Supercomputing Frontiers and Innovations

2018, Vol. 5, No. 4

Scope

- Enabling technologies for high performance computing
- Future generation supercomputer architectures
- Extreme-scale concepts beyond conventional practices including exascale
- Parallel programming models, interfaces, languages, libraries, and tools
- Supercomputer applications and algorithms
- Distributed operating systems, kernels, supervisors, and virtualization for highly scalable computing
- Scalable runtime systems software
- Methods and means of supercomputer system management, administration, and monitoring
- Mass storage systems, protocols, and allocation
- Energy and power minimization for very large deployed computers
- Resilience, reliability, and fault tolerance for future generation highly parallel computing systems
- Parallel performance and correctness debugging
- Scientific visualization for massive data and computing both external and in situ
- Education in high performance computing and computational science

Editorial Board

Contents

DOI: 10.14529/jsfi180401

Adaptive Load Balancing in the Modified Mind Evolutionary Computation Algorithm*

Maxim K. Sakharov[1], *Anatoly P. Karpenko*[1]

The paper presents an adaptive load balancing method for the modified parallel Mind Evolutionary Computation (*MEC*) algorithm. The proposed method takes into account an objective function's topology utilizing the information obtained during the landscape analysis stage as well as the information on available computational resources. The modified *MEC* algorithm and proposed static load balancing method are designed for loosely coupled parallel computing systems and imply a minimal number of interactions between computational nodes when solving global optimization problems. A description of the proposed method is presented in this work along with the results of computational experiments, which were carried out with a use of multi–dimensional benchmark functions of various classes. Obtained results demonstrate that an effective use of available computational resources in the proposed method helps finding a better solution comparing to the traditional parallel *MEC* algorithm balancing. Further development of the proposed method requires more advanced termination criteria in order to avoid excessive iterations.

Keywords: load balancing, landscape analysis, mind evolutionary computation, global optimization.

Introduction

One of the distinct features of the real–world global optimization problems is the computational complexity of the objective functions. To cope with such problems within reasonable time, it is necessary to utilize parallel computing systems. Grid systems, made of personal computers and workstations, are widely used by scientific communities [8, 20]. Their availability is caused by a relatively low cost and easy scaling. Such systems belong to a class of loosely coupled computing systems.

When solving an optimization problem on parallel computing systems in general, and on loosely coupled systems in particular, one of the main difficulties is the optimal mapping problem [4, 20] is how to distribute groups of sub–problems over processors. It should be noted that a problem of optimal mapping of computational processes onto a parallel computing system is one of the main issues associated with parallel computations. It is well known that such a problem is NP complete and can be solved with exact methods within a very narrow class of problems [4, 5].

Most often, methods of load balancing are used to obtain an approximate solution to an optimal mapping problem. The main idea behind these methods is to distribute the computation tasks over the processors in such a way that the total computing and communication load is approximately the same for each processor.

There are two types of load balancing: static and dynamic. Static balancing is performed once, before the computational process starts. When computational complexity of the objective functions is unknown in advance, static load balancing can't be effective. In this case, dynamic load balancing should be implemented during calculation: the algorithm must re–distribute computation tasks between processors depending on their loading.

*The paper is recommended for publication by the Program Committee of the International Scientific Conference "Parallel computational technologies (PCT) 2018".

[1]Bauman MSTU, Moscow, Russian Federation

Regarding loosely coupled computational systems, it is required to minimize the interaction between computational nodes, hence re–distributing tasks during the calculations is highly inefficient. In this paper, we propose to use the landscape analysis (*LA*) [1, 12, 19] in order to evaluate computational complexity of objective functions and thus improve the efficiency of static load balancing.

Nowadays, the *LA* methods are widely used in global optimization problems to determine some distinct features of objective functions and, subsequently, classify these functions, for example, to adjust some free parameters of the utilized population algorithms [10]. Population algorithms represent a universal and powerful tool for solving global optimization problems. Their popularity is caused by the fact that they can be easily implemented and applied to various fields as they are based on the universal idea of evolution. However, their efficiency heavily depends on numerical values of their free parameters, which should be selected based on the characteristics of a problem in hand.

Within the landscape analysis approach, it is proposed to extract information on the objective function's landscape and topology at the cost of additional trials (approximately 1–10% of the total number of evaluations) [13]. In other words, the *LA* methods help identifying the search for sub–domains with either rugged or smooth topology, or for sub–domains where the objective function's values are almost identical, etc.

A class of Mind Evolutionary Computation (*MEC*) algorithms [2, 6, 7] is considered in this work. These algorithms belong to a family of population methods inspired by human society. Individual s is considered as an intelligent agent which operates in group S made of analogous individuals. During the evolution process, each individual is affected by other individuals within the group. This simulates the following logics. In order to achieve a high position within a group, an individual has to learn from the most successful individuals in this group. Groups themselves should follow the same principle to stay alive in the intergroup competition.

This work deals with the Simple *MEC* (*SMEC*) algorithm. It belongs to a class of *MEC* algorithms and was selected for the research because it is highly suitable for parallel computations, especially for loosely coupled systems. Generally, in order to be efficient on loosely coupled systems, the basic optimization algorithm must imply a minimum number of interactions between sub–populations which evolve on separate computational nodes. Only a few currently known population–based algorithms, including the *MEC* algorithm, meet this requirement.

The authors propose a modified parallel *MEC* algorithm with an incorporated *LA* procedure accompanied with the static load balancing method. Pperformance research was carried out to evaluate efficiency of the proposed approach in comparison with the traditional parallel *SMEC* algorithm [15, 17].

1. Problem Statement and the Basic Optimization Algorithm

A global deterministic unconstrained minimization problem is considered in this work

$$\min_{X \in R^{|X|}} \Phi(X) = \Phi(X^*) = \Phi^*, \tag{1}$$

where $\Phi(X)$ is the scalar objective function; $\Phi(X^*) = \Phi^*$ is the required minimal value; $X^* = (x_1, x_2, , x_{|X|})$ is $|X|$–dimensional vector of variables; $R^{|X|}$ is $|X|$–dimensional arithmetical space. Domain D_0 is defined as follows:

$$D_0 = \left\{ X | x^{min} \leq x_i \leq x^{max}, i \in [1 : |X|] \right\} \tag{2}$$

and used for generating the initial population of solutions. The *MEC* algorithm can be considered as a multi–population one. A multi–population consists of independent sub–populations with different instances of the *SMEC* algorithm. Each sub–population is made of leading groups $S^b = (S_1^b, S_2^b, , S_{|S^b|}^b)$ and lagging groups $S^w = (S_1^w, S_2^w, , S_{|S^w|}^w)$. The number of individuals within each group is set to be the same and equals $|S|$. Every group S_i^b or S_j^w has its own communication environment called a local blackboard and denoted C_i^b or C_j^w correspondingly. In addition, the whole sub–population $S = \{S^b, S^w\}$ has a general global blackboard C^g. The original *MEC* algorithm was presented in [2] and was named the simple Mind Evolutionary Computation algorithm (Simple *MEC*, *SMEC*). The *SMEC* algorithm is based on the operations of group initialization, similar-taxis and dissimilation. Similar-taxis and dissimilation are iteratively repeated until there is an increase in the maximum score of the leading groups. When the growth of this indicator stops, the current solution of a problem is declared a global minimum. By individuals score we mean the value of objective function $\Phi(X)$ in its current position [8].

In [16], the authors carried out a wide performance research of the *SMEC* algorithms efficiency depending on the values of the following free parameters: standard deviation σ, used for the generation of new individuals; removing frequency of lagging groups ω; ratio η between the number of leading $|S^b|$ and lagging $|S^w|$ groups in a sub–population. Obtained results were used to formulate the strategies for selecting optimal numerical values for those parameters.

2. Modified SMEC Algorithm

The modified Mind Evolutionary Computation algorithm is based on hybridization of the multi–population *SMEC* algorithm with the incorporated landscape analysis stage. The proposed *LA* method allows one to study the objective functions topology without any additional information on the problem in hand known a priori. Based on the results obtained during the landscape analysis, we can classify an objective function into one of six possible categories and form an adaptation strategy for the algorithm [14, 16]. To achieve this, the initialization stage of the *SMEC* algorithm was modified. New initialization stage with the incorporated *LA* procedure can be described as follows.

1. Generate N quasi–random $|X|$–dimensional vectors within domain D_0. Here, N is the total number of all groups in a multi–population (free parameter of the algorithm).
2. For every X_r, $r \in [1\colon N]$, calculate the corresponding values of objective function Φ_r and sort those vectors in ascending order of values Φ_r, $r \in [r\colon N]$.
3. Equally divide a set of vectors $(X_1, X_2, \ldots, X_N)$ into $|K|$ sub–populations (another free parameter).
4. For every sub–population K_l, $l \in [1\colon |K|]$, calculate a value of its diameter d_l, the maximum Euclidean distance between any two individuals.
5. Build a linear approximation for the dependency of diameter $d(l)$ on group number l, using the least squares method [14].
6. Calculate an estimation of the size of domain D_0 using the formula

$$d_0 = \sqrt{|X|(x^{max} - x^{min})^2}.$$

7. Classify objective function $\Phi(X)$ into one of the six categories provided in Tab. 1 based on the calculated parameters and determine the corresponding numerical values for the

SMEC algorithm for each sub–population. Numerical parameters in Tab. 1 are based on the empirical results [16].

Table 1. Classification of objective functions based on the *LA* results

	$d(l)$ increases	$d(l)$ constant	$d(l)$ decreases
$\frac{d_0}{d_1} > 2.5$	Nested sub–populations with dense first sub–population (category I)	Non–intersected sub–populations of the same size (category III)	Distributed sub–populations with potential minima (category V)
$\frac{d_0}{d_1} \leq 2.5$	Nested sub–populations with sparse first sub–populations (category II)	Intersected sub–populations of the same size (category IV)	Highly distributed sub–populations with potential minima (category VI)

In Fig. 1, an example of calculating a diameter is demonstrated for the first sub–population of individuals generated for two–dimensional Composition Function 1 from CEC 14 [11]. Circles represent an arbitrary neighborhood of the individuals; different colors of those circles correspond to different sub–populations. Additional examples of the landscape analysis procedure for various Composition Functions from the same set of benchmark problems are presented in Fig. 2.

Each classification category describes a specific topology of an objective function with the following pre–determined rules for calculating numerical values of the free parameters.

1. For objective functions from categories I and II, there is a high probability to find a global optimum within sub–population K_1. In such a case, numerical values for parameter σ are defined to increase the search intensification for the first sub–populations and the search diversification for the last sub–populations:

$$\sigma(l) = 0.25 + 0.75\frac{(l-1)}{(|K|-1)}.$$

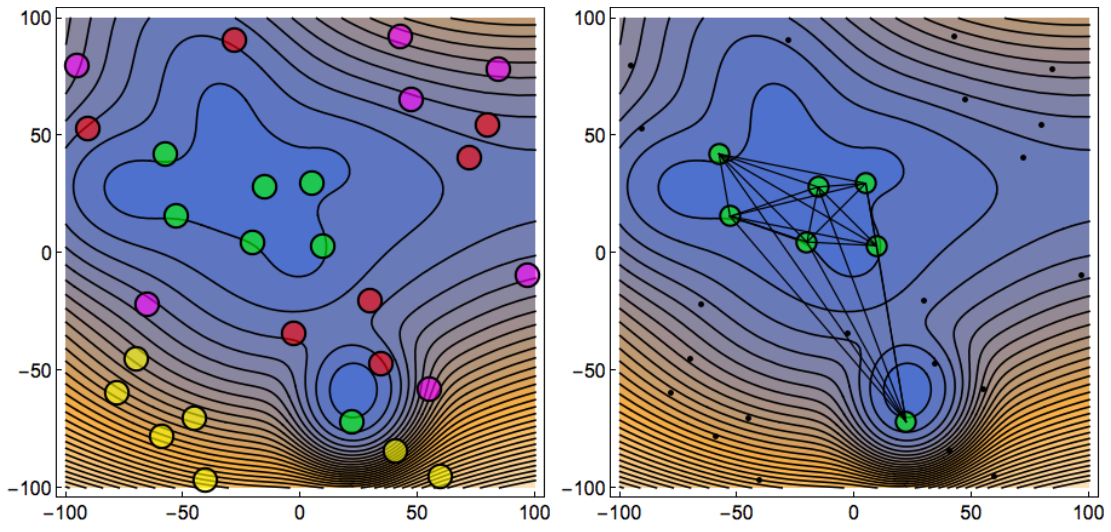
2. For objective functions from categories III and IV, numerical values for parameter σ are defined randomly from a specified range:

$$\sigma(l) = rand(0.1, 0.9).$$

3. For objective functions from categories V and VI, the first sub–population usually covers a large part of initial domain D_0. In such a case, the goal is to increase the search diversification for the first sub–populations:

$$\sigma(l) = 1 - 0.75\frac{(l-1)}{(|K|-1)}.$$

4. For categories with odd numbers, the removing frequency of lagging groups $\omega = 20$, while ratio between the number of leading and lagging groups in a sub–population $\eta = 50$. For categories with even numbers: $\eta = 75$, and $\omega = 25$ in order to provide enough time for lagging groups in a sub–population to explore their search sub–domains.



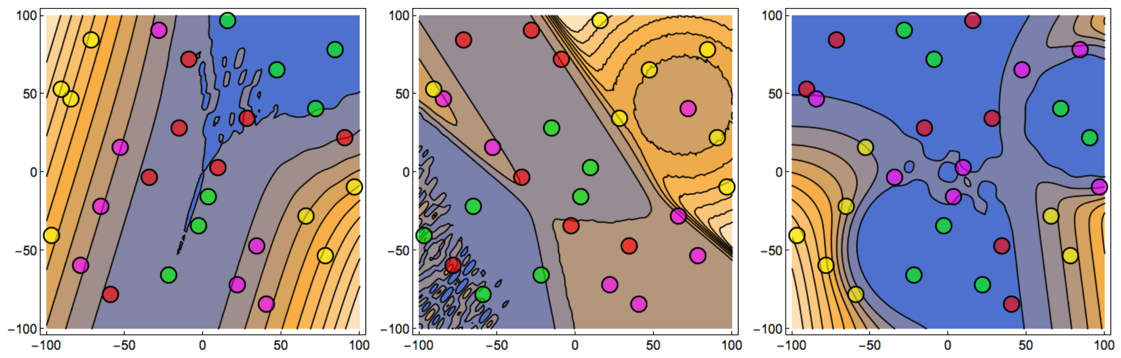
(a) Distribution of individuals for four sub–populations (b) Determining a diameter of the first sub–population

Figure 1. Determining a diameter of the first sub–population for the benchmark Composition Function 1 from CEC 14

3. Adaptive Load Balancing

As mentioned above, parallel optimization methods should be used in order to efficiently solve the real–world global optimization problems. Multi–population version of the *SMEC* algorithm can be naturally used to decompose the problem and map it onto computational nodes of the loosely coupled system. In such a case, each sub–population or a group of sub–populations evolves independently on separate computational nodes. This approach lies behind the idea of the traditional Parallel *MEC* algorithm and was used by the authors in [9] in order to design multi–memetic algorithms.

When performing landscape analysis, it is reasonable to use all extracted information in order to use available computational resources as efficiently as possible and, subsequently, increase the efficiency of the algorithm in general.



(a) Composition function 3 from CEC 14 (Group I) (b) Composition function 4 from CEC 14 (Group II) (c) Composition function 5 from CEC 14 (Group VI)

Figure 2. Results of the landscape analysis procedure for a few benchmark functions

We modified the initialization stage described above so that during step 2, the time required for calculations t_r is measured together with calculating the values of objective function Φ_r. The proposed adaptive load balancing method can be described as follows.

1. For each sub–population K_l, $l[1 : |K|]$, we analyzed all time measurements t_r for the corresponding vectors X_r, $r \in [1 : \frac{N}{|K|}]$ in order to determine whether there are outliers or not. All found outliers t^* are excluded from sub–populations. A new sub–population is composed from those outliers; it can be studied upon users request after the computational process is over.
2. If the number of sub–populations $|K|$ is equal to the number of available computational nodes M, then we send the first sub–population K_1 to one node. Individuals in other sub–populations are re–distributed between neighboring sub–populations starting from K_2, so that the average calculation time would be approximately the same for every sub–populations. Balanced sub–populations K_l, $l \in [2 : |K|]$ are then mapped onto the computational nodes. Then we go to step 5.
3. If the number of sub–populations $|K|$ is greater than the number of available computational nodes M, then we send the first sub–population K_1 to one node. The rest sub–populations are divided into equal groups of sizes $\frac{(|K|-1)}{(M-1)}$ in ascending order; the last group contains all remaining sub–populations.
4. Individuals in sub–populations are re–distributed between neighboring sub–populations starting from K_2, so that the average calculation time would be approximately the same for every sub–populations. Balanced sub–populations K_l, $l \in [2 : |K|]$ are then mapped onto the computational nodes.
5. The modified *SMEC* algorithm is launched on each node with the specific values of free parameters in accordance with Tab. 1.

4. Performance Investigation

The modified *SMEC* algorithm and the proposed load balancing method were implemented in Wolfram Mathematica. A set of numerical experiments with the use of multi–dimensional ($|X| = 16$) benchmark functions of various classes [3] was carried out to estimate performance of the proposed approach. During the study, efficiency of the parallel modified *SMEC* algorithm with the adaptive load balancing (*SMEC/LA*) and the parallel multi–population *SMEC* algorithm without the landscape analysis were compared.

The benchmark optimization functions considered in this paper are presented in Tab. 2 along with their known global optimal values. An original domain for generating initial population equals

$$D_0 = \{X : -5 \leq x_i \leq 5, i \in [1 : |X|]\}.$$

All numerical experiments were carried out using the multi–start method with 100 launches. The best obtained value of objective function Φ^* as well as its average value $\bar{\Phi}$ based on the results of all launches were utilized as the performance indices for comparison of the two algorithms and their software implementations along with maximum iteration number N_{max} among all computational nodes and the number or iterations N_{opt} from the node, where the optimal value was obtained.

In this work, LP_τ–sequence was used to generate the initial vectors for the LA procedure, as it provides a uniform coverage of the search domain [18]. Other parameters had the following values for both algorithms: the total number of groups in one sub–population $\gamma = 40$; the number of individuals in each group $|S| = 40$; the number of sub–populations $|K| = 8$.

The number of stagnation iterations $\lambda_{stop} = 50$ was used as a termination criterion for the algorithms. Tolerance used for identifying stagnation was equal to $\epsilon = 10^{-5}$. All computations were performed with a use of computational network made of $M = 8$ computational nodes and one master node. All computational nodes did not communicate with each other. Each node represents a personal computer with Intel Core i5–6600 CPU and 8GB RAM.

Table 2. Definitions of benchmark functions

Function	Definition	Global minimum
Griewank	$\Phi_1(X) = \sum_{i=1}^{\lvert X\rvert} x_i^2 4000 - \prod_{i=1}^{\lvert X\rvert} \cos\left(x_i\sqrt{i}\right) + 1$	$\Phi_1(X^*) = 0$ $X^* = (0, \ldots, 0)$
Rastrigin	$\Phi_2(X) = \sum_{i=1}^{\lvert X\rvert}(10 + x_i^2 - 10\cos(2\pi x_i))$	$\Phi_2(X^*) = 0$ $X^* = (0, \ldots, 0)$
Rosenbrock	$\Phi_3(X) = \sum_{i=1}^{\lvert X\rvert} \left(100(x_{i+1} - x_i^2)^2 + (1 - x_i^2)\right)$	$\Phi_3(X^*) = 0$ $X^* = (1, \ldots, 1)$
Zakharov	$\Phi_4(X) = \sum_{i=1}^{\lvert X\rvert} x_i^2 + \left(\sum_{i=1}^{\lvert X\rvert} 0.5\ i\ x_i\right)^2 + \left(\sum_{i=1}^{\lvert X\rvert} 0.5\ i\ x_i\right)^4$	$\Phi_4(X^*) = 0$ $X^* = (0, \ldots, 0)$

The obtained results (Tab. 3) demonstrate superiority of the proposed *SMEC/LA* algorithm in comparison with the simple parallel *MEC* in terms of accuracy.

Table 3. Results of numerical experiments

Function	*SMEC*	*SMEC/LA*
Griewank	$\bar{\Phi}_1 = 1.63\text{E}{-2}$	$\bar{\Phi}_1 = 2.44\text{E}{-5}$
	$\Phi_1^* = 9.19\text{E}{-3}$	$\Phi_1^* = 1.21\text{E}{-5}$
Rastrigin	$\bar{\Phi}_2 = 7.27\text{E}{+1}$	$\bar{\Phi}_2 = 7.69\text{E}{-3}$
	$\Phi_2^* = 5.25\text{E}{+1}$	$\Phi_2^* = 3.54\text{E}{-6}$
Rosenbrock	$\bar{\Phi}_3 = 2.73\text{E}{+1}$	$\bar{\Phi}_3 = 8.91\text{E}{-1}$
	$\Phi_3^* = 2.12\text{E}{+1}$	$\Phi_3^* = 9.45\text{E}{-2}$
Zakharov	$\bar{\Phi}_4 = 2.79\text{E}{+1}$	$\bar{\Phi}_4 = 1.16\text{E}{+0}$
	$\Phi_4^* = 2.10\text{E}{+1}$	$\Phi_4^* = 1.13\text{E}{-1}$

For all benchmark functions, results obtained with the use of *SMEC/LA* are better than ones obtained using just parallel *SMEC* by several orders of magnitude both for average values $\bar{\Phi}$ and least found values Φ^*.

On the other hand, the *SMEC/LA* algorithm requires more iterations N_{max} than *SMEC* algorithm (Fig. 3). This can be explained by the fact that both parallel algorithms wait until all computations are over; and in case of the *SMEC/LA* algorithm, it contains groups made of individuals with large values of the objective function. Such a group can require many iterations to converge to some local optimum. The comparison of the number of iterations N_{opt} required to find a global optimum proves our assumption (Fig. 4). The results demonstrate that the global optimum with the use of *SMEC/LA* method can be found relatively rapidly and, therefore, some unnecessary iterations can be avoided with the use of more advanced termination criteria.

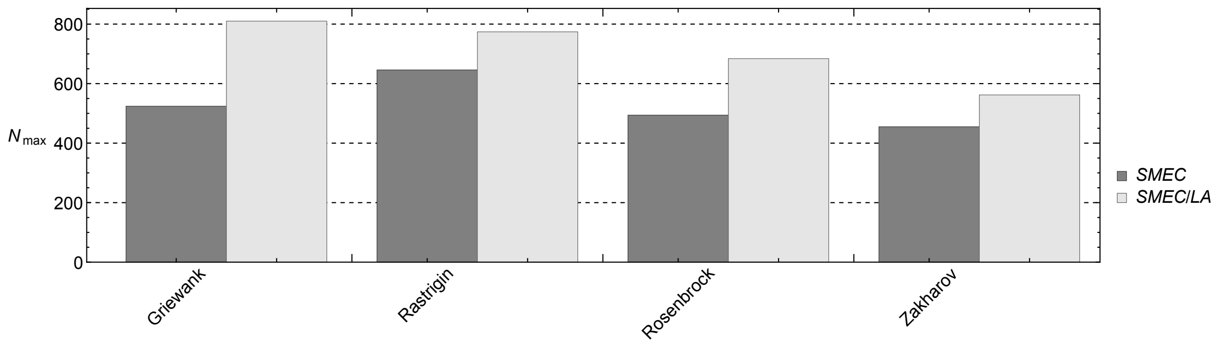


Figure 3. Overall iteration number N_{max} for different benchmark functions

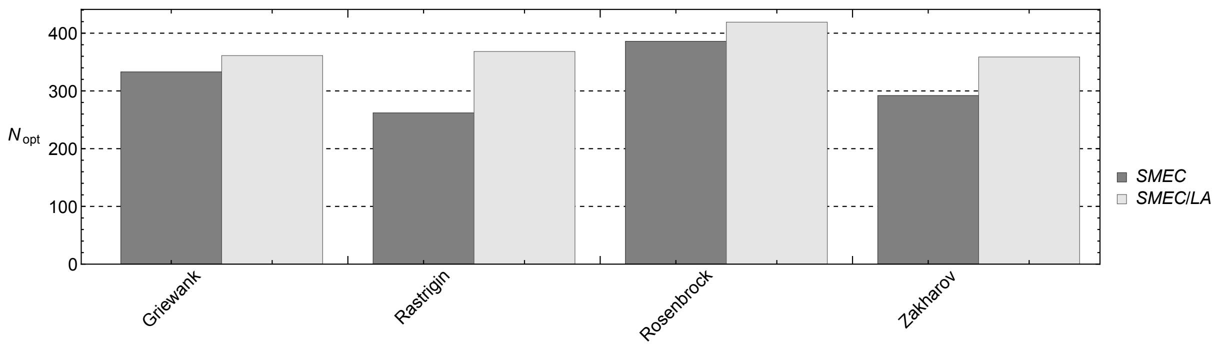


Figure 4. Iteration number N_{opt} required to find a global optimum for different benchmark functions

Conclusions

This paper presents a new adaptive load balancing method designed for the multi–population *SMEC* algorithm with the incorporated landscape analysis procedure. The algorithm is capable of adapting to various objective functions using an adaptation based on the results of objective functions classification.

