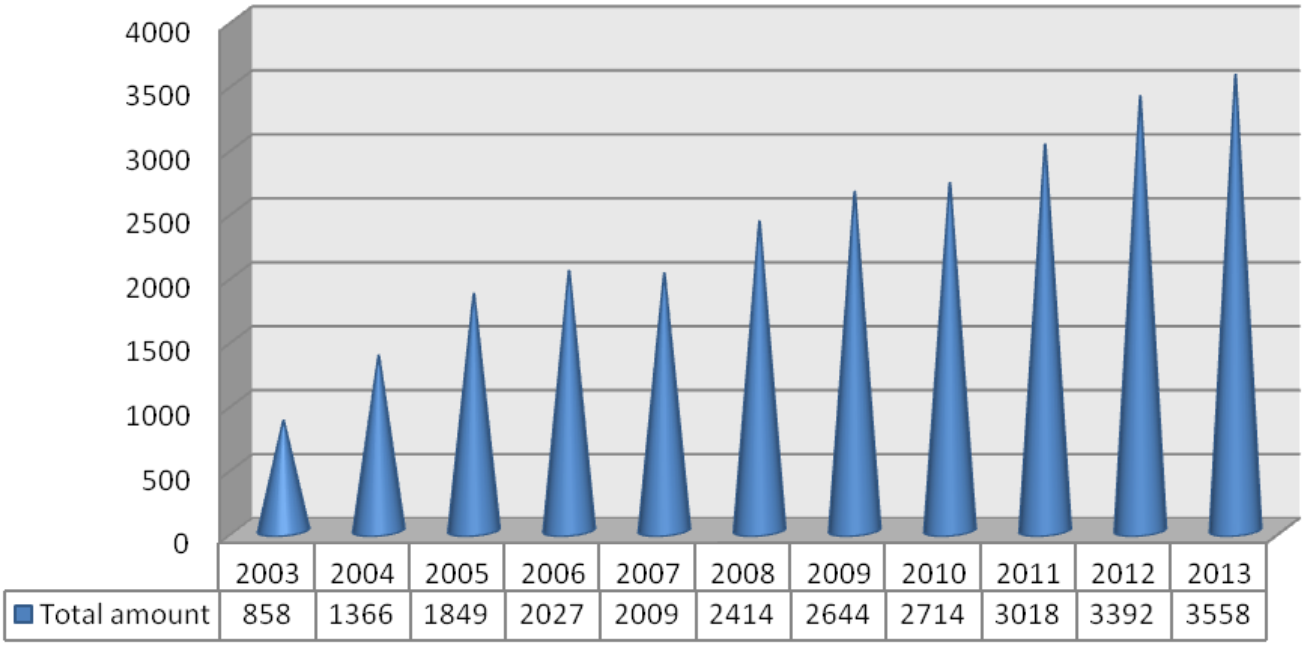

Figure 1. Pubmed publications related to molecular dynamics from 2003 to 2013.

tions is that they are completely controlled by the user so by altering, elaborating and removing specific parametes their properties can be examined.

The groundbreaking idea to share complicated, unsolved scientific problems through a video game with the public sounds like a science fiction scenario. However, this extremist idea has been recently applied in practice. In a recent example, computer gamers solved a problem in HIV research that had been puzzling scientist for years, in only three weeks of continuous gaming (Lv *et al.* 2013). This radiant idea could be applied to many other areas, such as molecular dynamics.

Benefits of parallel programming using GPU

Parallel programming is based on the assumption that problems involving large volumes of data can be broken down into smaller parts, which can be resolved by parallel redundant computations in each core (Almasi & Gottlieb 1989). Parallel computing can utilize multiple processors or computers that work together to accomplish a common task. Each processor (or computer) implements a separate part of the cleaved problem. This computational method has been the cornerstone for solving problems that could not be resolved with the use of a single CPU in real-time. Parallel computing, principally in the form of multi-core processors, is the main model of programming in modern computer architecture (Asanovíc *et al.* 2006).