Performance investigation was carried out in this work with the use of multi–dimensional benchmark optimization functions of various classes. The proposed algorithm along with the proposed load balancing technique was capable of localizing all known global optima with high precision. The parallel algorithm worked in the synchronous mode; that led to a significant increase in the maximum number of iterations. This disadvantage can be overcome using either asynchronous mode or some advanced termination criteria.

Further research will be devoted to a more extensive study of various objective functions and their classification as well as the efficiency of adaptation.

References

1. Bischl, B., Mersmann, O., Trautmann, H., Preub, M.: Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In: Proceedings of the fourteenth

international conference on Genetic and evolutionary computation conference, New York, NY, USA. pp. 313–320 (2012), DOI: 10.1145/2330163.2330209

2. Chengyi, S., Yan, S., Wanzhen, W.: A Survey of MEC: 1998-2001. In: IEEE International Conference on Systems, Man and Cybernetics IEEE SMC2002, Hammamet, Tunisia. October 6-9. Institute of Electrical and Electronics Engineers Inc., vol. 6, pp. 445–453 (2002), DOI: 10.1109/ICSMC.2002.1175629

3. Floudas, A.A., Pardalos, P.M., Adjiman, C., Esposito, W.R., Gumus, Z.H., Harding, S.T., Klepeis, J.L., Meyer, C.A., Schweiger, C.A.: Handbook of Test Problems in Local and Global Optimization. Kluwer, Dordrecht (1999)

4. Foster, I.: Designing and building parallel programs: concept and tools for parallel software engineering. Boston, Addison–Wesley (1995)

5. Gubenko, G.: Dynamic load Balancing for Distributed Memory Multiprocessors. Journal of parallel and distributed computing, 7, 279–301 (1989)

6. Jie, J., Han, C., Zeng, J.: An Extended Mind Evolutionary Computation Model for Optimizations. Applied Mathematics and Computation, 185, 1038–1049. (2007), DOI: 10.1016/j.amc.2006.07.037

7. Jie, J., Zeng, J.: Improved Mind Evolutionary Computation for Optimizations. In: Proceedings of 5th World Congress on Intelligent Control and Automation, 15–19 June 2004, Hang Zhou, China, pp. 2200–2204 (2004), DOI: 10.1109/WCICA.2004.1341978

8. Karpenko, A.P.: Modern algorithms of search engine optimization. Nature–inspired optimization algorithms. Moscow, Bauman MSTU Publ. (2014) (in Russian)

9. Karpenko, A.P., Sakharov, M.K.: Multi–memetic global optimization based on MEC. Information Technologies, 7, 23–30 (2014)

10. Kerschke, P., et al.: Cell mapping techniques for exploratory landscape analysis. EVOLVE A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation V, Springer, 115–131 (2014), DOI: 10.1007/978-3-319-07494-8_9

11. Liang, J.J., Qu, B.Y., Suganthan, P.N.: Problem Definitions and Evaluation Criteria for the CEC 2014. Special Session and Competition on Single Objective Real–Parameter Numerical Optimization. Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Technical Report, Nanyang Technological University, Singapore (2013)

12. Mersmann, O., et al.: Exploratory landscape analysis. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation, July 12–16, Dublin, Ireland. ACM, pp. 829–836 (2011), DOI: 10.1145/2001576.2001690

13. Munoz, M.A., Kirley, M., Halgamuge, S.K.: Exploratory landscape analysis of continuous space optimization problems using information content. IEEE Transactions on Evolutionary Computation, 19(1), 74–87 (2015), DOI: 10.1109/TEVC.2014.2302006

14. Sakharov, M., Karpenko, A.: A New Way of Decomposing Search Domain in a Global Optimization Problem. In: Proceedings of the Second International Scientific Conference Intelligent Information Technologies for Industry (IITI17). Springer, pp. 398–407 (2018), DOI: 10.1007/978-3-319-68321-8_41

15. Sakharov, M., Karpenko, A.: New Parallel Multi–Memetic MEC–Based Algorithm for Loosely Coupled Systems. In: Proceedings of the VII International Conference on Optimization Methods and Application Optimization and applications OPTIMA–2016, September 25 – October 2, Petrova, Montenegro. pp. 124–126 (2016)

16. Sakharov, M., Karpenko, A.: Performance Investigation of Mind Evolutionary Computation Algorithm and Some of Its Modifications. In: Proceedings of the First International Scientific Conference Intelligent Information Technologies for Industry (IITI16), Springer, pp. 475–486 (2016), DOI: 10.1007/978-3-319-33609-1_43

17. Sakharov, M.K., Karpenko, A.P., Velisevich, Ya.I.: Multi–Memetic Mind Evolutionary Computation Algorithm for Loosely Coupled Systems of Desktop Computers. Science and Education of the Bauman MSTU, 10, 438–452 (2015), DOI: 10.7463/1015.0814435

18. Sobol, I.M., Distribution of points in a cube and approximate evaluation of integrals. USSR Comput. Maths. Phys., 7, 86–112 (1967)

19. Vassilev, V., Fogarty, T., Miller, J.: Smoothness, ruggedness and neutrality of fitness landscapes: from theory to application. Advances in evolutionary computing. New York, NY, USA, Springer, 3–44 (2003)

20. Voevodin V.V., Voevodin Vl. V.: Parallel Computations. SPb.: BHV–Peterburg, (2004) (In Russian)

DOI: 10.14529/jsfi180402

Multicore Platform Efficiency Across Remote Sensing Applications

***Ekaterina O. Tyutlyaeva*[1], *Alexander A. Moskovsky*[1], *Igor O. Odintsov*[2], *Sergey S. Konyukhov*[1], *Alexey A. Poyda*[3], *Mikhail N. Zhizhin*[4,5], *Igor V. Polyakov*[4,6]**

A wide range of modern system architectures and platforms targeted for different algorithms and application areas is now available.

Even general-purpose systems have advantages in some computation areas and bottlenecks in another. Scientific applications on specific areas, on the other hand, have different requirements for CPU performance, scalability and power consumption.

The best practice now is algorithm/architecture co-exploration approach, where scientific problem requirements influence the hardware configuration; on the other hand, algorithm implementation is re-factored and optimized in accordance with the platform architectural features.

In this research, two typical modules used for multispectral nighttime satellite image processing are studied:

- measurement of local perceived sharpness in visible band using the Fourier transform;
- cross-correlation in a moving window between visible and infrared bands.

Both modules are optimized and studied on wide range of up-to-date testbeds, based on different architectures. Our testbeds include computational nodes based on Intel Xeon E5-2697A v4, Intel Xeon Phi, Texas Instruments Sitara AM5728 dual-core ARM Cortex-A15, and NVIDIA JETSON TX2.

The study includes performance testing and energy consumption measurements. The results achieved can be used for assessing serviceability for multispectral nighttime satellite image processing by two key parameters: execution time and energy consumption.

Keywords: energy consumption, performance analysis and optimization, cross-platform analysis, energy consumption analysis, multispectral image processing, nighttime remote sensing.

Introduction

This paper describes the cross-platform analysis of the nighttime remote sensing multispectral image processing algorithms.

The timeliness and relevance of the nighttime remote sensing was reaffirmed by such studies as correlation of electric lighting on the Earths surface with socioeconomic trends [1], monitoring of night fishing boat lights [2], detection and characterization of combustion sources [3], and global survey of natural gas flaring [4].

The first step to design a suitable HPC system for processing remote sensing data is analyzing the applicability of modern platforms to the typical algorithms used in multispectral image processing.

A wide range of modern system architectures and platforms targeted for different algorithms and application areas is now available. Even general-purpose systems have advantages in some computation areas and bottlenecks in another.

[1]RSC Technologies, Moscow, Russian Federation
[2]RSC Labs, Moscow, Russian Federation
[3]National Research Centre "Kurchatov Institute", Moscow, Russian Federation
[4]Space Research Institute, Moscow, Russian Federation
[5]University of Colorado Boulder, Boulder, USA
[6]Department of Chemistry, Moscow State University, Moscow, Russian Federation

Following the current trends in system architecture, all the experiments were conducted on different platforms:

- Modern Intel® architectures Intel® Xeon® E5 (2697A v4).
- multicore architecture Intel® Xeon Phi® 7250.
- GPU Nvidia Jetson TX2.
- Dual-core ARM (Texas Instruments Sitara AM5728 dual-core ARM Cortex-A15).

Scientific applications used in specific areas have different requirements for CPU performance, scalability and power consumption. The co-design approach is widely considered as best practice for designing an effective and economical system. This approach implies that scientific problem requirements influence the hardware configuration; on the other hand, algorithm implementation is re-factored and optimized in accordance with the platform architectural features.

Two typical algorithms used in nighttime image processing have been selected therefore for further cross-platform examination:

- `Correlation Module` Inter-channel image cross-correlation in a moving window.
- `Sharpness Module` Spectral and spatial measure of local perceived sharpness [9].

The study was conducted using multispectral images from VIIRS radiometer onboard of Suomi National Polar Partnership (SNPP) satellite.

The image processing modules have been implemented and optimized for the different hardware architectures.

According to the latest trends, the key research issue remains in providing a holistic solution that can collectively minimize energy consumption by HPC facility [5]. So, the energy consumption study in addition to the performance analysis is required to choose the most suitable architecture for future HPC system.

This paper mainly contributes to characterization of selected HPC architectures in terms of energy consumption and execution time while running the remote sensing data processing tasks. To perform this study, different implementation and architecture-dependent optimization of a source code were developed, and both time and energy consumptions were measured and analyzed using the chosen architectures.

The rest of the paper is structured as follows. Section 1 reviews earlier research of performance and energy consumption in hyperspectral imaging field. Section 2 provides a detailed specification of the compared architectures. After that, Section 3 describes the selected benchmarking algorithms used in nighttime image processing. Section 4 describes the parallel implementation and architecture-specific optimizations of these algorithms on selected architectures. Section 5 describes software and hardware used for measurements. Section 6 provides tables with measured results as well as the testing protocol. Finally, the last Section concludes the paper with discussion of the obtained results.

1. Related Work

Advances in sensor technologies are resulted in substantial increase in spatial, spectral and temporal resolution of satellite imagers. For example, Visible Infrared Imaging Radiometer Suite (VIIRS) onboard of the Suomi NPP satellite generates 3 TB of multispectral images for every month of nighttime observations. Both re-processing of the full 6-year archive of the nightime images and recent addition of the second sattelite with the same imager require an upgrade in energy efficiency and computing performance of the current processing environments.

There are some research efforts aimed at energy efficient processing of hyperspectral image data. For example, in 2013 Remon et al. [6] presented a detailed assessment of performance and energy consumption of hyperspectral unmixing algorithms on multi-core platform equipped with 4 AMD Opteron 6172 processors.

Another study entitled "Energy consumption characterization of a Massively Parallel Processor Array (MPPA) platform running a hyperspectral SVM classifier" [7] presents a study of the MPPA-256-N power dissipation and energy consumption while running a SVM hyperspectral classifier. This paper also includes comparison with GPU 780Ti GTX.

2. Hardware

In Tab. 1, codenames and specifications of the studied testbeds are listed.

Table 1. Testbeds Specifications

Codename	CPU	# Cores	Memory	GPU (subject to the availability)
Broadwell	Intel® Xeon® E5-2697A v4	2x 16	128 GB DRAM DDR4/2133MHz	-
KNL	Intel® Xeon Phi® 7250 7250	68	MCDRAM Intel® 16GB + Intel® 16GB + 32GB DDR4/2133MHz	-
ARM	Texas Instruments Sitara AM5728 dual-core ARM Cortex-A15/1.5GHz	2	DDR3, 2 GB	Not used: 2x PowerVR SGX544 3D GPU cores; Vivante GC320 2D GPU core
Jetson	ARM Cortex-A57 (quad-core)/2GHz + NVIDIA Denver2 (dual-core) /2GHz	6	8GB 128-bit LPDDR4/1866Mhz	256-core Pascal/1300MHz

3. Algorithms

3.1. Sharpness

`Sharpness` module is the most computationally intensive part of the automatic system for detecting fishing boat lights from nighttime images of the VIIRS multispectral radiometer [2].

VIIRS Boat Detector (VBD) considers all isolated bright spikes that are sharply visible on the sea's night surface as candidates for boats. In the moon light, the interference by clouds and lunar glint are taken into account as well. This Sharpness Module processes visible images from the VIIRS Day/Night Band (DNB). If part of the image appears blurry according to the Sharpness Module result, it will be discarded from the search for the isolated electric lights from boats.

The flow graph of the module is shown in Fig. 1

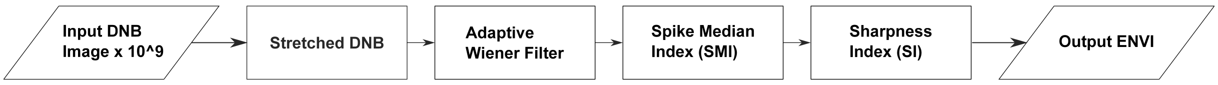


Figure 1. Sharpness Module Flow Graph

The `Sharpness module` reads input from the VIIRS DNB image stored in HDF5 format. Output data are stored in binary ENVI format. Data processing includes the following steps:

- Logarithmic transformation of the brightness histogram (stretch).
- Applying the Wiener filter [8].
- Computing the Spike Median Index (SMI).
- Computing the Sharpness Index (SI) [9] in a moving window of Block Size × Block Size. The Direct Fourier Transforms and Overdetermined real linear systems solving routines are repeatedly performed during this step.

3.2. Cross-Correlation

The `Cross-correlation` module calculates correlations between two spectral bands, visible and infrared. The main idea of the algorithm is validating the detected sources in different spectral bands under moonlit conditions.

The validation is carried out by performing a synchronous computing the linear Pearson's correlation between the corresponding moving windows in two spectral images. If the visible and infrared images are locally well-correlated, it means that the signal in the visible images is coming from moonlit clouds. If the local correlation is weak, it means that the visible signal is coming from the sea surface.

4. Implementation Details

The original versions of both algorithms were implemented using Matlab programming language.

We implemented the studied modules using C++. The source code was refactored to reach the maximal level of compiler-assisted optimization. The final C++ version of the code was implemented in a straight-line manner; all repeatedly performed loops had single entry and single, not data-dependent exit.

Input and output data details for both algorithms are presented in Tab. 2. HDF5-1.8.19 was used for parsing and reading HDF5 data.

4.1. Intel Version

In order to achieve the best performance on Intel testbeds, the vectorization features were used. In this context vectorization means using of the Intel SSE instruction set, which is an extension to the x86 architecture [10].

The efficient memory access was used by data alignment to the 32 byte boundaries (for Intel Advanced Vector Extensions (Intel AVX)) and 64 byte boundaries (for Intel AVX-512).

Table 2. Data Specifications

Algorithm	Stage	Data Format	Size, MB	Data Type	Dimensions H×W
Sharpness	Input	HDF5/DNB	60	Double	3072×4064
	Output	ENVI	96	Double	3072×4064
Cross-Correlation	Input	ENVI,HDF5/DNB	48,60	Float, Double	3072×4064
	Output	ENVI	96	Double	3072×4064

Intel compiler pragmas were used to inform the compiler of where it can safely ignore data dependencies and to inform that data is aligned.

Repeated operations with data arrays were implemented in a consecutive manner to use direct load from memory in a single SSE instruction.

Moreover, the typical trip count of the loop based on the typical image size is advised to the compiler in the `Cross-Correlation` module.

Mathematical calculations such as vector logarithm computation, Direct Fourier Transforms and solving the Overdetermined real linear systems were performed using the Intel MKL Library (2017).

The Processor-specific options of the form `-ipo -O3 -xMIC-AVX512 (for KNL)/-xCORE-AVX2 (for BRW)` were used to generate optimized and specialized code for processors.

The hybrid (MPI + OpenMP) parallelization scheme for an efficient application of the multicore architectures was used for this implementation. Each MPI rank processes its own images independently, so there are minimum communications between the processes. The OpenMP threads were used on Sharpness Index computation step. OpenMP threads process independent data in different positions of a moving window.

This version was tested on Intel testbeds with codenames Broadwell and KNL (Tab. 1).

4.2. ARM Version

The ARM-optimized version of FFTW3 open source library was used for Direct Fourier Transforms. The LAPACK library (3.7.1-4) was used for solving the overdetermined real linear systems. The processor-specific options were used for compilation.

Current implementation uses only ARM Cortex-A15 cores; the GPU cores are idle during the computation. So, the ARM testbed still have room for code optimization to achieve maximum possible performance.

The simple MPI-only parallelization scheme was used for this implementation, where each MPI rank processes its own images independently. An additional OpenMP parallelization layer is not required in this case due to the absence of hyperthreads. We used MPICH MPI implementation optimized for the ARM.

This version was tested on the ARM testbed (Tab. 1).

4.3. CUDA Version

The `Sharpness` algorithm is optimized for Jetson testbed according to the algorithm's logical structure described in subsection 3.1

The preparation steps, such as data input, stretch, applying the Wiener filter, and computing the Spike Median Index (SMI) are performed on CPU. The most computationally intensive step (computing the Sharpness Index) is implemented using CUDA (V8.0.62). This step is performed using GPU cores only.

The `Cross-Correlation` algorithm is also implemented using CUDA. The number of threads used in each block is justified with the image's width; the number of blocks is justified with the image's height.

CUDA threads effectively use GPU resources, so the MPI layer is not used in this implementation.

This version was tested on Jetson testbed (Tab. 1).

5. Measuring Equipment

Running Average Power Limit (RAPL) energy sensors, available in recent Intel CPUs, were used for measuring energy consumption for Intel testbeds (Broadwell and KNL). According to the Intel research [11], RAPL software power closely follows the actual power measurements. RAPL reports various energy readings. This includes energy consumption for the processor packages and the DRAM packages.

PAPI library [12] was used on Intel testbeds as an interface to RAPL energy consumption measurements [13]. PAPI provides a uniform access to performance counters as well as to RAPL data, so it provides the opportunities for enhanced measurements in the feature.

Hantek DSO2000 Series USB Oscilloscope [14] was used for power measurements for ARM and NVidia testbeds. Electric current was measured in ampers at every second of testing. Voltage was measured before execution of test series.

The execution time was measured using PAPI Library ($PAPI_get_real_nsec()$ function) on all testbeds.

6. Performance and Energy Consumption Study

This section presents experimental results of processing of time and energy consumption measurements of the modules reported in Section 3, measured using the equipment described in Section 5 on the testbeds listed in Section 2.

The testing procedure consisted of measurements regarding energy consumption and execution time. The testing procedure included a series of 10 executions per each combination of input data set and input feature sets.

Appropriate preparatory steps had been done prior to each execution, especially removing the results of previous computations and cleaning up the caches and swap.

The aggregate result is calculated as a median value of the measured results. Median value is used for understanding the central tendency of benchmarking results and for filtering out values that are skewing the results (for example, abnormally big values caused by temporal system processes' routines).

Input data for parallel processing was duplicated, so each of MPI rank processes separates a copy of input data. (According to the real case of archive processing, where each MPI rank should process a separate image). The numbers of MPI processes and CUDA threads are carefully adjusted according to the available number of cores and implementations for each architecture.

Testing results for the `Sharpness` module are listed in Tab. 3; results for the `Cross-Correlation` module are shown in Tab. 4.

Table 3. `Sharpness` Module Execution Statistics

Characterization	Broadwell	KNL	ARM	Jetson
Images processed	32	68	2	1
Execution Time, sec	39.229	206.121	355.5	33
Energy Consumed, J	9947	30757	3511	258

Table 4. `Cross-Correlation` Module Execution Statistics

Characterization	Broadwell	KNL	ARM	Jetson
Images processed	32 (× 2)	68 (× 2)	2 (× 2)	1 (× 2)
Execution Time, sec	25.7	39	20	5
Energy Consumed, J	4382	5107	179	27

It is important to note that measuring tools used in this research (see Section 5) oversee global energy consumption of the system, not just the energy consumed by the module under study. So, energy consumption results listed in Tab. 3, 4 refer to the total testbeds' consumptions during execution of the studied module.

As stated above, the number of processes and threads was selected according to the architecture requirements and implementation details. For Intel architectures in particular the optimal number of MPI processes refers to the number of physical cores; the number of OMP threads refers to the number of hyperthreads per core (2 OMP threads per MPI process for Broadwell testbed and 4 OMP threads per MPI process for KNL testbed). For ARM testbed, only MPI processes weere used. Finally, only CUDA threads were used for Jetson testbed. Thus, the number of pictures, processed in a parallel, differs for each testbed; execution time and consumed energy also vary in a wide range. So, it is difficult to define the appropriate testbed for these modules.

In this context, in is important to outline that rapid technological progress in multispectral imaging area stimulates new methods and challenges coming to existense in analysis and interpretation of hyperspectral data sets. This, in turn, leads to re-processing of data collected over the last year(s). So, the re-processing procedure is maintained systematically.

According to the current data, one Visible Infrared Imaging Radiometer Suite (VIIRS) day/night band (DNB) image corresponds to 5 min observations' data. Therefore, observation data archived for 1 year contains approximately 52560 DNB images. Table 5 shows the estimated time and energy to process a 1-year archive using the the `Sharpness` module in conformity with the experimental results mentioned above.

Table 5. `Sharpness` Module's Estimated Time To Process a 1-year archive

Characterization	Broadwell	KNL	ARM	Jetson
Time to Process, hours	17.9	44.3	2595.15	481.8
Energy Consumed, kJ	16337.9	23773.3	92269.08	13560.48

As shown in Tab. 5, the best energy consumption (13560 kJ) is reported for Jetson testbed. Following a close second is Broadwell testbed with 16337 kJ estimated energy consumption to process a 1-year archive. However, the execution time is much longer in Jetson testbed (481 hours corresponding to 20 days), while Broadwell testbed should complete processing of the archive with 18 hours.

As an alternative solution, the GPU cluster can be constructed to reduce the computation time and increase the performance. While this approach could benefit for the time of processing, energy consumption would be increased due to communication costs. Moreover, it is worth noting that increasing the number of components affects resilience of the solution.

Conclusion

This paper presents a research regarding execution time and energy consumption at different testbeds while running a multispectral image processing module.

Intel® Xeon® E5 and Nvidia Jetson TX2 demonstrated the most efficient results regarding computation performance and energy consumption criteria.