Recently computer engineers explored the applicability of parallel programming using GPUs (Figure 2). Algorithms running on GPUs can often accelerate the calculation speed up to 100 times compared to common CPUs, enabling the development of several applications for neural networks, MD, physics simulations and countless other fields. However, it is essential to note that the benefits of GPU computing cannot be reached without the combinational work of the GPU with the CPU. Each unit is developed for different types of computing. CPUs have few cores that make it very effective in single-threaded serial processing, while GPUs have thousands of small cores developed for parallel computing. The reason behind such large increase in GPUs performance is their built, which is based on an escalating array of multithreaded Streaming Multiprocessors (SMs). Each one of GPU multiprocessors can carry out hundreds or even thousands of threads simultaneously. The architecture used to empower the administration of such a big number of threads is called single-instruction, multiple-thread (SIMT). This ability for parallel computing allows GPUs to fragment large and complex tasks into several small tasks, which run separately and contemporaneously on different threads of different multiprocessors (Papangelopoulos *et al.* 2014).

Available GPUs for parallel programming

Presently, there are several commercialized, CUDA-enabled GPUs adopted on desktops and notebooks, equally. The top ranked GPUs are mostly promoted by NVIDIA (Figure 3). For instance, GeForce 750, GTX 750, GTX 750 TI for desktop and GTX 850M, 840M, 830M for notebook users. These GPUs can attain the dominant computational capability, as is shown on NVidia's web page. GeForce GTX 860M has been released in two editions, the former with compute capability up to 5, and the latest up to 3, which depends on the number of CUDA cores. Addressed to the purchas-

ing public, NVidia is promoting Quadro and Tesla relevant products (NVIDIA Corporation, 2014). Particularly, Quadro GPUs specialise in professional visualization while Tesla in high-end imaging for technical and scientific computing. However, they do not achieve the highest score in the range of GPU computing. Indicatively, Tesla K40, K20 and Quadro K6000 gather the highest score (3.5) of these families. Other GPUs companies are Intel, Qualcomm, AMD, Altera and Samsung, which are able to implement GPGPU only by OpenCL (Stone *et al.* 2010).

OpenCL is the dominant open GPGPU programming language, even though GPGPU can be accelerated by modifying most commonly used programming languages, libraries and compilers. Some of these programming languages are C and C++ which use the NVCC compiler and, together with some addendums, are the base of the CUDA platform. Almost every high level programming language can be accelerated though C wrappers and swig. Most commonly used languages linked to CUDA are Fortran CUDA, PyCUDA, JCUDA, JCUBLAS and JCUFFT, Haskell, KAPPACUDA, Ruby, Python, Perl, GPULLIB and MATLAB, each one of which uses a specific compiler and certain libraries. (Rudy *et al.* 2011). Vector is a new GPGPU programming language developed by Howard Mao and his team, who also developed a compiler that compiles Vector into CUDA (Resch 2010). Another coding language is CHESTNUT, which is a GPU programming language for non-experts as its developers characterize it (Stromme *et al.* 2012). The CUDA compute capability plays a significant role in the selection of the programming language. Namelya compute capability of 1.x can be used with C and some simple extensions while compute capability of 2.x can be used with C++. Finally, there exists Harlan; a new GPGPU programming language introduced by Eric Holk and his laboratory team (Holk *et al.* 2011).

Molecular dynamics simulators on CPU and GPU

As a result of this ever-growing trend, many programs which can implement several MD methods were developed, either using both CPU and GPU or using only GPU. Examples of this software are listed in Table 1, accompanied by their principal characteristics. As is apparent from the fifth column of the table, each program that is able to run a simulation on both GPU and CPU is superior in speed. Thus, when the simulation is running on a GPU, it requires less time to complete the task. Finally, the immediacy of each application depends on the habits and the skills of each user. If the user is not familiar with the command prompt, the best solution is to choose a program that provides a GUI interface. The program selection also depends on the operating system in which the user wants to install the program and the reasons for installing it.