As a result, Intel® Xeon® E5 can be recommended for periodical re-processing of large archives of multispectral images in a reasonable time (days or weeks of full HPC cluster load). NVidia Jetson TX2 could be used for near real-time image processing, for example, at a direct receiving station, because it shows good results in per-picture processing.

However, ARM testbed must be further studied to fully exploit its potential. We intend to continue this work in the following directions:

- In the nearest future, we are planning to study other types of architectures, including Russian VLIW Elbrus CPUs and Intel® Xeon® Scalable Processors.
- We are planning to carry out a more detailed analysis of correlations between energy consumption and other performance metrics, including cache misses, the number of cycles and executed instructions, and so on.

Designing an energy-efficient system for processing multispectral observation data is a complex task that introduces new programming and optimization challenges. However, the results listed in this paper could be helpful for selection of the most appropriate architectures.

Acknowledgements

This research was supported by the grant from the Russian Ministry of Education and Science (The government contract 14.607.21.0165, unique identifier RFMEFI60716X0165).

References

1. Elvidge, C., Baugh, K., Zhizhin, M., Hsu F.-C., Ghosh T.: VIIRS night-time lights, International Journal of Remote Sensing 38(21), 5860–5879 (2017), DOI: 10.1080/01431161.2017.1342050

2. Elvidge, C., Zhizhin, M., Baugh, K., Hsu, F.-C.: Automatic Boat Identification Sys-

tem for VIIRS Low Light Imaging Data. Remote Sensing Journal 7(3), 3020–3036 (2015), DOI: 10.3390/rs70303020

3. Elvidge, C., Zhizhin, M., Hsu, F.-C., Baugh, K.: What is so great about nighttime VIIRS data for the detection and characterization of combustion sources? In: Proceedings of the Asia-Pacific Advanced Network, vol. 35. pp. 33–48. DOI: 10.7125/APAN.35.5

4. Elvidge, C., Zhizhin, M., Baugh, K., Hsu, F.-C., Tilottama, G.: Methods for Global Survey of Natural Gas Flaring from Visible Infrared Imaging Radiometer Suite Data. Energies 9(1), 14 (2015), DOI: 10.3390/en9010014

5. Khan, S., Bouvrym, P., Engel, T.: Energy-efficient highperformance parallel and distributed computing. The Journal of Supercomputing 60(2), 163–164 (2012)

6. Remn, A., Snchez, S., Bernab, Quintana-OrtAntonio, E., Plaza, A.: Performance versus energy consumption of hyperspectral unmixing algorithms on multi-core platforms. EURASIP Journal on Advances in Signal Processing 2013(1), 68 (2013), DOI: 10.1186/1687-6180-2013-68

7. Madroal, D., Lazcano, R., Fabelo, H., Ortega, S., Salvador, R., Callic, G. M., Jurez, E., Sanz, C.: Energy consumption characterization of a Massively Parallel Processor Array (MPPA) platform running a hyperspectral SVM classifier. In: Design and Architectures for Signal and Image Processing (DASIP) 2017, 27–29 September, Dresden, Germany. pp. 1–6. IEEE (2017), DOI: 10.1109/DASIP.2017.8122112

8. Lim, J. S.: Two-Dimensional Signal and Image Processing. Englewood Cliffs, NJ, Prentice Hall (1990)

9. Vu, C.T., Phan, T.D., Chandler, D.M: S3: A spectral and spatial measure of local perceived sharpness in natural images. IEEE Trans. Image Processing, 21(3), 934–945 (2012), DOI: 10.1109/TIP.2011.2169974

10. Intel Corporation: A Guide to Vectorization with Intel C++ Compilers (2012) `https://software.intel.com/en-us/articles/a-guide-to-auto-vectorization-with-intel-c-compilers`, accessed: 2018-11-29

11. Rotem, E., Naveh, A., Ananthakrishnan, A., Weissmann, E., Rajwan, D.: Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge. IEEE Micro, 32(2), 20–27 (2012), DOI: 10.1109/MM.2012.12

12. Jagode, H., YarKhan, A., Danalis, A., Dongarra, J.: Power Management and Event Verification in PAPI. In: 9th Parallel Tools Workshop, Dresden, Germany, September 2–3, 2015. pp. 41–51 (2015), DOI: 10.1007/978-3-319-39589-0_4

13. Weaver, V. M., Johnson, M., Kasichayanula, K., Ralph, J., Luszczek, P., Terpstra, D., Moore, S.: Measuring Energy and Power with PAPI. In: 2012 41st International Conference on Parallel Processing Workshops. pp. 262–268, Pittsburgh, PA (2012), DOI: 10.1109/ICPPW.2012.3

14. Product Details Page: DSO2000 Series. `http://www.hantek.com/en/ProductDetail_2_44.html`, accessed: 2018-11-29

DOI: 10.14529/jsfi180403

A Study on Cross-Architectural Modelling of Power Consumption Using Neural Networks

***Miloš Puzović*[1]*, Eun Kyung Lee*[2]*, Vadim V. Elisseev*[3]**

On the path to Exascale, the goal of High Performance Computing (HPC) to achieve maximum performance becomes the goal of achieving maximum performance under strict power constraint. Novel approaches to hardware and software co-design of modern HPC systems have to be developed to address such challenges.

In this paper, we study prediction of power consumption of HPC systems using metrics obtained from hardware performance counters. We argue that this methodology is portable across different micro-architecture implementations and compare results obtained on Intel® `64`, IBM® POWER™ and Cavium ThunderX® ARMv8 microarchitectures. We discuss optimal number and type of hardware performance counters required to accurately predict power consumption.

We compare accuracy of power predictions provided by models based on Linear Regression (LR) and Neural Networks (NN). We find that the NN-based model provides better accuracy of predictions than the LR model. We also find, that presently it is not yet possible to predict power consumption on a given microarchitecture using data obtained on a different microarchitecture. Results of our work can be used as a starting point for developing unified, cross-architectural models for predicting power consumption.

Keywords: hpc, power consumption, modelling, exascale, cross-architectural, neural networks.

Introduction

Upcoming High Performance Computing (HPC) systems are on the critical path towards delivering the highest level of performance for large scale applications. If contemporary technologies were used to build even more powerful HPC systems, the power consumption required by those systems would be unsustainable, as it would require hundreds of megawatts of power. Thus, current HPC systems must be built considering *energy efficiency* as the first and foremost design goal. Currently, the most power efficient HPC system on the Green500 [26] list is Shoubu System B located at ACCC, RIKEN with 17GFlops/Watt. In order to achieve a sustainable power draw, future HPC systems will have to feature power efficiency of around 50GFlops/Watt. Such power efficiency levels require novel software/hardware co–design, with software guiding static and dynamic power management.

Successful development of such software relies on availability of tools for measuring power consumption and temperature. Temperature and power sensors have been introduced to provide these type of measurements. Unfortunately, as a result of the fabrication process used to build microprocessors, measurements of changes in power and temperature happen significantly later then the actual event (*thermal inertia*). An alternative to using power and temperature sensors would be to directly measure processor events that are causing power and temperature changes thus eliminating time lag limitation. The best proxy for measuring processor events is the *hardware performance counters*, because they offer a reliable interface to detect power and temperature variations within a real system. Considering a very large number of hardware performance counters typically available on modern systems, the task of selecting counters that are most representative of the full system power is becoming a challenge. Moreover, complexity of

[1]The Hartree Centre, STFC Daresbury Laboratory, Sci-Tech Daresbury, Cheshire, UK
[2]IBM T.J. Watson Research Center, Yorktown Heights, NY, USA
[3]IBM Research, STFC Daresbury Laboratory, Sci-Tech Daresbury, Cheshire, UK

this problem significantly increases when different types of micro–architecture implementations are considered. The matter is further complicated by the fact that each architecture has a limited number of registers that can be simultaneously recorded without resorting to multiplexing and therefore reducing the accuracy. For example, IBM POWER8 architecture can simultaneously track at most six different performance counters, while the latest Intel 64 architecture can track up to eight.

In this paper, we extend recent works [2, 13] on estimation of power consumption of HPC systems with metrics obtained using hardware performance counters on three different micro–architecture implementations: Intel 64 Broadwell, IBM POWER8 and Cavium ThunderX ARMv8 architecture and argue that the methodology is portable across different micro–architecture implementations. We study advantages and disadvantages of a model presented in [2] and [13] in dealing with emerging HPC workloads. We improve accuracy of power consumption predictions by adoptong a model based on Neural Networks (NN) and discuss the optimal number and the type of hardware performance counters required to accurately predict power consumption within few percents of the actual power consumption. We believe that our results can guide system designers in implementing hardware counters which measure similar events between different types of architectures, which in turn will allow developing more general models of power consumption.

The rest of the paper is structured as follows. Section 1 describes the use of hardware performance counters for power consumption estimations. Section 2 explains advantages and disadvantages of the power estimation model from [2, 13] and describes methodology that was used to collect data on three different HPC systems. Section 3 describes NN based approach to developing a more accurate prediction model and shows results that we have obtained using the new model to estimate power on emerging HPC workloads. Finally, the section entitled "Conclusions and Future Work" presents conclusions and future directions of our work.

1. Related Work

The use of hardware performance counters to estimate power consumption has been around for some time [10]. Such approach is appealing, because it does not require information about power consumption of individual functional units, but its accuracy relies on selecting the most suitable hardware performance counters and on having a *very* representative set of applications, which can fully characterise the target platform in order to build a reliable expression for power estimation. Previous studies [18] have shown that it is sufficient to use only instructions per cycle (IPC) to characterise behaviour of an operating system. In this case, the constructed power consumption model is a *static* power model. In [4], authors have proposed a *dynamic* power model, where the framework used to construct it contains a set of heterogeneous power models. In all these examples, the power model has been constructed only for a micro–processor and memories, but not for events related to a chipset, I/O and disk. For majority of workloads, that is acceptable in terms of accuracy as microprocessors and memories consume most of the power, and the rest of the system makes up between 10% and 20% of total power (dependant on the type of the workload). The work presented in [7] shows how to assess the total power of a system using hardware performance counters. The authors use the *trickle–down* approach where values of *power inducing* hardware performance counters are propagated within different subsystems in order to simulate the total power. The model they have built was accurate within 9%.

The work presented in [23] considers two processor configurations at the opposite ends of a performance spectrum: a low power processor and a high performance processor. The authors have found that there exists a subset of hardware performance counters, which allows evaluating dynamic power consumption for each architecture with an average error of 5%.

The study presented in [2] demonstrates that it is possible to estimate power consumption on HPC architectures for workloads with very high CPU utilisation (all cores are utilised at more than 90%) with an average error within 5%.

Machine learning–based modelling has been gaining popularity for optimising power and energy consumption. It has been used for power and energy modelling on HPC kernels with different code variants [27] and for predicting a user's demands from the usage history for managing idle servers [12]. Genetic algorithm has been used to predict power using a wide range of hardware activity counters [17]. Neural Network has been used to minimise the cooling energy consumption [1] of a server. Borghesi et al. [8] have proposed a machine learning approach, which relies on resource requests from a user and an application to estimate power consumption of HPC workloads. Their model was able to handle cases when CPUs were not fully utilised.

However, all previous models were dealing with a particular micro architecture. In this paper, we present in–depth study on power consumption evaluation using hardware performance counters within three different micro–architecture implementations: Intel 64 Broadwell, IBM POWER8 and Cavium ThunderX ARMv8.

2. Motivation

2.1. Linear Regression Model

As the starting point for predicting power consumption, we used the model introduced in [2]. The model's equation for power consumption of a given application as a function of CPU frequency f is the following:

$$\mathbf{P}(f) = A(f) \times \ \mathbf{P}(f_0) + B(f) \times \ \mathbf{TPI}(f_0) + C(f). \tag{1}$$

In equation (1), power consumption $\mathbf{P}$ of an application running with CPU frequency f is estimated by measurements performed at reference frequency f_0. In order to use equation (1), we need to measure, power and transactions per instruction ($\mathbf{TPI}$) for each application at frequency f_0. Transactions per instruction are defined as a ratio of the number of cache lines written to and read from memory ($\mathbf{C}$) to a number of instructions executed ($\mathbf{I}$) :

$$\mathbf{TPI}(f) = \frac{\mathbf{C}(f)}{\mathbf{I}(f)}.$$

Coefficients $A(f)$, $B(f)$ and $C(f)$ are system-specific. They are used for characterising power on a given HPC system at CPU frequency f. It is important to note that for every new microarchitecture implementation, it is always necessary to obtain new $A(f)$, $B(f)$ and $C(f)$ coefficients as they are not *transferable.*

Equation (1) assumes that we have a way to measure power consumed by an application at a specific frequency. This is not always possible as described in the Section . Furthermore, the majority of power sensors expose *power lag* and *distortion* where power consumption lags behind the actual benchmark activity and the shape of power consumption does not actually match the benchmark activity [9]. Therefore, we need to modify equation (1) to use measurements obtained from hardware performance counters. The new equation is:

$$\mathbf{P}(f) = A(f) \times \ \mathbf{GIPS}(f_0) + B(f) \times \ \mathbf{GBS}(f_0) + C(f). \qquad (2)$$

In equation (2), **GIPS** is *the instruction rate*, which is defined as a number of giga instructions executed per second, and **GBS** is *the memory bandwidth*, which is defined as a number of gigabytes written to and read from the memory per second. With such metrics, coefficients A, B and C at frequency f have the following meaning:

- A is the power consumed by all executed instructions.
- B is the power consumed by data transfers.
- C is the power consumed statically.

Table 1. Intel Xeon E5 v4, IBM Power System S822LC and Cavium ThunderX characteristics

Architecture		POWER8	Intel `x86-64`	ARMv8 64bit
Processor		IBM Power System S822LC	Intel Xeon E5-2698 v4	Cavium ThunderX
Core	Frequency	3.5 GHz	2.3 GHz	2.0 GHz
	# of cores	10	16	48
	# of threads	80	32	48
Execution unit	Type	out-of-order	out-of-order	in-order
	# of issue/commit	10 / 8	8 / 4	2 / 2
L1D Cache	Policy	NUCA	Write-allocate	Write-through
	Type	Private	Private	Private
	Size	64 KB/core	32 KB	32KB
	Associativity	8-way	8-way	32-way
L1I Cache	Size	32KB/core	32KB	78 KB
	Associativity	8-way	8-way	39-way
L2 Cache	Policy	NUCA	Write-back	Write-back
	Type	Private	Private	Shared
	Size	512KB/core	256KB	16MB
	Associativity	8-way	8-way	16-way
L3 Cache	Policy	NUCA	Write-back	N/A
	Size	8MB/core	40MB	N/A
	Type	Shared	Shared	N/A
SMP Interconnect	Bus Type	Integrated SMP interconnect	QPI	CCPI
	Bus speed	9.6 GB/s per channel	9.6 GB/s	10.3 GHz
Memory	Type	DDR4 1600	DDR4 2133	DDR4 2133
	# of channels	8	4	4
	Access speed	1600 MHz	2133 MHz	2100 MHz

2.2. Methodology

In order to estimate power as given in equation (2), we need to read hardware performance counters for the total number of instructions that have been executed (*or retired*) and the total number of bytes that have been read and written to a memory controller.

Tab. 1 shows relevant characteristics of three different microarchitecture implementations that we are using throughout the paper. Tab. 2 shows hardware performance counters that were used to measure events that are required to calculate **GIPS** and **GBS** for equation (2) for the model described in Section 2.1. We used architecture-independent `libpfm` library [14] to read performance counters for events listed in Tab. 2. This approach allowed us to run our profiled benchmarks unmodified on both architectures.

To find coefficients A, B and C, we ran a set of compute kernels as in [2] that provide a broad spectrum of **GIPS** and **GBS** characteristics. We used NAS Parallel Benchmarks (NPB) [3] and STREAM [19], which have been designed to test the performance of HPC systems. NPB suite is a mix of workloads that has been derived from computational fluid dynamics, unstructured

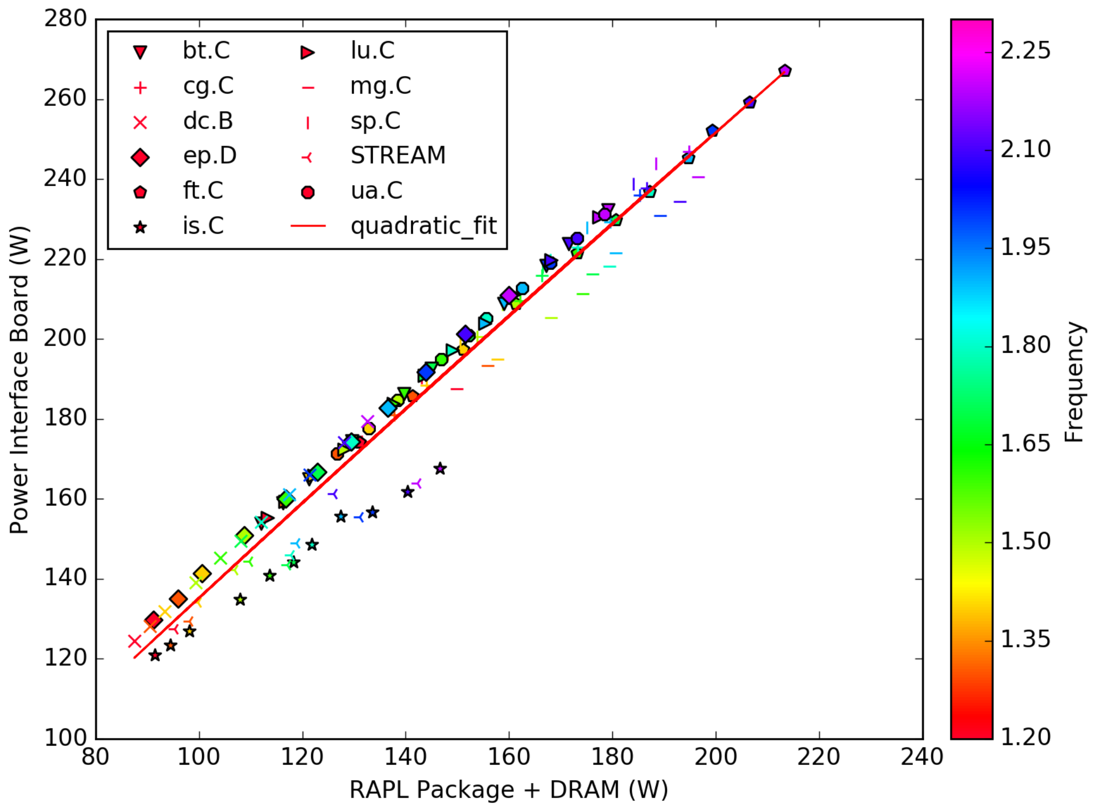


Figure 1. Correlation between the total node power and the power consumed by cores and memories

adaptive mesh, parallel I/O, multi-zone applications and computational grids, whilst STREAM is a simple synthetic benchmark that is used to measure sustainable memory bandwidth and corresponding computation rate for straightforward compute kernels. We used the OpenMPI [24] version of benchmarks with different classes/problem sizes in order to cover a wider range of workloads. As for the NPB suite convention, benchmark class is appended as a suffix to the benchmark's name; for example, `ft.C` stands for *discrete 3D fast Fourier Transform, class C.*.

Firstly, we studied which part of an HPC node contributes the most to power consumption. For brevity, results presented in this section were obtained on Intel Xeon E5 v4 microarchi-

Table 2. Hardware performance counters used on Intel Xeon E5 v4, IBM POWER8 and Cavium ThunderX microarchitecture implementations to calculate **GIPS** and **GBS** provided in equation (2)

Microarchitecture	Memory	Instructions
Intel Xeon E5 v4	UNC_CBO_CCACHE_LOOKUP.I UNC_CBO_CCACHE_LOOKUP.ANY_REQ UNC_ARB_TRK_REQUEST.EVICTIONS	EVEN_INSTR_RETIRED
IBM POWER8	PM_MEM_READ PM_MEM_PREF MEM_RWITM	PM_RUN_INST_CMPL
Cavium ThunderX ARM	L2D_CACHE_REFILL_LD L2D_CACHE_REFILL_ST L2D_CACHE_WB_VICTIM L2D_CACHE_WB_CLEAN	CPU_CYCLES

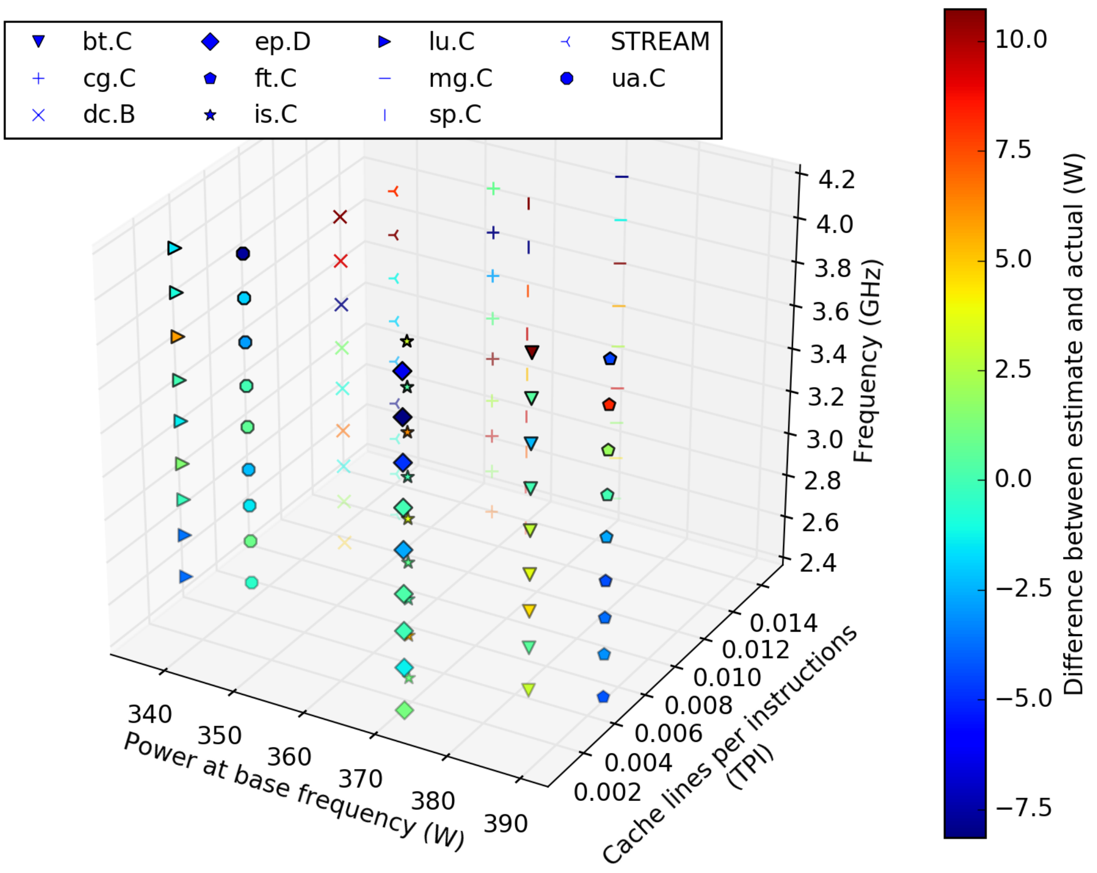


Figure 2. Accuracy of power consumption estimation of NPB and STREAM benchmarks using equation (2) on IBM POWER8

tecture, but we has the same observations on IBM POWER8 micro-architecture. In order to measure power consumption on IBM POWER8 (S822LC model) server, we used an On-Chip Controller (OCC) that collects temperature and power data from various sensors. This data is available either from a Baseboard Management Controller(BMC) via IPMI protocol or from the system memory via kernel module [5]. We used a power interface board (PIB) to get total power consumption on an Intel-based server. The PIB has a *hot-swap controller* (HSC), TI LM25062 [25] that measures and reports power. It works by having a shunt resistor and an I-sense/V-sense circuit. Thus, it is possible to position HSC with the sense circuit between the 12V DC power connector and the on-board voltage regulators to monitor the total power consumed by the attached HPC node. In addition to total power of the HPC node, we also measured power consumed only by processor and the memory. This was measured by reading information from model-specific registers (MSR) that are used for the Running Average Power Limit (RAPL) future. If there is high correlation between these two measurements for NPB benchmarks, then we can deduce that it is sufficient to only model cores and memory performance events in order to estimate total power as it can be calculated from cores and memory power using the fitted line. Fig. 1 shows the results we have obtained.