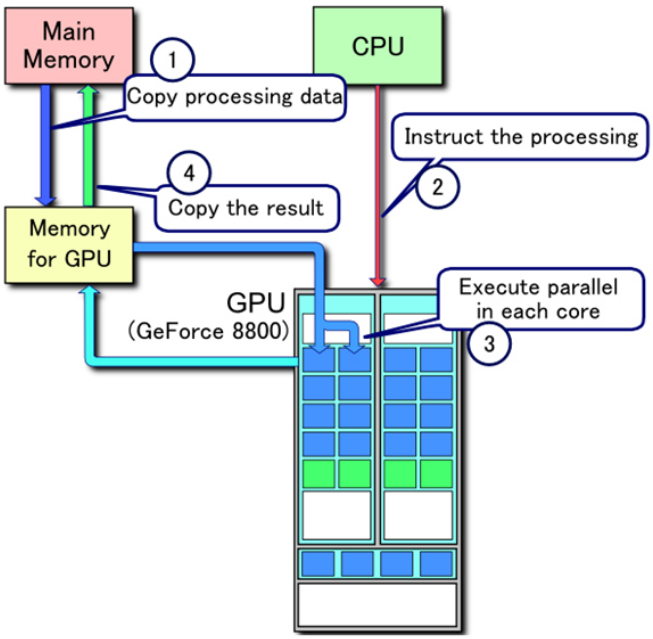

Figure 2. Diagram of Cuda parallel programming using GPU (Johnson and Vijayalakshmi 2013).

On the other hand, there are plenty of disadvantages in using CUDA. It can be accelerated only through NVIDIA's GPUs. Thus ATI's, AMD's and other users are barred from this technology. The time bottleneck which can come up at the communication between GPU and CPU still exists. Perhaps a huge progress has taken place in the area of research and development, but the number of GPU users is stable. CUDA is still under development and it is not known whether it will consolidate the marketplace or more problems will arise and CUDA will hit rock bottom (Che *et al.* 2008, Pan *et al.* 2008).

Conclusion

Video games were the starting point of GPU technology. GPUs caused a temporary retreat in the course of CPU-based computer technology. The many benefits of parallel programming and GPUs gained the interest of many scientists. GPGPU is a very promising computing technology, which holds plenty to offer in every field of science, especially in bioinformatics. Many applications running on GPUs have already been constructed and a plethora of researchers is involved in this programming method. Nowadays, there is a wide range of techniques performed with parallel programming and GPUs. The combination of GPU video game technology with molecular dynamics simulations revolutionalized computational biology and drug discovery. Additionally, many computational issues have been solved such as the issue of time and computing power. In the near future a superabundance of achievements through GPU molecular dynamics is expected to occur. One of them could be the understanding of the relations between 3D structure, dynamic model and