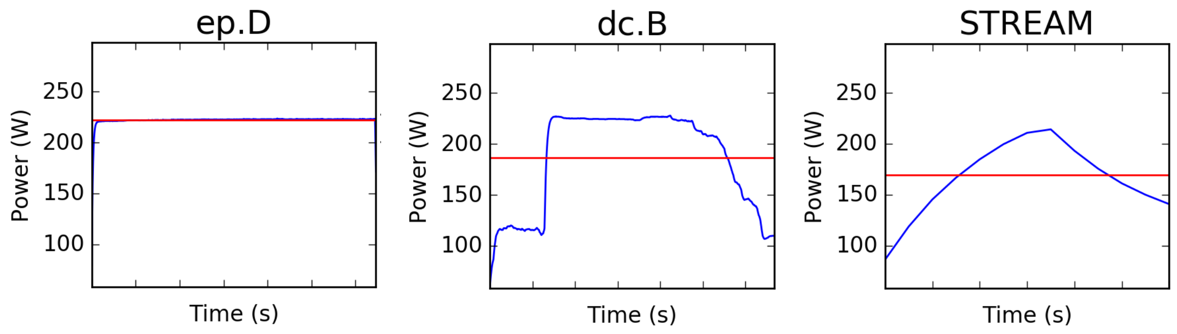


Figure 3. Actual instantaneous power (blue) and mean power (red) used by the model to represent power consumption of the application

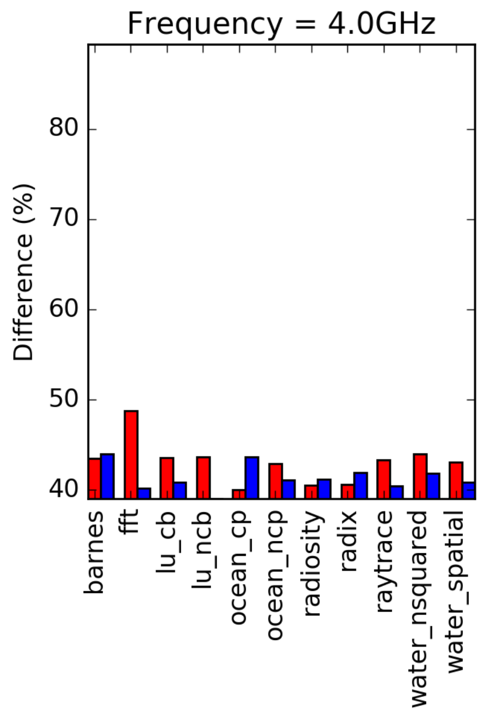
There is almost perfect correlation between RAPL and PIB values that can be approximated by a quadratic fit[4]. The outliers are three benchmarks *integer sort* (`is.C`), `STREAM` and *multigrid* (`mg.C`). These three benchmarks are using uncore events that are not covered by RAPL.

Secondly, in order to calculate coefficients A, B and C on Intel Xeon E5 v4, we ran NPB and STREAM benchmarks at CPU frequencies ranging from 1.2 GHz to 2.3 GHz in steps of 0.1 GHz and performed a linear regression using the obtained results and equation 2. We used the nominal frequency of 2.0 GHz as a baseline frequency (f_0). For IBM POWER8 we have used CPU frequencies ranging from 2.5 GHz to 4.0 GHz and nominal frequency of 3.4 GHz. Since Cavium ThunderX does not support power gating, we eliminated implementation of this microarchitecture in this test. As soon as we obtained the coefficients, we estimated accuracy of the model by comparing estimated and actual power consumption for the benchmarks used by regression analysis. Figure 2 shows the difference between the estimated power consumption using a linear regression model and actual power reading from IBM POWER8 for all NPB and STREAM benchmarks and frequencies from 2.5 GHz to 4.0 GHz.

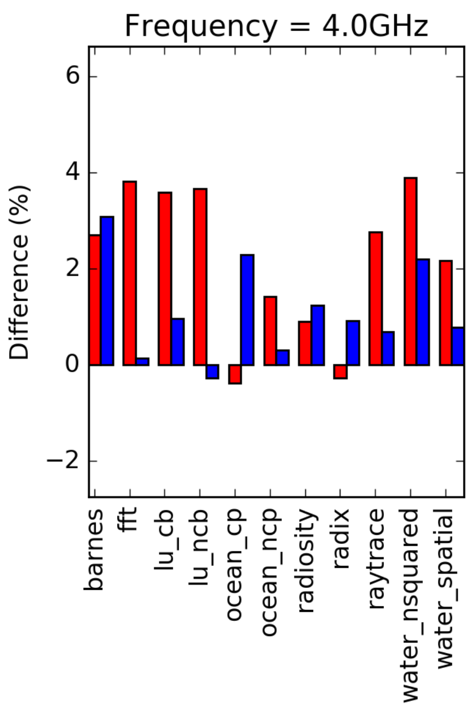
If the model is good, we expect most of benchmarks under various frequencies to be in one of the shades of green or light blue because in this case the difference between estimated and actual power is close to zero. This is applicable for such benchmarks as *unstructured adaptive mesh* (`ua.C`), *lower-upper Gauss-Seidel solver* (`lu.C`) and *block tri-diagonal solver* (`bt.C`). On the other hand, for such benchmarks as *data cube* (`dc.B`) and `mg.C`, accuracy depends on the CPU frequency. At higher frequency, the model overestimates the consumed power, while at lower frequencies it underestimates it. It also should be noted that for benchmarks `ft.C` and *embarrassingly parallel* (`ep.D`) the model estimates power quite accurately at high frequencies while it overestimates it at lower frequencies. Both of these behaviours could be caused by choosing the nominal frequency (namely, 3.4 GHz) as a baseline frequency. Furthermore, we can also notice on Fig. 2 that NPB and STREAM benchmarks do not cover the whole spectrum of possible combinations. For example, we can see that none of the benchmarks are in the top-right corner, where there is a high TPI and high average power consumption; and also, none are in the bottom-left corner where there is a low TPI and mid- to low-power consumption. This shows that some benchmarks might be over- or underestimated because they belong to an uncovered region.

Thirdly, the model provided in [2] uses power consumption averaged over the application's runtime. This approximation works well when there is no significant power variability during its

[4]The model of quadratic fit is $-0.0003 \times\ x^2 + 1.2614 \times\ x + 12.3405$



(a) Regression model



(b) Neural Network-based model

Figure 4. Power estimation accuracy of SPLASH2 benchmark suite on IBM POWER8

operation. But that is not always the case. Shown in blue in Fig. 3 is the shape of the running power during execution of the benchmarks, and shown in red is the constant power that is used for building the model. While comparing Fig. 3 with Fig. 2, We noticed that that model is fairly accurate for such benchmarks as `ep.D`, where there is an overlap between the blue and the red lines. Benchmarks such as `bt.C`, `lu.C` and `ua.C` belong to this category as well. For benchmarks in which it is not the case, such as `dc.B` and `STREAM`, we noted significant differences. This can have a significant impact on accuracy of the estimation.

Finally, in order to conduct further testing of the model, we ran all benchmarks from the SPLASH2 [28] benchmark suite. None of the results from these benchmarks were used by the linear regression model to find coefficients. Figure 4 shows results for POWER8 at the frequency of 4.0 GHz. The red bar is the whole benchmark run (including sequential and parallel regions of the code); the blue bar indicates that only parallel region of the code is measured. With only parallel region measured, the CPU utilisation is higher and, as a result, we expect to see higher accuracy of the power estimation model.

Figure 4 (a) shows that the model based on equation (2) provides accuracy of power evaluation only between 30% and 40%. It is a clear indication that the linear regression model has difficulty in covering a broader range of application with different execution profiles. Figure 4 (b) shows that the NN-based model is accurate within 5% for exactly the same range of parameters.

In the next Section, we will discuss in details how to use the Neural Network-based approach to improve accuracy of power estimation. In order to improve the accuracy as shown in analysis in this Section, it is not enough just to focus on performance counters that account for executed instructions and data transfers. If we apply the Neural Network-based model developed in the next Section on hardware performance counters from Tab. 2, we drop power consumption misprediction to a further extent than the one shown in Fig. 4 (b). Furthermore, it is necessary to increase the number of benchmarks in the training set in order to cover the wide spectrum of future application characteristics. In this regard, we start the next Section by describing additional hardware performance counters that we sampled in order to improve accuracy. We also review an extended benchmark set before describing the Neural Network model that utilities them.

3. The Neural Network Model

In this Section, we provide details of our Neural Network–based power consumption prediction model, as well as the training sets used and the model's validation. Our model improves linear regression model predictions while using the same hardware performance counters, and extends the model by taking into account a wider range of counters.

Our model is a proof–of–the-concept prototype that can make fine–grained power predictions. We used the same hardware given in Section 2 as described in Tab. 1. The MATLAB Neural Network Toolbox [11] is used for the modelling.

3.1. Hardware Performance Counters and Benchmarks

It is possible to reveal plenty of power consumption information by monitoring hardware performance counters because they directly measure events that correlate with the energy used during the application execution. The aim of our work is to find the minimum set of hardware performance counters that highly correlate with the amount of power consumed. As in [23], we looked at a large number of available hardware performance counters across three different micro architecture implementations and selected the following counters:

- **Instructions Per Cycle (IPC)** – it was previously shown that power consumption of a processor is highly dependent on this metric.
- **Dispatched/Fetched Instructions (IFETCH)** – the previous metric (IPC) only accounts for instructions that have been retired, but it doesn't take into account instructions that have been speculatively executed. These instructions still consume power. Therefore, we keep track of them using this counter.
- **Stalls (STALL)** – due to multiple issues and out-of-order execution, contemporary processors *stall* due to dependencies such as data and resource conflicts. The conflicts draw power and are not accounted by any of the previous counters.
- **Branch hit ratio (BR)** – in order to find contribution to power consumed of speculatively executed instructions due to branch misprediction, we also measure percentage of correctly predicted branches during the application execution and use that information to further refine the results from **IFETCH** and **STALL** hardware performance counters.
- **Floating point instructions (FLOPS)** – for HPC applications, the largest contributor towards power consumption are instructions that are utilising the floating point unit as they represent the majority of executed instructions.
- **Cache and memory hit and miss** – once there is a miss, the processor needs to bring data from the memory in order to operate on it; the power is required to move this data. Due to the fact that with the previous counters we only measured power that is consumed within the processor, we also measure the number of hits in local cache (**L1**) and the number of misses in the shared last level cache (**LCCM**).

Table 3. Hardware performance counters used on Intel Xeon E5 v4, IBM POWER8 and Cavium ThunderX microarchitecture implementations to record metrics as described in Section 3.1

Event	Intel Xeon E5 v4	IBM S822LC	Cavium ThunderX
IPC	EVENT_CPU_CLK_UNHALTED EVENT_INST_RETIRED	PM_RUN_CYC PM_RUN_INST_CMPL	CPU_CYCLES INST_RETIRED
IFETCH	EVENT_ISSUED	PM_INST_DISP	ISSUE
STALL	EVENT_RESOURCE_STALLS	PMU_CMPLU_STALL	STALL_BACKEND
BR	EVENT_BR_INST_EXEC EVENT_BR_MISP_EXEC	PM_BR_CMPL PM_BM_MPRED_CMPL	BR_RETIRED BR_MIS_RETIRED
FLOPS	FP_ARITH_INST	PM_FLOP	ASE_SPEC VFP_SPEC
L1	MEM_LOAD_RETIRED_L1_HIT	PM_DATA_FROM_L2	L1D_CACHE_REFILL L1D_CACHE
LCCM	UNC_CBO_CCACHE_LOOKUP.ANY_REQ UNC_CBO_CCACHE_LOOKUP.I UNC_ARB_TRK_REQUEST.EVICTIONS	PM_MEM_READ PM_MEM_PREF PM_MEM_RWITM	L2D_CACHE_REFILL_LD L2D_CACHE_REFULL_ST L2D_CACHE_WB_VICTIM L2D_CACHE_WB_CLEAN

The hardware performance counters that we have obtained are intuitive. Power consumption of the processors depends on the number of directly executed instructions (**IPC**). The power used to move data and instructions from memories before execution is accounted by **L1** and the **IFETCH** counter, while moving data from further away is modelled by **LCCM** and the processor stalls (**STALL**) whilst it is waiting for data to become available. Furthermore, the accuracy of the processor stalls is improved by tracking correctly predicted branches (**BR**) as well. Finally, since the focus of this work lies on HPC–like workloads, we are also tracking the floating point instructions (**FLOPS**) since they represent the majority of instructions in these types of workloads. Table 3 shows the events that we sampled for each of the three microarchitecture implementations shown in Tab. 1.

In order to increase the size of the training set for our model, we have used extra **four** benchmark suites in addition to the NPB benchmark suite. The first two benchmark suites are Splash-2 [28] and the Princeton Application Repository for Shared–Memory Computers (PARSEC) [6]. Splash2 is a mature benchmark suite that contains a wide range of HPC and graphic applications, while PARSEC consists of benchmarks which are representatives of *emerging* workloads found in domains of data mining and media processing. We used 12 benchmarks from each of Splash2 and Parsec suites . The third benchmark suite is Mantevo [15]. Mantevo pioneered the concept of using *miniapps* to drive hardware/software co–design. The miniapps are proxies which combine some or all aspects of dominant computational kernels into standalone applications. We used 12 miniapps from Mantevo benchmark suite, which cover domains of finite elements, molecular dynamics, contact detection and electrical circuits. The fourth benchmark suite is a set of proxy applications from LLNL [16] that represent Monte Carlo particle transport, radiation diffusion and *Livermore Loops*. We used 7 benchmarks from this suite.

3.2. Model Description and Validation

The computational neural network (NN) is inspired by biological neural networks to predict or approximate functions that can depend on a large number of inputs. There are two phases when in using a neural network: training and deployment. During the training phase, neurons in each layer are adjusted iteratively using the training data. Then, the trained neurons are used to predict the new output in the deployment phase. Fig. 5 shows our neural network design. First,

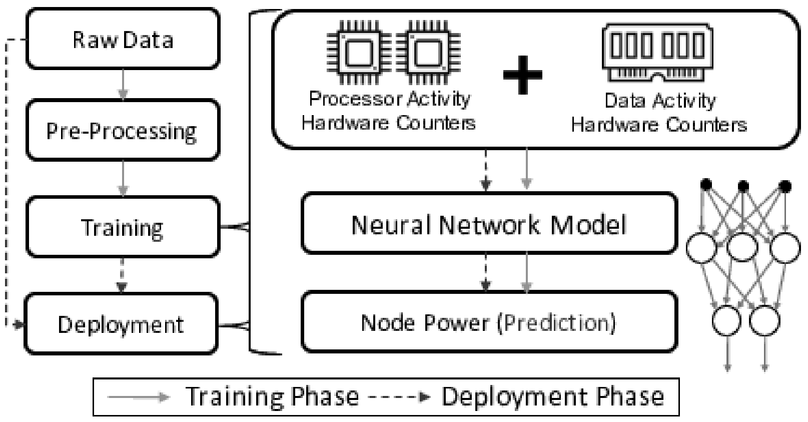


Figure 5. Neural Network–based prediction approach

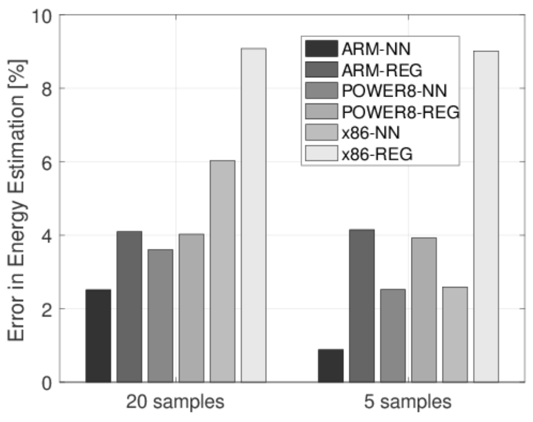
input data is pre-processed to obtain steady-state power. Second, pre-processed input data is fed to the input layer of the neural network for training.

In our case, the input layer consists of processor activity hardware counters and data activity hardware counters as shown in Tab. 3, which we identified to be the major sources of power draw. An entire data set was collected for three different microarchitectures using five benchmark suites. In order to have the same basis for the comparison between the neural network and a regression model, we updated the regression model from Section 2.1 to take into account hardware performance counters that we sampled as shown in Tab. 3. We conducted experiments with different numbers of runs per each benchmark to test the impact of data set size on the neural network model's accuracy. As shown in Fig. 6 (a), we noted that if the same benchmarks are run multiple times, overall accuracy of the NN model increases. Figure 6 (b) shows that accuracy of the NN model increases as we increase the number of different benchmarks in the training set.

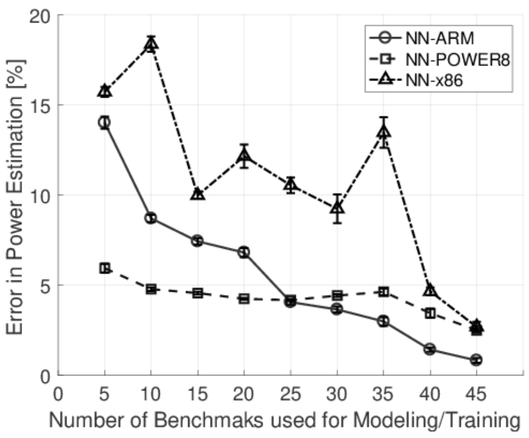
If we 5 times run each benchmark from each suite on Cavium ThunderX ARM implementation, the error in power consumption estimation is around 3%, while if each benchmark is ran 20 times, the error drops below 1%. We noticed no further increase in accuracy by increasing the number of runs beyond 20, which indicates a convergence threshold for the NN model. Neither accuracy improvement was observed by the regression model (REG) as well.

Accuracy improvement with an increased number of runs of the same benchmark occurs due to the fact that a benchmark's profile differs from run to run as illustrated in Fig. 7. For example, by using the Cavium ThunderX microarchitecture we obtain a different spread of values for **IFETCH** counter between different runs of PARSEC benchmarks. Such variance in data provides additional data points for NN model training, which improves its accuracy. As mentioned in the previous paragraph, we determined that the ideal number of runs for each benchmark from each suite is 20. It should be noted that data variance/noise has a negative impact on accuracy of the linear regression model.

After the training phase, the neural network is deployed to provide node power predictions. The information flow of the training phase and the deployment phase is shown in Fig. 5 as a solid blue line and a dotted red line, respectively. In a neural network, each connection between neighbouring layers has a weight to scale data and a bias that allows shifting the activation function. Data points from the input layer are inserted as inputs to the next consecutive layers (hidden layers). Then, the hidden layers sum the data fed to them, scale (weight) the data, and process it until the data reaches the last layer that outputs the predicted node power:



(a) The number of each benchmark runs



(b) The number of benchmarks

Figure 6. Power estimation accuracy as a function of benchmark runs and number of benchmarks

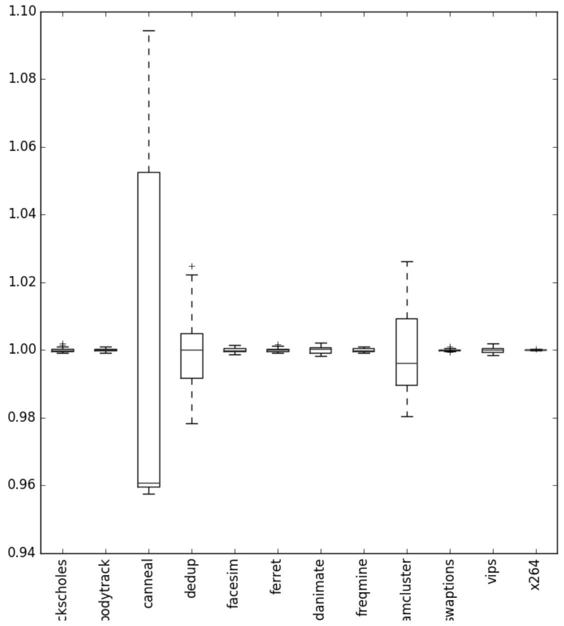
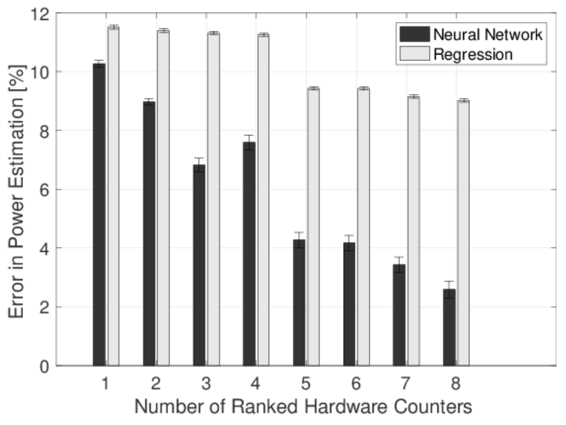
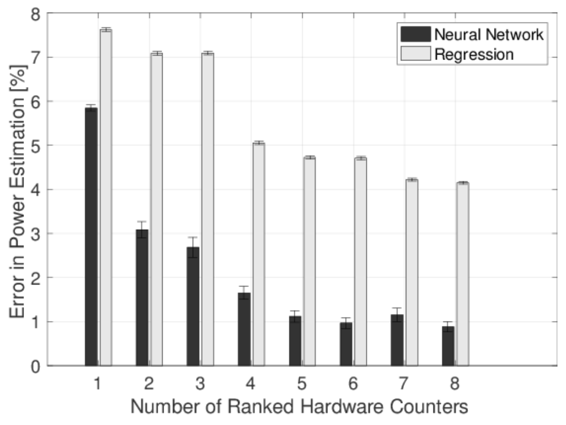


Figure 7. The variance of **IFETCH** hardware performance counter on Cavium ThunderX microarchitecture implementation



(a) Intel 64



(b) Cavium ThunderX

Figure 8. Error in power estimation between regression and neural network model for Intel 64 and Cavium ThunderX ARM microarchitecture implementations

$$w_{ij}(k+1) = w_{ij}(k) - \eta\frac{\delta e_k}{\delta w_{ij}}, \tag{3}$$

where η is the learning rate parameter which determines the rate of learning. w_{ij} represents the scalar value of weight on the connection from layer i to j. e_k represents the error of NN at kth iteration. $\delta e_k/\delta w_{ij}$ determines the weighted search direction for this iterative method.

The weights and biases of the network are updated only after the entire training set has been applied to the network. We used 10–fold cross-validation to validate our model where 90% of randomly selected data is used as a training set and 10% is used as a validation set. The gradients calculated for each training set are added together to determine the change in weights, and biases. The weights and biases are updated in the direction of the negative gradient of the performance function.

For our neural network model we have tried three commonly used back–propagation algorithms: *Levenberg–Marquardt* [21], *Scaled conjugate gradient* [20], *Resilient* [22], and *Bayesian Regularization*. The Bayesian Regularization algorithm performs the best in terms of accuracy (showing the minimum mean error) but has more computational overhead (showing higher training time). Scaled conjugate gradient and Resilient perform the best in terms of computational overhead, showing less time for training than the other algorithms. As a result of this analysis, we decided to use the well-balanced Levenberg–Marquardt back propagation algorithm for all our experiments. However, the Bayesian Regularization back–propagation can be used provided the model's accuracy is a constrain. Scaled conjugate gradient or resilient algorithms can be used provided the learning overhead is a constraint.

Figures 8 and 9 show results that we have obtained on three micro-architectures. For each microarchitecture, the NN model provides higher accuracy compared to the regression model. The largest improvement in accuracy was observed for Cavium ThunderX ARM architecture as it is the simplest architecture to model with an in–order processor and a simplified two–level cache hierarchy. Even though this microarchitecture is the simplest of the three studied, the linear regression model is less accurate than the neural network model. Note that both models are least accurate on Intel 64 microarchitecture, while they are the most accurate on IBM POWER8 microarchitecture. This is due to the fact that performance counters which we identified in Section 3.1 are better indicators of power consumption on IBM POWER8. As part of our further work, we are planning to research which counters are better suited for Intel 64 and

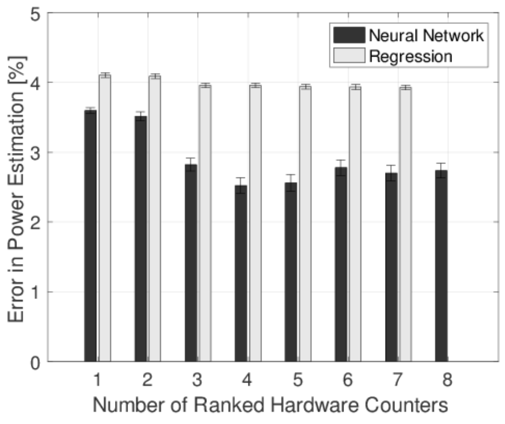


Figure 9. Error in power estimation between LR and NN models for POWER88 micro-architecture implementation

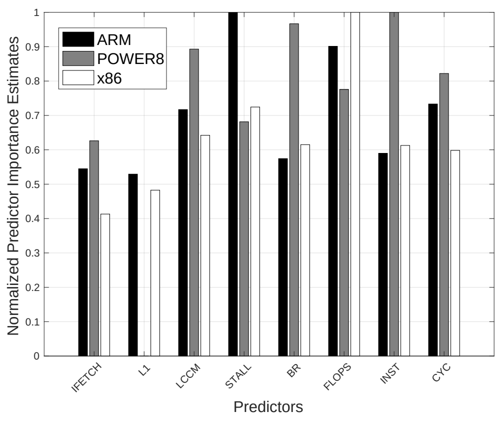


Figure 10. Importance of each hardware performance counters in power consumption estimation using the neural network model

Cavium ThunderX architectures. It is also important to note that significant improvements in accuracy for all three different microarchitectures stop around the fifth *most important* hardware counter. Figure 10 shows importance of each hardware performance counter from Section 3.1 for power consumption estimation.

Figure 10 suggests that it would be very difficult to use a neural network model developed for one architecture in order to estimate power consumption on the other architecture. Importance of different hardware counters for power consumption estimation is different for different microarchitectures. For example, Intel 64 and Cavium ThunderX microarchitectures are of similar importance to the same hardware performance counters except for **STALL** and **FLOPS**. **STALL** is more important for Cavium ThunderX micro-architecture because it is an in-order processor, and once there is a stall, instructions throughput and power consumption drop significantly. On the other hand, the floating point unit on Intel 64 is optimised for a very high throughput, and as a result, it consumes plenty of power on the floating-point heavy code. IBM POWER8 micro-architecture implementation behaves differently from Intel 64 and Cavium ThunderX. Since IBM POWER8 has been optimised for a high throughput, the counters that reflect the number of executed instructions and correctly predicted branches are the most important. Also, since the power cost of a cache that is missing and/or shared in a local is high, the access to remote memory (**LLC**) is very important when measuring power consumption. Note

Table 4. The error in estimating power on IBM POWER8 S822LC and Cavium ThunderX ARM when using the neural network model trained on Intel 64 microarchitecture implementation

Microarchitecture Implementation	IBM POWER8 S822LC	Cavium ThunderX ARM
Error in power estimation (%)	77%	64%

that due to a very well-developed pre-fetching mechanism on IBM POWER8 microarchitecture, local cache hits (**L1**) do not contribute to power consumption estimation on this architecture.

Nevertheless, we tried to apply a model trained on results from Intel 64 to estimate power of benchmarks' ran on Cavium ThunderX and IBM POWER8 microarchitecture implementations. The results are shown in Tab. 4. As expected, since Intel64 and Cavium ThunderX give similar importance to the same hardware performance counters, the power estimation on Cavium ThunderX microarchitecture implementation in Tab. 4 is more accurate then on IBM POWER8. Unfortunately, the accuracy is significantly worse when compared to Fig. 8 and 9. This is mainly due to the difference in importance of the same hardware performance counters that each microarchitecture implementation assigns. Furthermore, since the regression model is better off then the neural network model in both cases, there is space for improvement in the training phase of the neural network model in order to improve prediction accuracy. As part of the further work, we are planning to add results from Tab. 4 as part of training to the neural network model in order to make the neural network aware of different weights that each microarchitecture implementation assigns to the semantically same hardware performance counters.

Conclusions and Future Work

We demonstrated how to use hardware counters to develop models of power consumption for a broad range of HPC applications on three different microarchitecture implementations: Intel 64, IBM POWER8 and Cavium ThunderX ARMv8.

We demonstrated 2× to 3× better accuracy predictions using the NN model for power consumption compared to the LR model.

We provided a comparative analysis of the importance of different hardware counters for power consumption across three microarchitectures. We believe that results of our work can contribute to software/hardware co–design efforts towards developing unified, cross-architectural models to predict power consumption.

References

1. Acun, B., Lee, E.K., Park, Y., V. Kalé, L.: Neural Network-Based Task Scheduling with Preemptive Fan Control. In: International Workshop on Energy Efficient Supercomputing (E2SC). ACM (2016), DOI: 10.1109/E2SC.2016.016

2. Auweter, A., Bode, A., Brehm, M., Brochard, L., Hammer, N., Huber, H., Panda, R., Thomas, F., Wilde, T.: A Case Study of Energy Aware Scheduling on SuperMUC. In: Supercomputing – 29th International Conference, ISC 2014, Leipzig, Germany, June 22-26, 2014. Proceedings. pp. 394–409 (2014), DOI: 10.1007/978-3-319-07518-1_25

3. Bailey, D.H., Barszcz, E., Barton, J.T., Browning, D.S., Carter, R.L., Dagum, L., Fatoohi, R.A., Frederickson, P.O., Lasinski, T.A., Schreiber, R.S., Simon, H.D., Venkatakrishnan, V., Weeratunga, S.K.: The NAS Parallel Benchmarks - Summary and Preliminary Results. In: Proceedings of the 1991 ACM/IEEE Conference on Supercomputing. pp. 158–165. Supercomputing '91 (1991), DOI: 10.1145/125826.125925

4. Bansal, N., Lahiri, K., Raghunathan, A., Chakradhar, S.T.: Power Monitors: A Framework for System-Level Power Estimation Using Heterogeneous Power Models. In: VLSI Design. pp. 579–585. IEEE Computer Society (2005), DOI: 10.1109/icvd.2005.138

5. Bhat, S.G.: Openpower based Inband OCC sensors. `https://github.com/shilpasri/-inband_sensors` (2017), accessed: 2018-08-31

6. Bienia, C.: Benchmarking Modern Multiprocessors. Ph.D. thesis, Princeton University (2011)

7. Bircher, W.L., John, L.K.: Complete System Power Estimation Using Processor Performance Events. IEEE Trans. Comput. 61(4), 563–577 (2012), DOI: 10.1109/tc.2011.47

8. Borghesi, A., Bartolini, A., Lombardi, M., Milano, M., Benini, L.: Predictive Modeling for Job Power Consumption in HPC Systems. In: High Performance Computing – 31st International Conference, ISC High Performance 2016, Frankfurt, Germany, June 19-23, 2016, Proceedings. pp. 181–199 (2016), DOI: 10.1007/978-3-319-41321-1_10

9. Burtscher, M., Zecena, I., Zong, Z.: Measuring GPU Power with the K20 Built-in Sensor. In: Proceedings of Workshop on General Purpose Processing Using GPUs. pp. 28:28–28:36. GPGPU-7, ACM, New York, NY, USA (2014), DOI: 10.1145/2588768.2576783

10. Contreras, G., Martonosi, M.: Power Prediction for Intel XScale®Processors Using Performance Monitoring Unit Events. In: Proceedings of the 2005 International Symposium on Low Power Electronics and Design. pp. 221–226. ISLPED '05 (2005), DOI: 10.1109/lpe.2005.195518

11. Demuth, H., Beale, M.: Neural Network Toolbox for Use with MATLAB. `http://www.mathworks.com/help/nnet/` (2017), accessed: 2018-08-31

12. Duy, T.V.T., Sato, Y., Inoguchi, Y.: Performance evaluation of a green scheduling algorithm for energy savings in cloud computing. In: Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on. pp. 1–8. IEEE (2010), DOI: 10.1109/ipdpsw.2010.5470908

13. Elisseev, V., Baker, J., Morgan, N., Brochard, L., Hewitt, T.: Energy Aware Scheduling Study on BlueWonder. In: 4th International Workshop on Energy Efficient Supercomputing, E2SC@SC 2016, Salt Lake City, UT, USA, November 14, 2016. pp. 61–68 (2016), DOI: 10.1109/e2sc.2016.014

14. Eranian, S.: Perfmon2: a flexible performance monitoring interface for Linux. In: Proceedings of the Ottawa Linux Symposium. pp. 269–288 (2006), `http://perfmon2.sourceforge.net/ols2006-perfmon2.pdf`

15. Heroux, M.A., Doerfler, D.W., Crozier, P.S., Willenbring, J.M., Edwards, H.C., Williams, A., Rajan, M., Keiter, E.R., Thornquist, H.K., Numrich, R.W.: Improving Performance via Mini-applications. Tech. Rep. SAND2009-5574, Sandia National Laboratories (2009), DOI: 10.2172/993908

16. Heroux, Michael A and Neely, Rob, and Swaminarayan, Sriram: ASC Co-Design Proxy App Strategy. Tech. Rep. LLNL-TR-592878, Los Alamos National Laboratory (2013), DOI: 10.2172/1055856

17. Huang, W., Lefurgy, C., Kuk, W., Buyuktosunoglu, A., Floyd, M., Rajamani, K., Allen-Ware, M., Brock, B.: Accurate Fine-Grained Processor Power Proxies. In: Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture. pp. 224–234. IEEE (2012), DOI: 10.1109/micro.2012.29

18. Li, T., John, L.K.: Run-time Modeling and Estimation of Operating System Power Consumption. In: Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. pp. 160–171. SIGMETRICS '03 (2003), DOI: 10.1145/781047.781048

19. McCalpin, J.D.: Memory Bandwidth and Machine Balance in Current High Performance Computers. IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter pp. 19–25 (1995)

20. Moller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. NEURAL NETWORKS 6(4), 525–533 (1993), DOI: 10.1016/s0893-6080(05)80056-5

21. More, J.J.: The Levenberg–Marquardt Algorithm: Implementation and Theory. Numerical Analysis, ed. G. A. Watson, Lecture Notes in Mathematics 630, Springer Verlag pp. 105–116 (1977), DOI: 10.1007/bfb0067700

22. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In: Neural Networks, 1993., IEEE International Conference on. pp. 586–591 (1993), DOI: 10.1109/icnn.1993.298623

23. Rodrigues, R., Annamalai, A., Koren, I., Kundu, S.: A Study on the Use of Performance Counters to Estimate Power in Microprocessors. IEEE Trans. on Circuits and Systems 60-II(12), 882–886 (2013), DOI: 10.1109/tcsii.2013.2285966

24. Intel: OpenMPI. `https://www.open-mpi.org/` (2017), accessed: 2018-08-31

25. Texas Instruments: System Power Management and Protection IC With PMBus^{TM}. Texas Instruments (2013)

26. The Green500: The Green Lists. `https://www.top500.org/green500/lists/2017/11` (2017), accessed: 2018-08-31

27. Tiwari, A., Laurenzano, M.A., Carrington, L., Snavely, A.: Modeling power and energy usage of HPC kernels. In: Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International. pp. 990–998. IEEE (2012), DOI: 10.1109/ipdpsw.2012.121

28. Woo, S.C., et al.: The SPLASH-2 Programs: Characterization and Methodological Considerations. In: Proceedings of the 22nd Annual International Symposium on Computer Architecture. pp. 24–36. ISCA '95, ACM, New York, NY, USA (1995), DOI: 10.1109/isca.1995.524546

DOI: 10.14529/jsfi180404

Autotuning Techniques for Performance-Portable Point Set Registration in 3D

***Piotr Luszczek*[1], *Jakub Kurzak*[1], *Ichitaro Yamazaki*[1], *David Keffer*[2], *Vasileios Maroulas*[3], *Jack Dongarra*[1,4,5]**

We present an autotuning approach applied to exhaustive performance engineering of the EM-ICP algorithm for the point set registration problem with a known reference. We were able to achieve progressively higher performance levels through a variety of code transformations and an automated procedure of generating a large number of implementation variants. Furthermore, we managed to exploit code patterns that are not common when only attempting manual optimization but which yielded in our tests better performance for the chosen registration algorithm. Finally, we also show how we maintained high levels of the performance rate in a portable fashion across a wide range of hardware platforms including multicore, manycore coprocessors, and accelerators. Each of these hardware classes is much different from the others and, consequently, cannot reliably be mastered by a single developer in a short time required to deliver a close-to-optimal implementation. We assert in our concluding remarks that our methodology as well as the presented tools provide a valid automation system for software optimization tasks on modern HPC hardware.

Keywords: portable performance engineering, point set registration, autotuning with code generation, combinatorial optimization.

Introduction

Many aspects of computer vision commonly uses algorithms for registration of point sets in three-dimensions (3D). But in many areas of science, these methods can be used for analyzing 3D data arriving from experimental hardware instruments. In particular, this is necessary in order to produce unambiguous descriptions of atomic-scale structures from large data sets originating in Atomic Probe Tomography (APT) [15, 19] and multimodal electron microscopy (EM) [14, 24]. In the study and design of High Entropy Alloys (HEAs), APT can generate data sets that include from 10^6 to 10^7 atoms in a single frame. Work is underway for electron microscopes to relay time-resolved frames, resulting in an explosion of data that truly puts the analysis of the output of these analytical techniques of registration squarely within the realm of "big data" analytics. The ultimate goal, when processing APT data sets, is to be able to resolve both atomic identity and atom position. The incoming instrument data is in the form of sets of atomic coordinates (x, y, z) in three-dimensional (3D) space accompanied by identification of the atom type. Such data is in many ways similar, in its basic form, to visualization tasks but the registration of the points will be followed by derivation of physics, chemistry, or material science profiles that inform the scientists of emergent properties of the analyzed samples. This serves as a motivation for fast and accurate implementations of the registration algorithms that are the subject of this paper.

In order to arrive at novel properties such as resistance to high-temperature, corrosion, fracture and fatigue [8, 26], large amount of work has been performed to evaluate various techniques for characterizing local atomic environments [1]. To a significant extent, we follow here a similar approach and adopt tools from image reconstruction and pattern analysis in the

[1]Innovative Computing Laboratory, University of Tennessee, Knoxville TN, US
[2]Department of Materials Science & Engineering, University of Tennessee, Knoxville TN, USA
[3]Department of Mathematics, University of Tennessee, Knoxville TN, USA
[4]Oak Ridge National Laboratory, Oak Ridge TN, USA
[5]School of Computer Science and School of Mathematics, The University of Manchester, Manchester, UK

generic field of machine vision in order to construct detailed atomic structures with highly-resolved locations from often defective data sets, especially those coming from APT experiments. Mathematically, of most importance is the minimization of the Frobenius norm: $\|\cdot\|_F$; computed for a set of matrices representing the difference between a model reference configuration, $\mathbf{m}$, denoting the true, average local structure, and the local configuration (data), $\mathbf{d}_i$, around atom i, where for an APT experiment, i ranges from 1 to $k \approx 10^7$:

$$\min_{R_i, P_i} \sum_{i=1}^{k} \|\mathbf{m} - P_i \mathbf{d}_i R_i\|_F^2 \,, \tag{1}$$

where each configuration has a unique permutation, P_i, and rotation, R_i, matrix (both real and orthogonal) in order to make it invariant to the arbitrary orientation and numbering generated by the experimental process.

A formulation of the problem that is based on Bayesian principles is under development and it explores Markov chain Monte Carlo scheme. A more classic approach is taken here to espouse benefits of autotuning whereby automation is applied to efficient evaluation of a large variety of implementation variants. In particular, we use Expectation-Maximization (EM) Iterative Closest Point (ICP), or EM-ICP for short. EM-ICP is a stochastic method for registration of surfaces. It improves issues found in other algorithms related to minimizing non-convex cost function. Other registration algorithms are given in Section 1.

In a simplified yet generic mathematical form, registration of point sets $X = \{x_i | x_i \in \mathbb{R}^d\}$ and $Y = \{y_i | y_i \in \mathbb{R}^d\}$ may be expressed as:

$$\min_{f \in \mathcal{T}} \|f(X) - Y\|, \tag{2}$$

where the point sets come from a 3D space (rather then a space of arbitrary dimension d):

$$X = \{x_1, x_2, \ldots, x_\ell\}, Y = \{y_1, y_2, \ldots, y_m\} \text{ with } x_i, y_j \in \mathbb{R}^3 \tag{3}$$

and the function f, drawn from the eligible set of transformations $\mathcal{T}$, is in our case a simple linear transform chosen to represent a combination of rotation, scaling, and translation:

$$f : X \mapsto R \times X + t \equiv Y \tag{4}$$

these restrictions result in a transformation that is called *rigid registration* and is the primary focus of this manuscript. However, a more general *non-rigid registration* is possible and it allows the transformation to be affine. For such a case, it is possible to include anisotropic scaling and skews. Furthermore, a generalization is possible which allows as input an unknown point set. This is close to the APT data sets which by their very nature cannot account for all atoms in the sample. Note also the noise resilience whereby the original formulation is often assumed to be robust to input measurement errors in the sense that it can handle a distribution of errors imposed on input data with either outliers or some points missing or both. Again, this caters directly towards APT-generated point clouds.

A much-simplified overview of the EM-ICP algorithm is depicted in Fig. 1. Starting with the initial guesses for the components of the transformation as an identity rotation and zero scaling matrix $R^{(0)}$ and zero translation matrix $t^{(0)}$ which consequently get updated in an iterative fashion by minimizing Mean Square Error (MSE).

- **Step 1**: Start with the initial guess of identity rotation and zero translation:

$$R^{(0)} \leftarrow I,$$

$$t^{(0)} \leftarrow 0$$

- **Step 2**: Iterate until convergence:
  - **Step 2.1**: Find closest point

$$\vec{y}_{i*} \in Y \quad \text{for each} \quad \vec{x}_i \in X$$

  in:

$$i* = \arg\min_{j=1,\dots n_y} \left\| \vec{x}_i - R^{(k-1)}\vec{y}_j + \vec{t}^{(k-1)} \right\|$$

  - **Step 2.2**: Build the ordered correspondence

$$Y^* = \{y_{1*}, \dots, y_{n_x*}\}$$

  - **Step 2.3**: Find rigid transform such that

$$\min_{\mathrm{MSE}(X,Y^*)} \{R^*, \vec{t}^*\}$$

  to minimize overall error
  - **Step 2.4**: Update the current transformation with the newly found one:

$$R^k \leftarrow R^*, \qquad \vec{t}^k \leftarrow \vec{t}^*$$

Figure 1. Outline of the algorithmic steps of EM-ICP (MSE = mean square error)

The rest of the article is organized as follows. Section 1 is devoted to a short survey of the related work. In Section 2, we present performance profiles of the implementations of the registration algorithm under consideration. Section 3 contains detailed performance results on various hardware platforms. Section 4 discusses in more detail some the important aspects of the results. The conclusion section summarizes the study and points to potential directions for further work.

1. Related Work

The algorithm called Iterative Closest Point (ICP) [2, 27] has two important properties: simple implementation structure and the low levels of computational cost. This characterization on both counts has resulted over time in increased popularity and both of these aspects have contributed to proliferation of numerous variants [7, 22] including the one we consider here in more detail called EM-ICP [10]. The Expectation Maximization (EM) algorithm for Gaussian Mixture Model (GMM) may be shown [3] to be equivalent to Robust Point Matching (RPM) algorithm [9] that alternates soft-assignment of correspondences and transformation. Multiple versions of RPM have been developed [4, 5, 21]. Finally, there is Coherent Point Drift (CPD) algorithm [20] which performs the so called non-rigid registration while using a suitable regularizer.

Implementations of these methods is the primary focus of the following sections. Reference codes are available in multiple forms but most often they may be characterized as sequential

Table 1. Performance profile of EM-ICP implementation in CUDA running on NVIDIA Kepler accelerator

Kernel symbolic name	Percentage of time spent
normalize	49%
update	31%
SGEMV	16%
SGEMM	3.0%
SDOT	0.5%

implementations. Parallel counterparts based on OpenMP or explicit multithreading for multicore processors are rare. Even more so are enhancements for hardware accelerators and we attempt to breach that gap. For example, some support for ICP is available, in the Point Cloud Library (PCL) [23] as part of a larger set of methods. Some of these codes may be used as the basis for our implementations but they require changes for efficiency. Additionally, we perform updates to make them adhere to the modern software stack [18]. More specifically, we needed to introduce the current version of CUDA which is better than what is available in the code with CUDA version 5 being the last supported release. Additionally, by adding utilization of the GPU hardware platforms, we expanded test results. Some implementations[6] could potentially be adapted with cosmetic adjustments to allow them to use more recent versions of CUDA such as version 6.5 but this is not a guarantee of good performance. In summary, the implementations we generate are highly customized and target the most recent versions available and supported by NVIDIA: 7.5 and 8.0 with initial work towards compatibility with even newer releases of CUDA 9.0 and 10.0. Unfortunately, these latest versions of CUDA were not widely available to the public during the initial test runs.

2. Analysis of Bottlenecks based on Code Profiling

Our survey of existing codes for EM-ICP revealed that the freely available implementations[7] focus on visualization tasks and image processing workflows that are often optional in case of scientific instruments. Our focus is to provide very high ingest rates of the data coming from the hardware sensors. Our goal is to automatically derive an implementation that is able to process them in time before the long-term storage system becomes overwhelmed. Based on a survey of available solutions, we concluded that their support for High Performance Computing (HPC) techniques is poor and we faced the choice of retrofitting the codes for multithreading, modern accelerator libraries, and performance profiling or write our own version from scratch. We chose the former and used this updated code as a reference point that we optimized on a variety of modern HPC platforms.

As the first step in the performance engineering, we acquired an application profile and identified the bottleneck portions of the code that may then be targeted with our autotuning methodology in order to maximize the potential speedup benefits. Table 1 shows a typical performance profile on one of the tested devices. It is representative of the time breakdown that

[6]One such implementation is ICPCUDA available at `https://github.com/mp3guy/ICPCUDA`.

[7]We did not consider commercial implementations for this study but only the codes that can be obtained under open source or educational license.