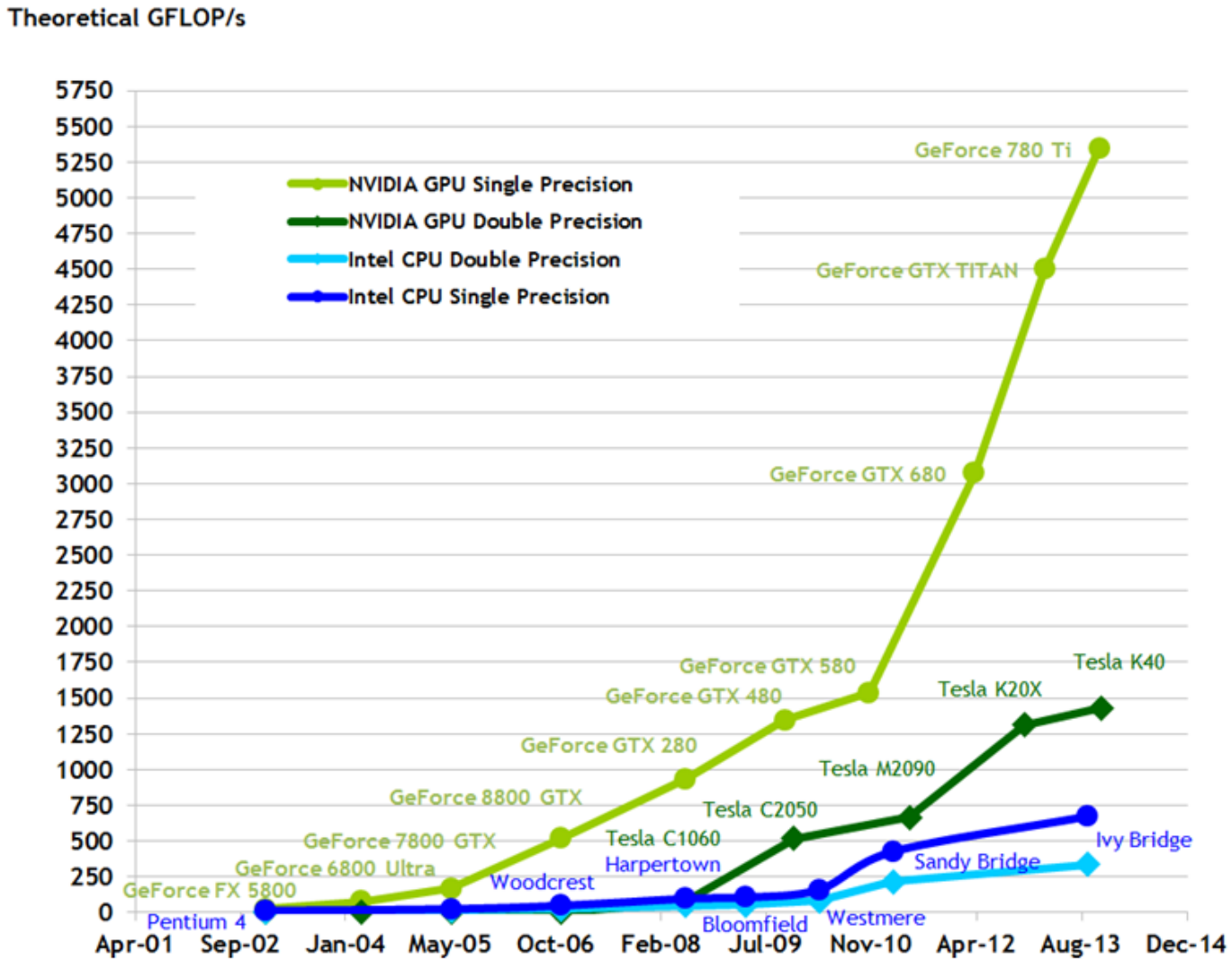

Figure 3. Memory Bandwidth for the CPU and GPU (NVIDIA, 2014).

functionality of a molecule. Drug discovery and design, genomics and computational chemistry are some of the areas most expected to adopt this programming architecture.

Funding

This work was partially supported by: (1) The BIOEXPLORE research project. BIOEXPLORE research project falls under the Operational Program "Educationand Lifelong Learning" and is co-financed by the European Social Fund (ESF) and National Resources. (2) European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Educationand Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thales. Investing in knowledge societythrough the European Social Fund.

References

Alder BJ & Wainwright TE 1959 Studies in molecular dynamics. I. General Method *Journal of Chemical Physics* **31** 459-466

Alerstam E, Lo WC, Han TD, Rose J, Andersson-Engels S & Lilge L 2010 Next-generation acceleration and code optimization for light transport in turbid media using Gpus *Biomed Opt Express* **1** 658-675

Almasi GS & Gottlieb A 1989 *Highly parallel computing* The Benjamin/Cummings series in computer science and engineering. CA, USA: Benjamin/Cummings

Asanovíc K, Bodik R, Catanzaro B, Gebis J, Husbands P, Keutzer K, Patterson D, Plishker W, Shalf J, Williams S & Yelick K 2006 The landscape of parallel computing research: a view from Berkeley *Electrical Engineering and Computer Sciences, University of California at Berkeley*

Attwood TK, Gisel A, Eriksson NE & Bongcam-Rudloff E 2011 *Concepts, historical milestones and the central place of bioinformatics* in Modern Biology: A European Perspective. Bioinformatics – Trends and Methodologies. Eds M Mahdavi. InTech

Browna WM, Kohlmeyerb A, Plimptonc SJ & Tharringtona AN 2012 Implementing molecular dynamics on hybrid high performance computers - particle–particle particle-mesh. *Comp Phys Comm* **183** 449-459

Table 1. Main softwares using GPU for MD.

Softwares for Molecular Dynamics							
Name	CPU	GPU	GUI	Speed (GPU vs CPU)	3D Graphics	Service	OS
Abalone	YES	YES	NO	4-29x	YES	Executes bio molecular simulations of proteins, DNA, ligands	Windows
ACEMD	NO	YES (multi-GPU support)	YES(Charmm -GUI)	160 ns/day	NO	Program for MD simulations running on NVidias GPUs	Windows, Linux
AMBER	YES	YES (multi-GPU support)	YES	100 ns/day	NO	Executes MD simulator on biomolecules	MacOs, Linux, Windows (through Cygwin)
Ascalaph Designer	YES	YES	YES	4-29x	YES	General purpose MD simulator	Linux, Windows
CHARMM	YES	YES (multi-GPUsupport)	YES	32-35x	NO	Program for MDs, commercial	MacOs, Linux, Windows (through Cygwin)
Discovery Studio	YES	YES	YES	x5	YES	Simulation program with many capabilities	Windows, Linux, MacOS
DL POLY	YES	YES (multi-GPU support)	YES	4x	YES	Application to facilitate MD simulations of macro-molecules, polymers, ionic systems	Linux, Windows (through cyngwin), MacOS
GROMACS	YES	YES (multi-GPUsupport)	YES (GROMITA)	165 ns/day	NO	A hybrid CPU-GPU MD program, which perform high-level simulations	Linux, Windows (through cyngwin), MacOS
GULP	YES	NO	NO	------	NO	An application which optimizes MD	Linux, Windows, MacOS
HOOMD-blue	YES	YES (multi-GPUsupport)	NO	2x	NO	Molecular dynamic simulation toolkit	Linux, MacOS
LAMMPS	YES	YES (multi-GPUsupport)	YES (Lammpsfe)	3-18x	NO	MD simulation package	Linux, Windows (through cyngwin), MacOS
MD-display	YES	NO	YES	------	YES	A platform for 3D MD	Windows, Linux
MdynaMix	YES	NO	YES	------	NO	Program for general use of MD's simulations	Linux
MOE	YES	NO	YES	------	YES	Molecular Operating Environment	Windows, Linux, MacOS
ORAC	YES	NO	NO	------	NO	MD's simulations	Linux
Namd	YES	YES (multi-GPU support)	NO (YES through VMD)	4 ns/days	NO (YES through VMD)	Parallel molecular dynamics pro-gram for Biomolecular systems	Windows, Linux, MacOS, Solaris
NWChem	YES	YES	NO	3-10x	NO	Computational chemistry and MD simulations project	Linux, Windows, MacOS, FreeBSD
Packmol	YES	NO	YES	------	YES	An application to generate MD geometries	Linux, Windows, MacOS
Protein LocalOptimization Program	YES	NO	NO	------	NO	Program for protein modeling and simulations	Linux
Protomol	NO	YES	YES	-----	YES	Object-oriented programming for MD	Linux, Windows, Solaris
RedMD	YES	NO	NO	------	YES (using aninterface)	Package for MD simulations	Linux
TeraChem	YES	YES	YES	44-650x100	NO	MD program for very large molecules' simulations	Linux
VEGA ZZ	YES	YES	YES	10x	YES	2D and 3D viewer and editor	Linux (without GUI interface), Windows
VMD	YES	YES (multi-GPU support)	YES	100-125x	YES	A program for modeling, analyzing, visualizing biomolecu-les, it is intergrated with NAMD	Windows, Linux, MacOS, Solaris
WHAT IF	YES	NO	YES	------	YES	A visualizer for MD using an interface	Linux, Windows (under conditions)

Buch I, Harvey MJ, Giorgino T, Anderson DP & De Fabritiis G 2010 High-throughput all-atom molecular dynamics simulations using distributed computing. *J Chem Inf Model* **50** 397-403

Che S, Boyer M, Meng J, Tarjan D, Sheaffer JE & Skadron K 2008 Performance study of general-purpose applications on graphics processors using Cuda *J Parall Distr Comp* **68** 1370-1380

Fago A, D'Avino R & Di Prisco G 1992 The hemoglobins of notothenia angustata, a temperate fish belonging to a family largely endemic to the antarctic ocean *Eur J Biochem* **210** 963-970

Hesper B & Hogeweg P 1970 Bioinformatica: Een Werkconcept Kameleon *Kameleon* **1** 28-29

Hoang RV, Tanna D, Jayet Bray LC, Dascalu SM & Harris FC Jr 2013 A novel Cpu/Gpu simulation environment for large-scale biologically realistic neural modeling *Front Neuroinform* **7** 19

Holk E, Byrd W, Mahajan N, Willcock J, Chauhan A & Lumsdaine A 2011 declarative parallel programming for Gpus. In *14th biennial ParCo* edited by KD Bosschere, EH D'hollander, GR Joubert, D Padua, F Peters and M Sawyer. Amsterdam, Netherlands.