Table 2. Autotuning parameter space used for optimization of performance through OpenMP parallelization and directives

Parameter	Possible values
Compiler:	GNU `gcc`, Intel `icc`
Threading runtime:	GNU `gomp`, Intel `iomp`
Number of threads:	1, . . . , 40, . . . , 70, . . .
Speculation code and hinting:	yes, no
Hyper-threading:	yes, no
	enabled with OpenMP or kernel affinity of threads
Thread affinity:	compact, spread, round-robin
Main memory page mapping:	round-robin, application-specific (first touch, randomized, . . .)
Software-exposed parallelism:	collapse(2), collapse(3), . . .
Thread scheduling selection:	static, chunk-based, dynamic, . . .
Parallel grain selection (chunk size):	1, 2, 3, . . .
Vectorization length:	2, 4, 8, 16

we observed on the other machines used in tests. Note that this represents less computationally-intensive variant of the ICP method than has been reported by approaches that used Singular Value Decomposition (SVD) at each iteration [6]. The codes for point set registration require 32-bit floating-point arithmetic for computation and hence most of the code uses single-precision float data types and the profile from the figure uses that precision and changes to this will be explicitly noted. Also, the experimental hardware such as APT generates data that can be accurately represented in 32-bit floating-point digits.

The common technique for efficient implementations is to off-load the compute-intensive parts of the code to highly tuned numerical libraries. In the case of results from Tab. 1, the calls to NVIDIA CUBLAS make optimal use of the compute units (SGEMM) and the available memory bandwidth (SGEMV and SDOT). Unfortunately, once these sections of the code achieve the state close to hardware-optimal performance, the other parts of the code become the main sources of slowdown. These two slowdown parts are named update and normalize. In terms of the operations from algorithm in Fig. 1, they represent updates and normalization of the correspondence and error metrics throughout the iterations. Focusing on these two portions exclusively targets nearly 90% of the execution time across our tested hardware.

Once we identified the code sections that are important for performance engineering, we proceeded with defining the parameter space of available configurations that need to be explored for generating code that would execute efficiently. Table 2 shows a summary of that space for multicore hardware (GPU-specific considerations are presented below) with open source[8] and commercially developed[9] software stack. The search space for autotuning optimization is multidimensional and heterogeneous in the sense that it includes different categories of parameters, namely: binary

[8]Due to the fact that the GNU and LLVM compilers share the OpenMP runtime, we only considered GNU `gomp` but a LLVM-specific solution is under development.

[9]We only considered one commercial compiler but other choices are also possible, for example the PGI Group or Microsoft Visual Studio compilers that support a version of the OpenMP standard.

```
compiler = ("gcc", "icc")
runtime = ("gomp", "iomp")
omp_num_threads = range(1, 75)
speculation = (False, True)
hyper_threading = (False, True)
kmp_affinity = ("disabled", "compact", "scatter")
gomp_cpu_affinity = ("1-75:1", "1-75:2", "1-75:3", "1-75:4")
omp_palces = ("none","threads","cores", "sockets")
omp_proc_bind = ("close", "master", "spread","none")
memory_affinity = ("round robin", "random", "first touch")
collapse = (1, 2, 3)
schedule = ("default", "static", "chunk", "dynamic")
schedule_size = range(0, 64+1)
simd_length = (2, 4, 8, 16, 32)
```

Figure 2. Definition of the autotuning space for multicore and manycore CPUs

values (for example use, hyperthreading or not use it), ordinal/categorical/enumeration values (for example, compiler-version pairs: `gcc` 6.0, `gcc` 7.0, `icc` 2016, `icc` 2017), integer values (for example, an integer range of loop blocking parameter values), and continuous variables (for example, cache reuse ratio). Theoretically, the entire search space may be represented by a Cartesian product of these variables which would result in combinatorial explosion in the size of the search space. In practice, the search space is much less regular due to a number of software and hardware constraints. Some examples include:

- The problem to be optimized might be solved for the GNU compiler but is an issue for the Intel compiler due to unknown internal issues.
- The output results might be changing depending on the version of the compiler: often a newer version produces binaries whose results are an improvement but sometimes performance regressions may also occur.
- The interactions that are internal to the compiler, its OpenMP runtime implementation, OS version, and finally the processor firmware could further complicate the produce optimization results.

We have developed Domain Specific Language (DSL) to assist the autotuning processes. It is called LANAI (LANguage for Autotuning Infrastructure) [16]. It deals primarily with the definition of search space for optimal implementation. While further details certainly go beyond the current scope, for completeness, we feel compelled to include some information how the symbolic description of the parametric space from Tab. 2 translates into a LANAI description that we present in Fig. 2. Due to the way we presented the search space constraints in the table, the code from the figure may be considered self-explanatory. However, the full complexity of LANAI is beyond the scope of this manuscript and includes parsing of the autotuning specification (ranges, constraints, and their mutual dependence), multiple stages of optimization that reduce the autotuning time, and code generation that allows search space exploration to take place at the speed of a native code without the need to ever write such a code by hand.

2.1. Optimizations specific to NVIDIA GPUs and CUDA Software Stack

So far, we discussed the autotuning optimization space with the focus on multicore hardware that may be, in a generic form, characterized by multipurpose compute cores with a deep memory hierarchy of caches and complex main memory structures. Hardware accelerators such as compute-oriented GPUs differ from multicore hardware in a number of important ways including higher number of floating-point units, higher bandwidth to/from the main memory, and higher latency

Table 3. Optimization space for a single kernel based on CUDA parallelization features

Parameter description	Possible values
Read coalescing for input points X	Yes / No
Write coalescing for output points Y	Yes / No
Thread grid shape $\mathbb{G}^{i \times j}$	$i, j \in \{1, \ldots, 1024\}$
Thread block shape $\mathbb{T}^{R \times C}$	$R, C \in \{1, \ldots, 1024\}$
Amount of Shared Memory used	$\{0, 2^{10}, 2^{11}, \ldots, 2^{16}\}$
SM or SMX streaming unit utilization (number of CUDA streams)	$\{1, 2, \ldots, 80\}$
Active threads per thread	$\{8, 16, 32\}$
Data affinity for grid-block: $\mathbb{G}^{i \times j} \times \mathbb{T}^{R \times C} \to \mathbb{R}^{X \times Y}$	row/column-wise, mixed
Floating-point precision	16-bit, 32-bit, 64-bit
Verification criteria	**Allowed values**
GPU occupancy[‡]	$30\% \div 95\%$

[‡] NVIDIA provides occupancy calculator for users. Good performance is not necessarily equivalent to good occupancy but there exist occupancy thresholds that are almost always well correlated with achieving sufficiently high levels of performance based on a relevant metric (bandwidth or compute intensity).

```
coalesce_input = (True, False)
coalesce_output = (True, False)
grid_x = range(1, 1024+1)
grid_y = range(1, 1024+1)
block_x = range(1, 1024+1)
block_y = range(1, 1024+1)
shared_memory = range(0, 2**16+1, 2**10)
BLK = (8, 16, 32)
precision = (16, 32, 64)
```

Figure 3. Definition of the autotuning space for GPUs

to the main memory. The memory hierarchy of GPUs is shallower and often features smaller caches that are shared on a per-device basis. Despite these differences, the breakdown of execution time from Tab. 1 is generally applicable to both types of compute platforms. Furthermore, this similarity applies across multiple GPU devices and compiler tool chains which might feature, for example, different approaches to instruction predication and branch removal algorithms. Such details depend on availability of Special Function Units (SFUs) on the target device and the hardware's predication window length. As a practical example, consider the difference between NVIDIA's Kepler and Maxwell architectures: the former features a high end compute cards and has full support for 64-bit precision floating point arithmetic while the latter only targets gaming and rending markets with 32-fold slowdown of 64-bit instructions.

The profiling on the NVIDIA hardware is done through either the `nvprof` command line tool or the `nvvp` GUI application. They are assisted by the hardware counters for minimal overhead on the running code. They were used to gather the profile and bottleneck information from the reference code.

To maximize the bandwidth achieved by our implementation, we aim at efficient use of global and shared memory banks as well as enforcement of coalesced reads through stride-1 accesses. This is done explicitly since the GPU compilers often do not automatically handle Instruction-Level Parallelism (ILP). Also by design, CUDA shifts the burden of exposing the Thread-Level Parallelism (TLP) to the user as the threads must be explicitly created and managed by the user code inside kernel functions. These considerations result in a modified search space for the GPU-optimized autotuning that is shown in Tab. 3.

As was the case for CPUs, Tab. 3 may be translated almost directly into LANAI specification shown in Fig. 3. There is a number of important issues worth noting. Because the grid and block dimensions are specified independently, for some combinations of dimensions the thread counts will exceed what's allowed on the GPU hardware. This may be counteracted with the LANAI's so called *constraint* or it may be left to the CUDA compiler and runtime: one of them will indicate insufficient resources for launching a kernel. Clearly, the former solution is preferred because it limits the overall size of the search space and consequently speeds up the autotuning process. On the other hand, *GPU occupancy* is much more vague criteria to meet as it needs to be calculated in a more complicated way and, additionally, it never had a very well-defined relation with achieved performance across a wide range of codes [25]. Nevertheless, it can still be used in the optimization process to either prune the search space or terminate the performance tests early when they fail to meet a predetermined occupancy requirement.

2.2. Using Limited Precision Arithmetic

2.2.1. Hardware Landscape for 16-bit Floating-Point

A new type of hardware extension for 16-bit floating-point precision arithmetic (FP16) has become much more main stream in the past few years. Initial experiments in deep network training with limited floating-point accuracy [11, 12] validated this for machine learning methods. Since then, a numerous hardware vendors and supercomputing sites have been involved with the trend that links computational Artificial Intelligence and HPC. Consider the announced AMD GPUs MI5, MI8, MI25 whose model number corresponds to the peak performance of the card in FP16: 5 Tflop/s, 8 Tflop/s, and 25 Tflop/s, respectively. Softbank's subsidiary ARM, announced publicly an extension to its NEON Vector Floating Point (VFP) to include FP16 in the V8.2-A architecture specification. NVIDIA GPUs that feature FP16 are widely available starting with Tegra TX1 and Pascal P100 cards. NVIDIA's Volta-based V100 and DG100 and the Xavier car platform feature further extensions of the FP16 support. TSUBAME 3 is one of the first supercomputers to prominently feature FP16 but other sites with NVIDIA Pascal hardware can fully utilize this new functionality. This is in line with our experiments in using FP16 in HPC benchmarking [17]. The ISO C++ standard is slated to add short float primitive data type that will likely be also added the C standard document following the IEEE 754-2018 specification and current support of _Float16 primitive data type.

2.2.2. FP16 Considerations for Point Set Registration

FP16 precision is officially defined by the IEEE 754-2008 standard. Its features are compared with the other floating-point precisions in Tab. 4. Note that FP16 was not meant as a format for compute but as a storage-only representation. This was reflected, among others, by the choice of the execution semantics of the Tensor Core unit in NVIDIA's Volta architecture that internally

Table 4. FP16 and Its Hardware Support IEEE 754 (2008)

Precision	Width	Exponent	Mantissa	Epsilon	Max
Quadruple	128	15	112	$\mathcal{O}(10^{-34})$	$\mathcal{O}(10^{4932})$
Extended	80	15	64	$\mathcal{O}(10^{-19})$	$\mathcal{O}(10^{308})$
Double	64	11	52	$\mathcal{O}(10^{-16})$	$\mathcal{O}(10^{308})$
Single	32	8	23	$\mathcal{O}(10^{-7})$	$\mathcal{O}(10^{38})$
Half†	16	5	10	$\mathcal{O}(10^{-3})$	65504

† defined only for storage

Table 5. Summary of hardware platforms used in the performance tests

Architecture	Manufacturer	Model	Name
Xeon x86	Intel	2620	Haswell
Xeon Phi	Intel	7280	Knights Landing
Tesla K40c	NVIDIA	GK110B	Kepler
Tesla P100	NVIDIA	GP100	Pascal

compute in FP32 arithmetic but consume FP16 operands. This is similar to the common practice of the way the Fused-Multiply-Add (FMA) instruction is implemented with higher precision for the intermediate results.

From the perspective of point set registration, FP16 has a potential benefit of increased bandwidth and vastly improved compute intensity. The former affords a 2-fold increase on the NVIDIA Pascal cards. The primary consideration is the limited range of the FP16 values as shown in Tab. 4. At the basic algorithmic level, the use of limited precision in most calculations may serve as a opportunistic regularization scheme which, among other things, could prevent overfitting the model to noisy input data as is the case in some scenarios.

3. Performance Results

3.1. Description of the Tested Hardware Platforms

Our autotuning experiments were performed on a number of hardware platforms including multicore, manycore, and GPU accelerators. A quick summary of these machines is given in Tab. 5. Out of the computers shown in the table, Intel Phi KNL could potentially be the newest and might be the least familiar to the reader. As an alternative to the superscalar x86, it represents a new architecture[10] from Intel that is solely based on low-power Atom Silvermont cores with the maximum of 72 cores in a chip connected with a mesh interconnect and divided logically into 4 so called quadrants with NUMA-like characteristics as far as data affinity is concerned. However, further details on cache and memory hierarchy with on-chip specification is out of scope of this article. Initially planned for two major hardware versions as either self-boot (self-hosted) or leveraged-boot (accelerator) units. Currently, only self-boot units are commercially available.

[10]Note that for the most part Xeon and Xeon Phi processors are binary compatible with the exception of the extra AVX512 instructions which made x86 debut in Intel Xeon Skylake Silver, Gold, and Platinum.

Table 6. Detail characteristics of the Intel Xeon Phi Knights Landing platform a manycore CPU used in the tests with the performance numbers coming from hardware specification or the vendor testing

Specification or Metric name	Peak or measured value
Core count	68
Hardware thread count	272
Vector FPUs length	512 bits
Main memory RAM	DDR4
Max DDR4 RAM	384 GiB
DDR4 latency	≈140 ns
Fast RAM	16 GiB MCDRAM
Max MCDRAM	16 GiB
MCDRAM latency	≈170 ns
MCDRAM configuration modes	flat, cached, mixed
Level 3 cache	0
Peak FP32	6093 Gflop/s†
SGEMM	4065 Gflop/s
Peak FP64	3046 Gflop/s‡
DGEMM	2070 Gflop/s
LINPACK Benchmark	2000 Gflop/s
STREAM MCDRAM	490 GB/s
STREAM DDR4	90 GB/s

† 68×1.4 GHz $\times$ 2 VPUs $\times$ FMA $\times$ 16 AVX lanes
‡ 68×1.4 GHz $\times$ 2 VPUs $\times$ FMA $\times$ 8 AVX lanes

The relevant details on the Intel Xeon Knights Landing (KNL) machine are given in Tab. 6 as they are available from public sources.

3.2. Application-Specific Performance Metric for Cross-Platform Comparisons: Derivation of the Performance Rate

We would like first prepare to for a more exhaustive comparison across both multiple hardware platforms *and* a wide range of input data sizes. We recognize the importance of using absolute timing for measuring performance. Here it would quickly become problematic if used in isolation. *Time-to-solution* is, by many accounts, a widely used metric relevant to the end user but we would also like to be able to attempt comparisons in a much more applicable fashion when there are many parameters that are not constant across the tested hardware and the input data sets. At the same time, we want our new metric to be close or even equivalent to the time-to-solution measurement as long as the data size remains constant. The common performance metric commonly used in HPC codes is Gflop/s which has important advantage that it is one metric to use across all input data sets and even applications. However, among many downsides is the fact that Gflop/s rating assumes that there is uniform (preferably linear) relationship between Gflop/s and the essential application speed (commonly time-to-solution). This particular downside results in multiple negative consequences such as artificial efforts to maximize the number of floating-point instructions that often come free as long as they are executed on data residing in Level 1 cache. Unfortunately, such operations contribute very little to the applications' ultimate goal that need faster execution for large memory footprints that do not fit in any cache level all at once. Instead, we derive an application-relevant metric based on the asymptotic performance theory [13].

For registration problems in particular, the performance metric we choose as relevant is the number of points in the cloud registered per second. Using this provides with a rate and we can quickly compare either images, scenes or surfaces that contain different number of points. Simultaneously, we are still able to map the rating back to time-to-solution when we know the exact point-count. We use Giga-points-per-second (Gpts) as the unit for the execution rate that is defined as:

$$r = \frac{t}{n_X n_Y} 10^{-9} \text{ Gpts},$$

where n_X and n_Y represent the number of the input and output points, respectively. The scaling factor of 10^{-9} is a standard SI prefix that allows all of the rates measured in this article to fall within the range of small numbers that are less than 50 that are familiar and human readable. The charts in the right of Fig. 4, Fig. 5, and Fig. 6 show this new metric applied to the timing charts shown in the left of the respective figures. As an added advantage, we are able to see subtleties of the optimal configurations. It turns out that in Fig. 5 the asymptotic performance rate is nearly 6 Gpts but the cache effects may be observe all the way until point clouds of size 5000 points when the working set fits in highest levels of the cache hierarchy and the performance rate is higher due to the much higher cache bandwidth and lower access latency of fetch image data.

Finally, rather than presenting a speedup, we only show all the performance results of all tested configurations without specifying the reference and most optimal ones. This enables the reader to see the range of results that may be observed in practice. Due to the automated way of obtaining the results, we can guarantee minimal human intervention while obtaining a fast implementation.

3.3. Timing and Performance Rate Results for GNU and Intel Compilers on the x86 Machine

In the first set of results, which are coming from the Intel x86 Haswell machine, we present ability of the GNU and Intel compilers to generate optimized code and it is shown in Fig. 4 and Fig. 5, respectively. To limit the total number of points present in the chart, we only included the most representative configurations and others are not shown. Specifically, the orig-no-omp configuration is treated as the reference measurement that may be considered sequential because the OpenMP parallelization was disabled for that run. Inclusion of this configuration in the charts allows us to show how the GCC compiler struggles to parallelize the registration loop nest as almost all other parallel configurations run generally at the same speed except for only one that is marked opt-omp-noif-collapse. Finally, that last configuration may be seen as clearly outperforming all the others because it enables far greater levels of parallelism and enables optimization as well better cross-thread work assignment for the GNU compiler. Regrettably, none of the optimizations tested allowed the GNU compiler to dip below 1 second running time for the largest point cloud size (40000 points). The Intel compiler had no such problem and ran under one second for quite a few optimization configurations. Regarding the weak results from

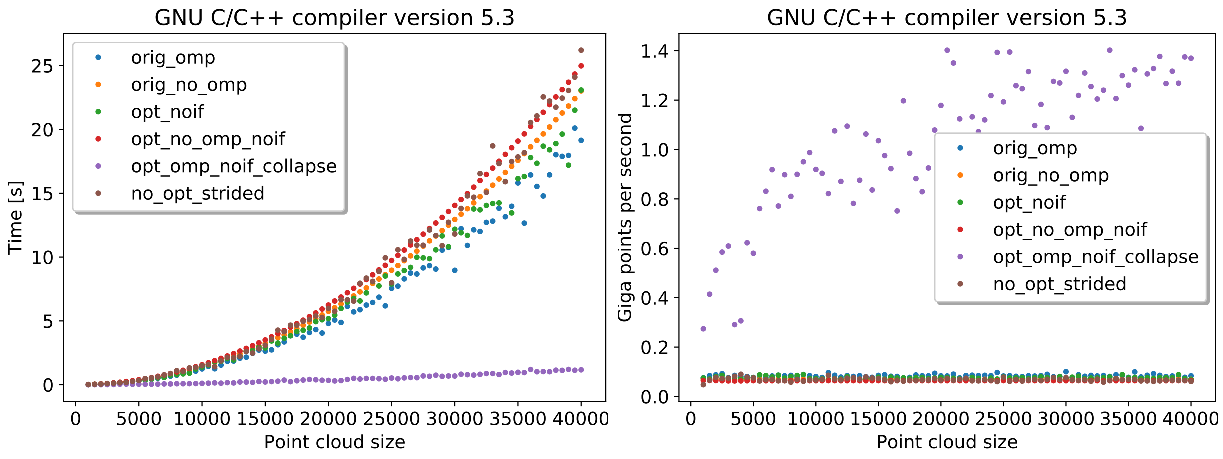


Figure 4. Point set registration timing and performance results on the x86 Haswell machine with the GNU compiler for various OpenMP configurations and the colors on left and right charts represent the same configurations

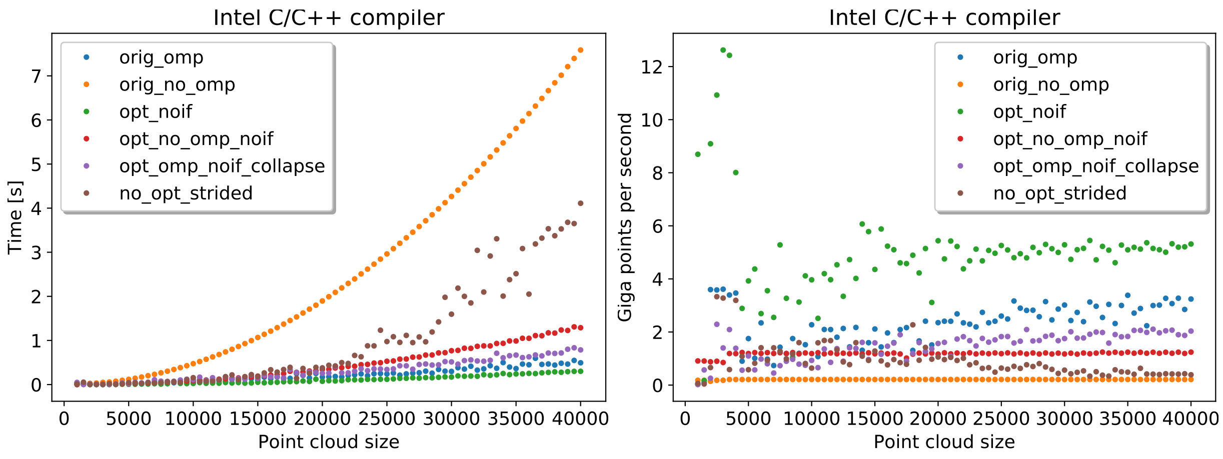


Figure 5. Point set registration timing and performance results on the x86 Haswell machine with the Intel compiler for various OpenMP configurations and the colors on left and right charts represent the same configurations

the GNU compiler, we did not conduct any further experiments to understand whether this may be related to either the generated instruction mix or more efficient OpenMP runtime. We leave this and other questions of such sort for future work based on similar hardware and software stacks. Also, we will no longer present GNU compiler results due to low observed performance. A detailed analysis of the instruction stream generated by GNU and Intel compilers is beyond the scope of this manuscript and we only attempted to use for both compilers similar flags including highest possible optimization level and relaxed floating point model to make aggressive code generation possible.