Hu G & Wang J 2014 Ligand selectivity of estrogen receptors by a molecular dynamics study *Eur J Med Chem* **74** 726-735

Jayasundar JJ, Xing J, Robinson JM, Cheung HC & Dong WJ 2014 Molecular dynamics simulations of the cardiac troponin complex performed with fret distances as restraints. *PLoS One* **9** e87135

Johnson J & Vijayalakshmi G 2013 Comparison study of parallel computing with Aluand Gpu (Cuda). *Int J Sci Res (IJSR)* **2** 209-212

Karplus M & McCammon JA 2002 Molecular dynamics simulations of biomolecules *Nat Struct Biol* **9** 646-652

Kirk D & Hwu WM 2010 *Programming Massively Parallel Processors Hands-on with Cuda*. MA, USA: Morgan Kaufmann Publishers

Kulkarni AK & Ojha RP 2014 Combined 1 H-NMR and molecular dynamics studies on conformational behavior of a model heptapeptide, GRGDSPC. *Chem Biol Drug Des*

Lai ZB, Wang M, Yan C & Oloyede A 2014 Molecular dynamics simulation of mechanical behavior of osteopontin-hydroxyapatite interfaces. *J Mech Behav Biomed Mater* **36C** 12-20

Lv Z, Tek A, Da Silva F, Empereur-mot C, Chavent M & Baaden M 2013 Game on, science - how video game technology may help biologists tackle visualization challenges. *PLoS One* **8** e57990

Mesirov JP, Schulten K & Sumners DW 1996 *Mathematical Approaches to Biomolecular Structure and Dynamics*. The Ima Volumes in Mathematics and Its Applications. New York, USA: Springer

Murakami T, Kasahara R & Saito T 2010 An implementation and its evaluation of password cracking tool parallelized on Gpgpu. *IEEE* 534 - 538

NVIDIA 2014 Cuda C Programming Guide. *NVIDIA Corporation*

NVIDIA Corporation. Cuda Gpus. NVIDIA Developer. Accessed from https://developer.nvidia.com/cuda-gpus at June 6, 2014.

Owens JD, Houston M, Luebke D, Green S, Stone JE & Phillips JC 2008 Gpu computing. *Proc IEEE* **6**

Pan L, Gu L & Xu J 2008 Implementation of medical image segmentation in Cuda. *IEEE* 82-85

Papageorgiou L, Vlachakis D, Koumandou VL, Papangelopoulos N & Kossida S 2013 Computer-aided drug design and biological evaluation of novel anti-Greek goat encephalitis agents. *Int J Syst Biol Biomed Tech* **2** 1-16

Papangelopoulos N, Vlachakis D, Filntisi A, Fakourelis P, Papageorgiou L, Megalooikonomou V & Kossida S 2014 State of the art Gpgpu applications in bioinformatics. *Int J Syst Biol Biomed Tech* **2** 24-48

Resch M 2010 *High Performance Computing on Vector Systems 2009* New York, USA: Springer

Rudy G, Khan MM, Hall M, Chen C & Chame J 2011. A programming language interface to describe transformations and code generation. *Lang Comp Parall Comp* **6548** 136-150

Scarpino M 2012 *Opencl in Action : How to Accelerate Graphics and Computation*. NY, USA: Manning

Sharma R, Gupta N, Narang V & Mittal A 2011 Parallel implementation of DNA sequences matching algorithms using pwm on Gpu architecture. *Int J Bioinform Res Appl* **7** 202-215

Stone JE, Gohara D & Shi G 2010 Opencl: a parallel programming standard for heterogeneous computing systems. *Comput Sci Eng* **12** 66-72

Stone JE, Hardy DJ, Ufimtsev IS & Schulten K 2010 Gpu-accelerated molecular modeling coming of age. *J Mol Graph Model* **29** 116-125

Stromme A, Carlson R & Newhall T 2012 Chestnut: A Gpu programming language for non-experts. *PMAM*

Ueng S, Lathara M, Baghsorkhi SS & Hwu WW 2008 Cuda-Lite: reducing Gpu programming complexity. *Lang Comp Parall Comp* **5335** 1-15

Vlachakis D, Bencurova E, Papangelopoulos N & Kossida S 2014 Current state-of-the-art molecular dynamics methods and applications. *Adv Protein Chem Struct Biol* **94** 269-313

Vlachakis D & Kossida S 2013 Molecular modeling and pharmacophore elucidation study of the classical swine fever virus helicase as a promising pharmacological target. *PeerJ* **1** e85

Vlachakis D, Tsagrasoulis D, Megalooikonomou V & Kossida S 2013 Introducing drugster: a comprehensive and fully integrated drug design, lead and structure optimization toolkit. *Bioinformatics* **29** 126-128