3.4. Timing and Performance Rate Results on Many-core KNL System and GPU Device Cards

Figure 6 presents the timing and performance rate results on the Intel Xeon Phi KNL machine. Two clear differences may be observed when comparing against x86 results:

- the cache effects are mostly not present for the point clouds with small data sizes as compared to all other x86 runs (originating either from GNU or Intel), also

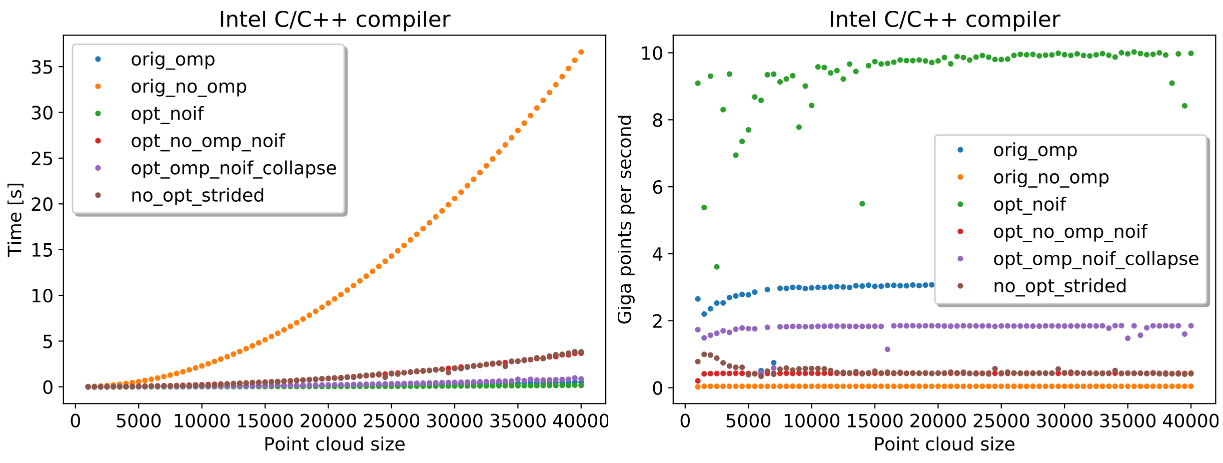


Figure 6. Point set registration timing and performance results on the Xeon Phi Knights Landing machine with the Intel compiler for various OpenMP configurations and the colors on left and right charts represent the same configurations

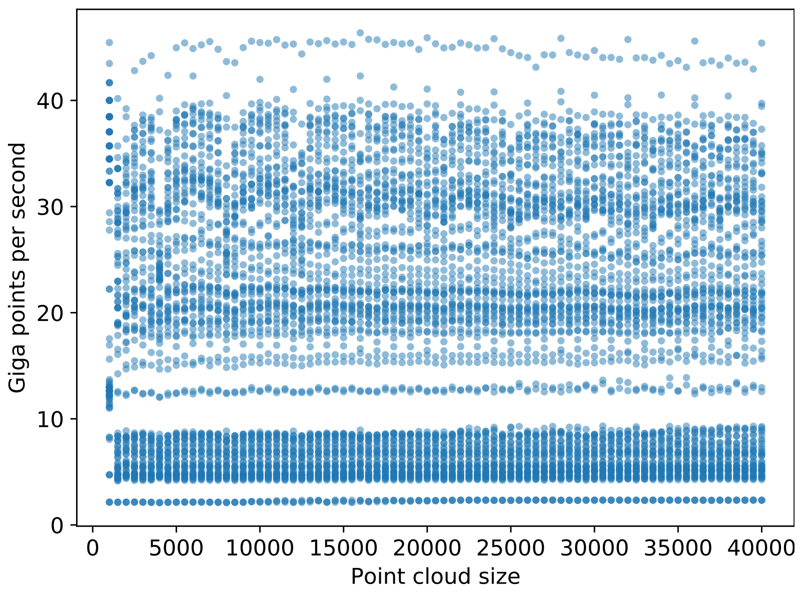


Figure 7. Point set registration performance rate results on the NVIDIA Pascal P100 card with the NVIDIA `nvcc` compiler for all loop blocking configurations and these results are shown for reference only to give the reader an idea of how many runs were performed, how they cluster, and range across point cloud sizes

- the variability of timing measurements is not present on KNL and the graphs are much more smooth especially for the data sizes when the working set exceeds the cache size and the data has to be streamed from the main memory. This may be attributed to the short speculative execution window in the KNL's execution unit design.

First, Fig. 7 shows all data points collected from running on the P100 card in FP32. Clearly presented in that manner, it is hard to discern specific properties that contributed to the achieved performance levels. We do so in the analysis that follows by taking into account the various hardware features that are specific to GPUs and how they affect the performance of various point set registration implementations.

In Fig. 8, we show our first attempt at more guided measurement of performance on the NVIDIA Kepler K40c GPU which has a feature of not using coalesced reads from the main

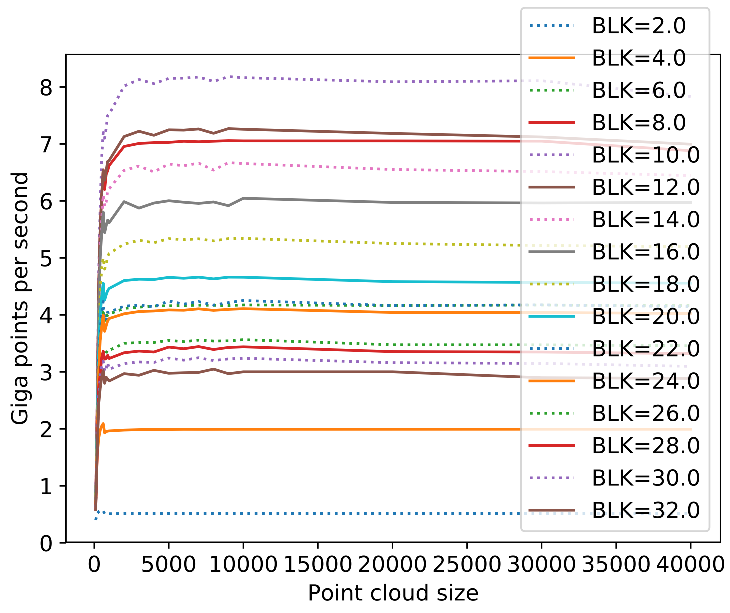


Figure 8. Point set registration performance rate results on the NVIDIA Kepler K40c card with the NVIDIA `nvcc` compiler for various loop blocking configurations with non-coalesced reads

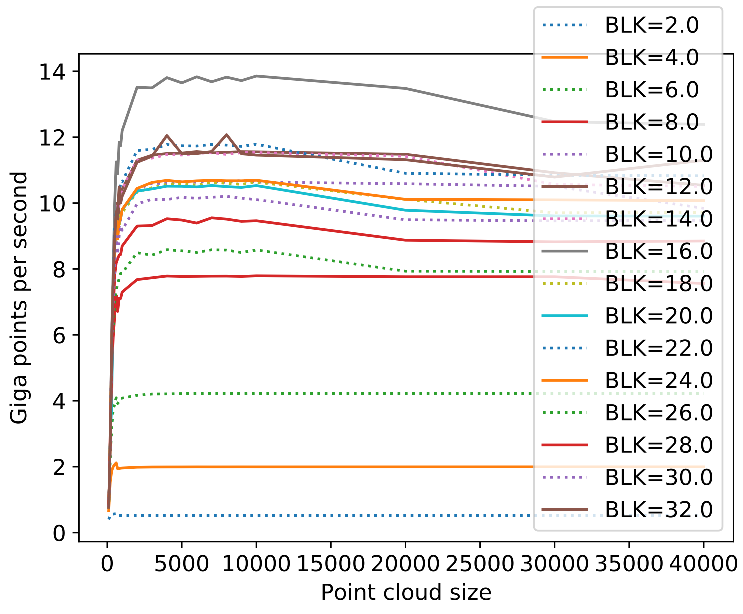


Figure 9. Point set registration performance rate results on the NVIDIA Kepler K40c card with the NVIDIA `nvcc` compiler for various loop blocking configurations with coalesced reads

memory. This clearly goes against the common optimization guidance that is taught to GPU novice users. To show the effect of the problem and how it helps to fix it, Fig. 9 shows the configuration when the coalescent reads were used. The performance increase is nearly 2-fold which confirms the importance of this optimization for the registration code and for our tested implementations.

An arguably more counter-intuitive result occurs when we forgo the use of the shared memory which is a user-controlled scratchpad cache. This special buffer is often used for complex memory patterns which on occasion lead to very high performance improvements. To a limited extent, this may be observed for the tested registration algorithm and we evaluated it for our implementations to find out if it happened to be the case for our tests. Figure 10 presents the results to show that

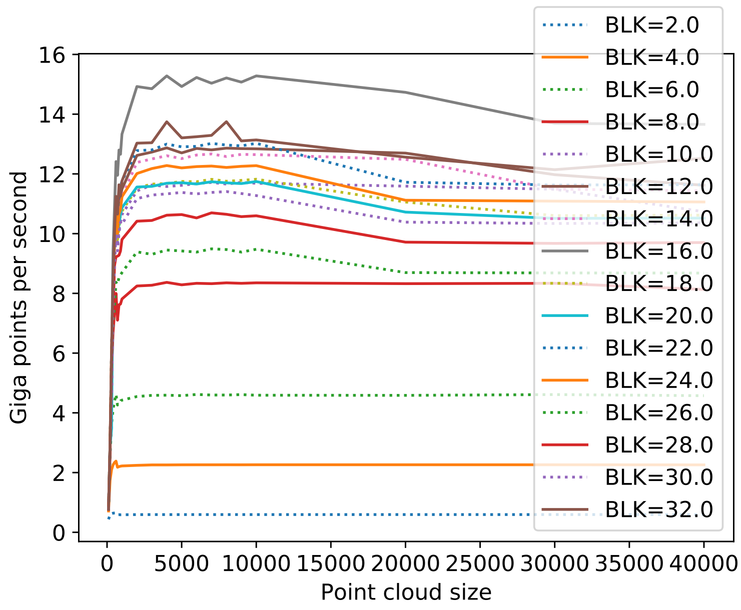


Figure 10. Point set registration performance rate results on the NVIDIA Kepler K40c card with the NVIDIA `nvcc` compiler for various loop blocking configurations without the use of shared

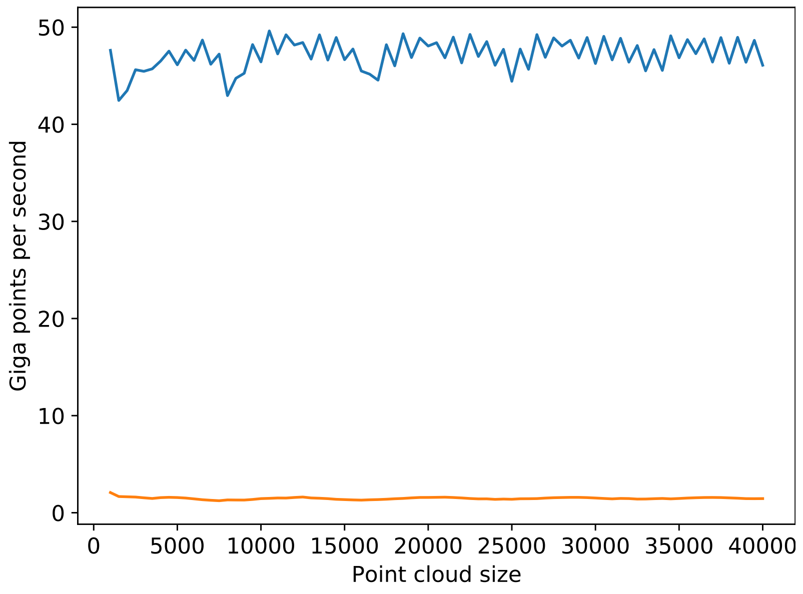


Figure 11. Point set registration performance rate results on the NVIDIA Pascal P100 card with the NVIDIA `nvcc` compiler for the best and worst autotuning configurations using FP16 arithmetic

it is possible to gain a percentage point or more in terms of performance if the shared memory is not used and the main memory is accessed directly.

3.5. Limited Precision Implementation

Finally, we close the results section with tests that measure the influence of low-precision hardware on the performance of our EM-ICP implementations. To that end, we use an NVIDIA Pascal GPU card with the P100 chip. From our perspective, that accelerator features dedicated FP16 units in each Streaming Multiprocessor (SM). These units perform the arithmetic instructions at twice the rate of the corresponding FP32 instructions. Also, the data size needed for the

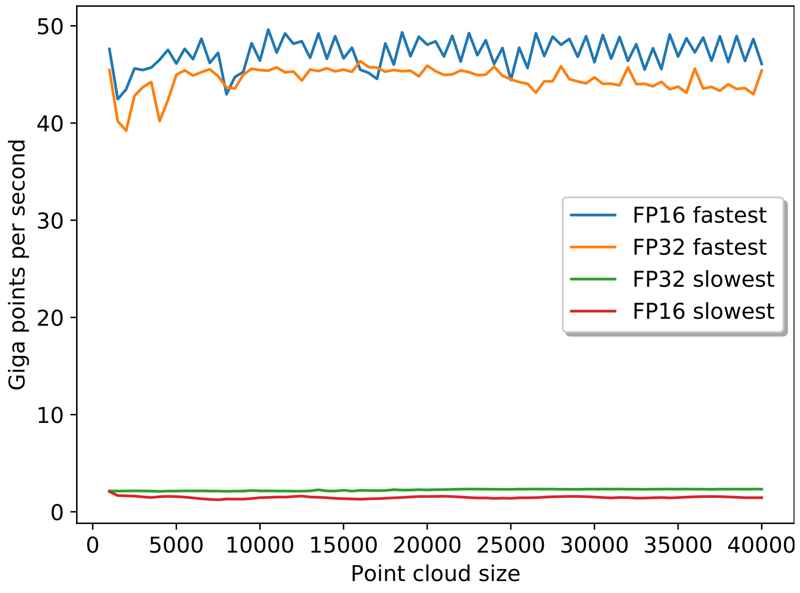


Figure 12. Point set registration performance rate results on the NVIDIA Pascal P100 card with the NVIDIA `nvcc` compiler for the best and worst autotuning configurations using either FP16 or FP32 arithmetic

```
#pragma omp parallel for
for ($i \leftarrow 0$; $i < n_X$; $i \leftarrow i+1$)
  for ($j \leftarrow 0$; $j < n_Y$; $j \leftarrow j+1$)
    $\boxed{\cdots}$
```

```
#pragma omp parallel for collapse(2)
for ($i \leftarrow 0$; $i < n_X$; $i \leftarrow i+1$)
  for ($j \leftarrow 0$; $j < n_Y$; $j \leftarrow j+1$)
    $\boxed{\cdots}$
```

Figure 13. Optimal Configuration for GNU Compiler on x86

FP16 instructions is twice as small and, consequently, it would be obvious to expect about a two-fold increase in performance for both compute-bound and bandwidth-bound codes. Both of these are present in the registration methods tested here.

To give the reader an idea as to what number of autotuning tests we conducted in an automated manner, and the number of the generated code variants from the eligible parameter configurations, we used Fig. 7 with all measurements. But to more clearly indicate the possible range of performance metrics, Fig. 11 shows the best and the worst performance rate when FP16 arithmetic is used. At first, it may seem somewhat underwhelming especially when compared with all the prior results presented so far in previous sections. To identify in detail how FP16 and FP32 arithmetic results compare against each other, we focus in Fig. 12 on just the most relevant data. The figure attempts to indicate how the FP16 arithmetic performs when compared with FP32 and whether it is faster in absolute terms. Unfortunately, the improvement is not as high as the 2-fold higher performance rate suggested by the raw hardware specification (2-fold faster execution and twice as much bandwidth) are not realized in practice in our experiments. We will examine this particular result next in greater detail by looking at low-level artifacts including as assembly-level code and the instruction mix that we believe contributes to the observed effect.

```
// FP16-FP32 conversions contribute additional
// overheads and hit against hardware limit
cvt.f32.f16      f, r  // convert from FP16
sqrt.approx.f32  f, f  // compute in FP32
cvt.rn.f16.f32   r, f  // convert to FP16
```

Figure 14. A Glance at PTX (NVIDIA's Pseudo-Assembly)

4. Discussion on GNU and FP16 Results

Figure 4 singled out the GNU `gcc` compiler's issues with generating optimized instruction mix for the registration code. Despite our efforts to enable optimized parallelization levels we saw the lack of efficiency in the use of multi-threading as it exhibited low core utilization rates which resulted in inferior execution speed. This potentially may be remedied by merging deep loop nests into a combined index set. That set is then divided (in some fashion) among the available OpenMP threads. This is done manually by inserting specifically crafted pragma directives. OpenMP standard specifies `collapse` clause to have this loop-nest merging performed by the GNU compiler. Such an optimization may be applied as shown in Fig. 13. Our autotuning framework easily allows the user to introduce the clause conditionally in the generated code variants and then tests the resulting binaries with a number of loop-collapsing parameters. In the end, this allows the autotuning process to find out the optimal setting for the tested hardware platform and the software stack combination.

Also somewhat surprising result was uncovered during the autotuning process and may be observed in Fig. 12. What is shown is comparatively small difference between the performance obtained from the implementation variants that use either FP16 and FP32 floating-point arithmetic. As our analysis indicates, this may be fully attributed to the overhead of conversion instructions. We confirmed this information by analyzing the PTX assembly-level code produced by the NVIDIA `nvcc` compiler and PTX code generator. The PTX code in question is shown in Fig. 14 in a simplified style to focus on relevant code and with majority the extra details removed. Mixing FP16 and FP32 data on the same GPU engages the hardware conversion units on the NVIDIA Pascal chips and those are limited in number. Much fewer of these units are available when compared, for example, with floating point units in either FP16 and FP32 floating-point precisions. These conversions slow down the overall execution within a single GPU thread warp and contributes to the overall slowdown of the code. Clearly, any performance gains are squandered and we loose what might have been obtained from using FP16 precision if only enough hardware units were available. We observed this phenomenon while we performed autotuning procedure and otherwise it might have not been exposed if it was obstructed by other bottlenecks. It might have also been missed in an non-optimized or manually optimized code. This kind of effects occur mostly in automatically generated code and are otherwise hidden due to a limited exposure of the compiler to the large variety implementation variants that are possible through autotuning.

Concluding Remarks and Future Work

In this article, we presented an application of the autotuning approach to the EM-ICP algorithm. EM-ICP is a stochastic method used for the point set registration problem and we used it for 3D point clouds. In our tests, we applied a variety of automated transformations which resulted in an improved performance. We also used an automated process of generating a set of implementation variants. First, this allowed us to largely exceed the performance achieved by the reference code that was only optimized manually. Second, we searched the large parameter space of potential performance-oriented implementations to arrive at stable and portable performance levels. Those made the resulting EM-ICP codes available not on one but across a large range of hardware platforms and software stacks. In particular, our methodology and autotuning framework generated implementations for multicore, many-core, and accelerator-equipped machines.

We plan on extending this work to a wider range of algorithm that are available for the variants of the registration problem. Creating efficient codes for these methods is a natural future direction for us. We also intend to pursue more experimental approaches. However, due to a large number of algorithms and algorithmic variants for the point set registration problem, we intend to guide our selection of the promising method by the need of the relevant scientific fields. There are many such fields in need of point set registration implementations for analysis of large data sets such as APT experiments. These computational science fields benefit the most from our efforts in the performance engineering domain that we presented above. This is because we manage to achieve high execution rates which clearly contribute to ability to process large volumes of data coming from the experimental instrumentation tools such as Atomic Probe Tomography microscopes in material science and HEA material design.

Acknowledgements

This research was supported by United States National Science Foundation through grant 1642441.

References

1. Bartok, A., Kondor, R., Csanyi, G.: On representing chemical environments. Physical Review B 87(18) (2013), DOI: 10.1103/PhysRevB.87.184115
2. Besl, P.J., McKay, N.D.: A method for registration of 3-D shapes. IEEE PAMI 14(2), 239–256 (1992), DOI: 10.1109/34.121791
3. Chui, H., Rangarajan, A.: A feature registration framework using mixture models. In: IEEE Workshop on MMBIA. pp. 190–197 (2000), DOI: 10.1109/MMBIA.2000.852377
4. Chui, H., Rangarajan, A.: A new algorithm for non-rigid point matching. In: CVPR. vol. 2, pp. 44–51. IEEE Press (2000), DOI: 10.1016/S1077-3142(03)00009-2
5. Chui, H., Rangarajan, A.: A new point matching algorithm for non-rigid registration. CVIU 89(2–3), 114–141 (2003), DOI: 10.1016/S1077-3142(03)00009-2
6. Du, S., Zheng, N., Xiong, L., Ying, S., Xue, J.: Scaling iterative closest point algorithm for registration of md point sets. Journal of Visual Communication and Image Representation 21(5–6), 442–452 (2010), DOI: 10.1016/j.jvcir.2010.02.005

7. Fitzgibbon, A.W.: Robust registration of 2D and 3D point sets. Image and Vision Computing 21, 1145–1153 (2003), DOI: 10.1016/j.imavis.2003.09.004

8. Gao, M.C., Yeh, J.W., Liaw, P.K., Zhang, Y.: High-entropy alloys: fundamentals and applications. Springer International Publishing Switzerland (2016), DOI: 10.1007/978-3-319-27013-5

9. Gold, S., Lu, C.P., Rangarajan, A., Pappu, S., Mjolsness, E.: New algorithms for 2D and 3D point matching: Pose estimation and corresp. In: NIPS. vol. 7, pp. 957–964. The MIT Press (1995), DOI: 10.1016/S0031-3203(98)80010-1

10. Granger, S., Pennec, X.: Multi-scale EM-ICP: A fast and robust approach for surface registration. In: et al., A.H. (ed.) ECCV 2002. pp. 418–432. LNCS 2353, (c) Springer-Verlag Berlin Heidelberg (2002), DOI: 10.1007/3-540-47979-1_28

11. Gupta, S., Agrawal, A., Gopalakrishnan, K., Narayanan, P.: Deep learning with limited numerical precision. CoRR abs/1502.02551 (2015), `http://arxiv.org/abs/1502.02551`, accessed: 2018-08-01

12. Gupta, S., Agrawal, A., Gopalakrishnan, K., Narayanan, P.: Deep learning with limited numerical precision. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 37, pp. 1737–1746. PMLR, Lille, France (2015), `http://proceedings.mlr.press/v37/gupta15.html`, accessed: 2018-08-01

13. Hockney, R.W., Curington, I.J.: $f_{\frac{1}{2}}$: A parameter to characterize memory and communication bottlenecks. Parallel Computing 10(3), 277–286 (1989), DOI: 10.1016/0167-8191(89)90100-2

14. Kim, Y.M., Morozovska, A., Eliseev, E., Oxley, M., Mishra, R., Selbach, S., Grande, T., Pantelides, S., Kalinin, S., Borisevich, A.: Direct observation of ferroelectric field effect and vacancy-controlled screening at the $BiFeO_3/La_xSr_{1-x}MnO_3$ interface. Nature Materials 13(11), 1019–1025 (2014), DOI: 10.1038/nmat4058

15. Larson, D., Prosa, T., Ulfig, R., Geiser, B., Kelly, T.: Local Electrode Atom Probe Tomography: A User's Guide. Springer-Verlag New York (2013), DOI: 10.1007/978-1-4614-8721-0

16. Luszczek, P., Gates, M., Kurzak, J., Danalis, A., Dongarra, J.: Search space generation and pruning system for autotuners. In: Proceedings of IPDPSW. pp. 1545–1554. The Eleventh International Workshop on Automatic Performance Tuning (iWAPT) 2016, IEEE, Chicago, IL, USA (2016), DOI: 10.1109/IPDPSW.2016.197

17. Luszczek, P., Kurzak, J., Yamazaki, I., Dongarra, J.: Towards numerical benchmark for half-precision floating point arithmetic. In: 2017 IEEE High Performance Extreme Computing Conference (2017), DOI: 10.1109/HPEC.2017.8091031

18. Luszczek, P., Kurzak, J., Yamazaki, I., Keffer, D., Dongarra, J.J.: Scaling point set registration in 3D across thread counts on multicore and hardware accelerator platforms through autotuning for large scale analysis of scientific point clouds. In: 2017 IEEE International Conference on Big Data (Big Data). pp. 2893–2902. Boston, MA, USA (2017), DOI: 10.1109/BigData.2017.8258258

19. Miller, M.K., Forbes, R.G.: Atom-Probe Tomography: The Local Electrode Atom Probe. Springer US (2014), DOI: 10.1007/978-1-4899-7430-3

20. Myronenko, A., Song, X., Carreira-Perpiñán, M.A.: Non-rigid point set registration: Coherent Point Drift. In: Schölkopf, B., Platt, J.C., Hoffman, T. (eds.) Advances in Neural Information Processing Systems 19. pp. 1009–1016. MIT Press (2007)

21. Rangarajan, A., Chui, H., Mjolsness, E., Davachi, L., Goldman-Rakic, P.S., Duncan, J.S.: A robust point matching algorithm for autoradiograph alignment. Medical Image Analysis 1(4), 379–398 (1997), DOI: 10.1016/S1361-8415(97)85008-6

22. Rusinkiewicz, S., Levoy, M.: Efficient variants of the ICP algorithm. In: International Conference on 3D Digital Imaging and Modeling (3DIM). pp. 145–152 (2001), DOI: 10.1109/IM.2001.924423

23. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA). Shanghai, China (2011), DOI: 10.1109/ICRA.2011.5980567

24. Sohlberg, K., Rashkeev, S., Borisevich, A., Pennycook, S., Pantelides, S.: Origin of anomalous Pt-Pt distances in the Pt/Alumina catalytic system. ChemPhysChem 5(12), 1893–1897 (2004), DOI: 10.1002/cphc.200400212

25. Volkov, V., Demmel, J.W.: Benchmarking GPUs to Tune Dense Linear Algebra. In: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing. pp. 31:1–31:11. SC '08, IEEE Press, Piscataway, NJ, USA (2008), `http://dl.acm.org/citation.cfm?id=1413370.1413402`, accessed: 2018-08-01, DOI: 10.1145/1413370.1413402

26. Zhang, Y., Zuo, T.T., Tang, Z., Gao, M.C., Dahmen, K.A., Liaw, P.K., Lu, Z.P.: Microstructures and properties of high-entropy alloys. Progress in Materials Science 61, 1–93 (2014), DOI: 10.1016/j.pmatsci.2013.10.001

27. Zhang, Z.: Iterative point matching for registration of free-form curves and surfaces. International Journal of Computer Vision 13(2), 119–152 (1994), DOI: 10.1007/BF01427149

DOI: 10.14529/jsfi180405

Benchmarking Quantum Chemistry Methods in Calculations of Electronic Excitations

Vladimir A. Mironov[1]***, Bella L. Grigorenko***[1,2]***, Igor V. Polyakov***[1]***,***
Alexander V. Nemukhin[1,2]

Quantum chemistry methods are applied to obtain numerical solutions of the Schrödinger equation for molecular systems. Calculations of transitions between electronic states of large molecules present one of the greatest challenges in this field which require the use of supercomputer resources. In this work we describe the results of benchmark calculations of electronic excitation in the protein domains which were designed to engineer novel fluorescent markers operating in the near-infrared region. We demonstrate that such complex systems can be efficiently modeled with the hybrid qunatum mechanics/molecular mechanics approach (QM/MM) using the modern supercomputers. More specifically, the time-dependent density functional theory (TD-DFT) method was primarily tested with respect to its performance and accuracy. GAMESS (US) and NWChem software were benchmarked in direct and storage-based TDDFT calculations with the hybrid B3LYP density functional, both showing good scaling up to 32 nodes. We note that conventional SCF calculations greatly outperform direct SCF calculations for our test system. Accuracy of TD-DFT excitation energies was estimated by a comparison to the more accurate *ab initio* XMCQDPT2 method.

Keywords: quantum chemistry, multi-scale approaches, parallel algorithms, fluorescent proteins.

Introduction

Over the past few decades, advancements in supercomputer technology led to a dramatic rise of computational resources available to scientists in chemistry and physics. Modern computational chemistry methods achieve numeric solution of the Schrödinger equation for molecular systems with different kind of approximations. The ultimate goal is to have an accurate enough description of a very large molecular system with as little computational effort as possible. Complexity of the computational chemistry methods ranges from $\mathcal{O}(N)$ to $\mathcal{O}(N!)$ where N is the size of a molecular system, while usually the most accurate methods are the most expensive ones. A popular compromise which allows conducting a routine study of large biomolecules is the use of fragmentation techniques and the density functional theory approach. Such methods are available in a wide number of modern quantum chemistry software which is more or less adapted to efficient use of modern computer clusters of multicore nodes.

In this communication, performance and accuracy of the time-dependent density functional theory (TD-DFT) in calculations of electronic excitation in a complex molecular model system (described in the next subsection) were studied. The TD-DFT implementations in two popular open-source quantum chemistry packages were used for benchmarking, namely NWChem [5] and GAMESS (US) [4, 7]. Both timings and the computed vertical excitation parameters (energies and oscillator strengths) were collected for analysis.

[1]Department of Chemistry, M.V. Lomonosov Moscow State University, Moscow, Russia
[2]N.M. Emanuel Institute of Biochemical Physics, Russian Academy of Sciences, Moscow, Russia

1. Methodology

The scaling and performance benchmarks were carried out on “Lomonosov-2” [3] supercomputer (1696 nodes, single-socket Intel Haswell-EP E5-2697v3, 64 Gb RAM, NVidia Tesla K40M). We used the NWChem v6.6 package installed on the supercomputer. The GAMESS (US) 2016r1 package was compiled manually using Intel Parallel Studio 2015. Both packages were linked with OpenMPI 1.8.4 message-passing library.

The model chemical system shown in Fig. 1 was used for benchmarking of quantum chemical calculations of electronic excitations.

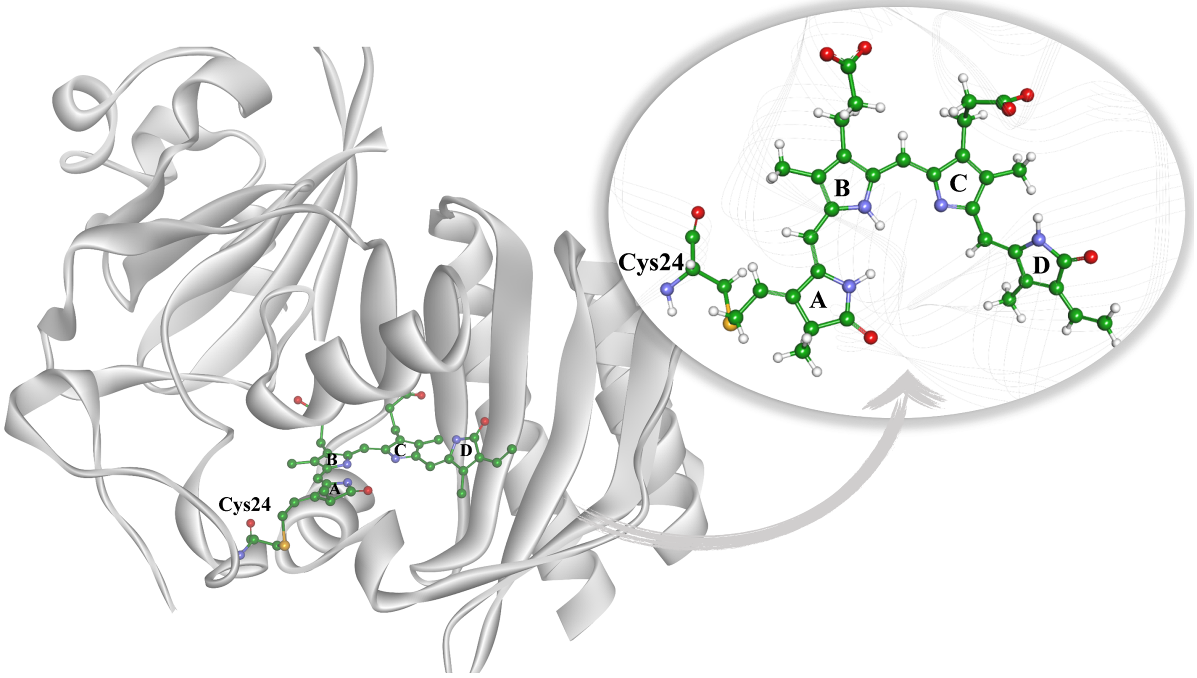


Figure 1. Molecular model of the near-infrared fluorescent protein (left) and the protein-bound biliverdin (BV) chromophore BV (right inset). Carbon atoms are shown in green, oxygen in red, nitrogen in blue, sulfur in yellow, hydrogen in white

Infrared and near-infrared fluorescent proteins are highly in demand for in vivo imaging because their absorption and emission bands fall into the optical transparency window of biological materials [2]. These proteins are engineered from the bacterial phytochrome domains. Detailed characterization of structural and spectral properties of the chromophore-containing protein domains is a necessary step when designing novel variants of these efficient fluorescent markers in living cells.

The equilibrium geometry parameters of this model system were computed using the quantum mechanics/molecular mechanics (QM/MM) approach. All QM/MM calculations were carried out using NWChem quantum chemistry package [5]. The QM subsystem of the optimized protein structure was used in TD-DFT benchmarks. The B3LYP functional [6] and a cc-pVDZ basis set were used in these TD-DFT calculations.

2. Results and Discussion

Each single point TD-DFT calculation has two steps. The first step is a regular ground-state DFT calculation. On this step the Kohn–Sham equations are solved through the usual iterative self-consistent field (SCF) procedure. The computed density is used on the second step which is actual TD-TFT calculation. Results of the second step are excitation parameters.

There are two algorithms of the SCF procedure. According to the storage-based SCF algorithm, all the required intermediate data are computed prior the first SCF iteration and reused on each following iterations. However, the amount of these data is usually very high. According to another SCF algorithm ("direct" SCF), the intermediate data are not stored at all but recomputed on every SCF iteration. In this case the overhead of recomputing intermediate data increases linearly with the number of iterations. The amount of SCF iterations required to solve the Kohn–Sham equations for the BV chemical system is quite large: it is ≈ 50 for both packages. Thus, it it not surprising that the storage-based SCF algorithm outperforms the direct one (see Tab. 1). The drawback of the storage-based SCF algorithm is its high demand for storage

Table 1. Strong scaling of single-point TD-DFT (B3LYP) calculation of the BV chemical system in NWChem quantum chemistry package. The data is applicable for both direct and storage-based SCF algorithms

Number of nodes	Time, s		Scaling efficiency		
	direct	storage-based	direct, rel. to 1 node	direct, rel. to 8 nodes	storage-based, rel. to 8 nodes
			NWChem		
1	16 455	-	100%	-	-
4	7344	-	56%	-	-
8	3907	2310	53%	100%	100%
16	2204	1288	47%	89%	90%
24	1619	1047	42%	80%	74%
32	1487	995	35%	66%	58%
			GAMESS (US)		
1	10 738	-	100%	-	-
4	2776	-	97%	-	-
8	1446	959	93%	100%	100%
16	788	539	85%	92%	89%
24	581	414	77%	83%	77%
32	474	349	71%	76%	69%

bandwidth and capacity. In this study, a `tmpfs` in-memory partition was used to hold intermediate data. The calculations were therefore limited by the number of nodes used and the amount of system memory on each node. In the particular setup of BV chromophore benchmark, the limit was 8 nodes having 32 GB memory per node. Additionally, the lack of I/O in direct SCF results in a somewhat better scaling when compared to the storage-based algorithm. It is especially important for the NWChem package when the SCF step in a storage-based calculation is up to 4 times faster than a direct SCF step (Tab. 2). We cannot directly compare the performance of the NWChem and GAMESS (US) quantum chemistry packages because they use different setups of exchange-correlation functional integration, different convergence criteria and other hidden

Table 2. The timings of SCF and excitation calculation steps in TD-DFT (B3LYP) benchmark for direct and storage-based SCF algorithms. The benchmark is for 16 nodes, wall clock time is given in seconds

Algorithm	Step	NWChem,s	GAMESS,s
direct SCF	SCF	1175	357
	excitation	1029	431
	total	2204	788
storage-based SCF	SCF	236	135
	excitation	1052	404
	total	1288	539

parameters. However, when using the default configuration in both packages, GAMESS (US) runs faster than NWChem on "Lomonosov-2" supercomputer on he same amount of nodes.

The recent paper [1] compares the results obtained using the TD-DFT and the more accurate *ab initio* XMCQDPT2 method.

Conclusion

TDDFT benchmarks show good scalability for the test system for up to 32 nodes, while GAMESS (US) significantly outperforms NWChem in both single node performance and in scaling. Conventional SCF calculations greatly outperform direct SCF calculations for our test system, and thus conventional SCF procedure should be used whenever a fast-enough storage is available.

Acknowledgements

This work was supported by the Russian Science Foundation (project №17-13-01051). This research is carried out using the equipment of HPC computing resources at Lomonosov Moscow State University [3].The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University supported by the project RFMEFI62117X0011.

References

1. Polyakov, I.V., Grigorenko, B.L., Mironov, V.A., Nemukhin, A.V.: Modeling structure and excitation of biliverdin-binding domains in infrared fluorescent proteins. Chemical Physics Letters (2018), DOI: 10.1016/j.cplett.2018.08.068

2. Chernov, K.G., Redchuk, T.A., Omelina, E.S., Verkhusha, V.V.: Near-infrared fluorescent proteins, biosensors, and optogenetic tools engineered from phytochromes. Chemical Reviews

117(9), 6423–6446 (2017), DOI: 10.1021/acs.chemrev.6b00700

3. Sadovnichy, V., Tikhonravov, A., Voevodin, V., Opanasenko, V.: "Lomonosov": Supercomputing at Moscow State University. In: Vetter, J.S. (ed.) Contemporary High Performance Computing: From Petascale toward Exascale, pp. 283–307. Chapman & Hall/CRC Computational Science, Chapman & Hall/CRC, Boca Raton, United States (2013)

4. Gordon, M.S., Schmidt, M.W.: Advances in electronic structure theory: GAMESS a decade later. In: Dykstra, C., Frenking, G., Kim, K., Scuseria, G. (eds.) Theory and Applications of Computational Chemistry: The First Forty Years, pp. 1167–1189. Elsevier, Amsterdam (2005), DOI: 10.1016/B978-044451719-7/50084-6

5. Valiev, M., Bylaska, E., Govind, N., Kowalski, K., Straatsma, T., Van Dam, H., Wang, D., Nieplocha, J., Apra, E., Windus, T., de Jong, W.: NWChem: A comprehensive and scalable open-source solution for large scale molecular simulations. Computer Physics Communications 181(9), 1477–1489 (2010), DOI: 10.1016/j.cpc.2010.04.018

6. Becke, A.D.: Density-functional exchange-energy approximation with correct asymptotic behavior. Physical Review A 38(6), 3098–3100 (1988), DOI: 10.1103/PhysRevA.38.3098

7. Schmidt, M.W., Baldridge, K.K., Boatz, J.A., Elbert, S.T., Gordon, M.S., Jensen, J.H., Koseki, S., Matsunaga, N., Nguyen, K.A., Su, S., Windus, T.L., Dupuis, M., Montgomery, J.A.: General atomic and molecular electronic structure system. Journal of Computational Chemistry 14(11), 1347–1363 (1993), DOI: 10.1002/jcc.540141112

DOI: 10.14529/jsfi180406

Developing Quasi-Steady Model for Studying Hemostatic Response Using Supercomputer Technologies

Petr V. Trifanov[1], *Valeriia N. Kaneva*[1,2,3], *Sergei V. Strijhak*[4], *Mikhail A. Panteleev*[1,2,3,5], *Fazoil I. Ataullakhanov*[1,2,3,5], *Joanne Dunster*[6], *Vadim V. Voevodin*[7], *Dmitry Y. Nechipurenko*[1,2,3]

Formation of the platelet plug represents a primary response to the vessel wall injury, but may also result in vessel occlusion. The decrease of the local blood flow due to platelet thrombus formation may lead to serious complications, such as ischemic stroke and myocardial infarction. However, mechanisms responsible for regulation of thrombus dynamics are not clear. In order to get a deeper insight into the role of blood flow and platelet interactions in the formation of the primary platelet plug we developed a particle-based model of microvascular thrombosis using quasi-steady flow approximation. In order to simulate thrombus dynamics at physiologically relevant timescales of several minutes, we took advantage of the supercomputer technologies. Our *in silico* analysis revealed the importance of platelet size heterogeneity for describing experimental data on microvascular thrombus formation. Thus, our model represents a useful tool for the supercomputer-aided computational analysis of thrombus dynamics in the microvessels on physiologically relevant timescales.

Keywords: cfd, particle-based model, biorheology, thrombosis, program profiling.

Introduction

Despite many decades of both fundamental and clinical research, complications associated with arterial thrombosis — such as myocardial infarctions and ischemic strokes — still represent a greater cause of mortality and morbidity in developed countries [7]. The reason why some of the vessel wall injuries result in the formation of stable occlusive thrombus, which compromises a local blood flow, while the others not — is one the greatest unresolved issues in the field of hemostasis and thrombosis. Thrombus formation at the site of a vascular injury is a highly complex process, which involves platelet adhesion, aggregation, plasma coagulation and takes place under conditions of the blood flow [6]. An enormous number of mechanical and biochemical interactions occurring within the growing thrombus represent a serious challenge for elucidation of the governing mechanisms and justify the relevance of computational models. During the last decade, various models of thrombus formation have been described, each focused on different aspects of the process [1]. However, a clear understanding of mechanisms responsible for complex dynamics of thrombus observed *in vivo* is still missing. In order to clarify the role of reversible platelet-platelet interactions and platelet-flow interactions in thrombus dynamics, we have developed a novel particle-based *in silico* model of microvascular thrombus formation. In this paper, we show that our model reproduces complex dynamics of thrombus observed after laser-induced

[1]Department of Physics, Lomonosov Moscow State University, Moscow, Russia
[2]Dmitry Rogachev National Research Center of Pediatric Hematology, Oncology and Immunology, Moscow, Russia
[3]Center for Theoretical Problems of Physicochemical Pharmacology, Moscow, Russia
[4]Ivannikov Institute for System Programming of the RAS, Moscow, Russia
[5]Faculty of Biological and Medical Physics, Moscow Institute of Physics and Technology, Dolgoprudny, Russia
[6]Institute for Cardiovascular and Metabolic Research, School of Biological Sciences, University of Reading, Berkshire, United Kingdom
[7]Research Computing Center of Lomonosov Moscow State University, Moscow, Russia

injury of arterioles in mice [11]. In order to simulate thrombus dynamics on the physiologically relevant timescales, we used quasi-steady flow approximation and took advantage of supercomputer technologies. Our approach involves stochastic modeling. Thus, multiple simulations are required for a statistically valid analysis of thrombus dynamics even for a single combination of model parameters. Such a computational demand can be satisfied by the supercomputer which allows performing multiple simulations in parallel. We performed such computations in several weeks using the resources of the Lomonosov-2 supercomputer.

In this paper, we describe the model in the Section 1. Section 2 is devoted to the model validation and Section 3 embodies evaluation of the model performance. Conclusion summarizes the study and points directions for further work.

1. Materials and Methods

In order to get an insight into complex dynamics of thrombus observed *in vivo*, including disruptions of external thrombus layers, we developed a particle-based model, which accurately describes platelet-platelet interactions. To address thrombus dynamics *in silico* on physiologically relevant timescales of several minutes, we restricted the model complexity by considering a two-dimensional case. Our particle-based model considers platelets as discs with certain radius, which may interact with vessel wall (considered as elastic boundary), other platelets and blood flow. Blood is considered as a Newtonian, incompressible fluid, described with Navier–Stokes and the continuity equations.

The model consists of two main modules. The first one describes the particle (platelet) dynamics in a given flow field, which is considered stationary during the given period of time. The second module is responsible for computation of the velocity and pressure fields in case the platelet dynamics, inferred by the first module, results in the change of the thrombus shape.

Platelet dynamics module. The model takes into account two types of inter-platelet interaction: primary reversible (glycoprotein-Ib-mediated) and secondary, which intensifies with the increase of platelet activation (integrins-mediated). The first type of interaction was described by stochastically associating and dissociating springs, while the second type was represented by deterministic Morse potential. Platelet activation was described as a time-dependent process.

Parameters of platelet interaction models were acquired based on experimental data. We inferred stochastic springs model parameters using experiments on platelet rolling over adhesive surface [3]. Parameters of Morse potential were obtained by fitting experimental data on forces between single activated platelets [8]. The Newtonian equations of motion for each particle were solved numerically with a modified version of the Verlet algorithm [12]. The hydrodynamic force, acting on a freely flowing particle, was described by the Stokes law, while forces acting on platelets within the aggregate were calculated using numerical solution of the continuity Navier–Stokes equations, provided by the CFD module. Platelets were randomly generated at the inlet according to local velocity values and mean platelet concentration profile inferred from experimental data on platelet margination effect [13].

Computational fluid dynamics module. If platelet dynamics calculated using the first module resulted in the change of the aggregate structure (e.g. attachment or detachment of the platelet to or from the aggregate, respectively, or significant displacement of platelets within the aggregate), then CFD module was initialized with new boundary conditions. All platelets, which form the aggregate, were considered as solid impermeable barriers with no-slip boundary condition on their surface (contours). In order to account for fluid motion inside the aggregate, the effective hydrodynamic radius of these platelets was considered two times smaller than the corresponding value used for the calculation of platelet interactions in the platelet dynamics module. The continuity equation and Navier–Stokes equations were solved numerically using the OpenFOAM simpleFoam solver [9], based on Semi-Implicit Pressure Linked Equations (SIMPLE) algorithm, with parabolic inflow boundary condition on the inlet and zero pressure condition on the outlet.

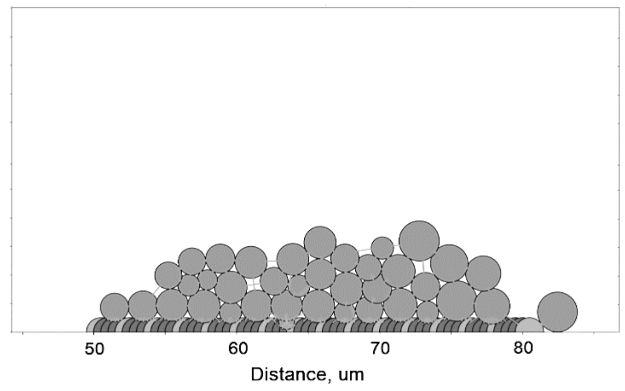
No-slip boundary conditions were chosen for the vessel walls and platelets within the aggregate. The unstructured mesh was generated using a gmsh generator [4]. When primary calculation is performed, the flow velocities and local pressure values are recalculated based on rescaled boundary conditions corresponding to a constant pressure drop, which better describes the hydrodynamic conditions observed *in vivo* [2].

The computational domain represented a rectangle with the length of 200 μm and height of 35 μm and mimicked a fragment of the arteriole. An injury was modeled as a row of firmly adhered platelets on the lower side of the vessel wall (Fig. 1). The viscosity equaled 10^{-3} $Pa \cdot s$, while maximum flow in the inlet was taken 8.75 mm/s in order to provide surface shear rate of 1000 s^{-1}, a typical scale for arterioles.

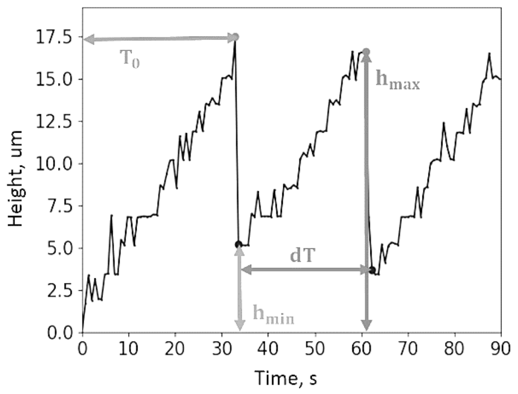
2. Results

In order to validate the model, we performed the analysis of thrombus dynamics obtained *in silico* and compared it to the corresponding data derived from *in vivo* experiments of thrombus formation in the presence of coagulation inhibitor [11]. In order to account for platelet size heterogeneity, the radius of each disk was generated from the Gaussian distribution, which fits experimental data on platelet size distribution [10]. We compared the dynamics of this model with a simple model variant where all platelets have the same radius of 1 micron.

Figure 1 shows a typical shape of thrombus and typical dynamics of thrombus height in the model with different platelet sizes.



(a) The typical view of thrombus: the layer of circles at the bottom — the injury, other circles — platelets, the lines between the surfaces of the platelets — the springs-mediated interactions



(b) Example of the thrombus height dynamics showing principal parameters used for analysis

Figure 1. Platelet aggregate structure and dynamics of its formation

The results of 12 simulations, each corresponding to 90 seconds of physiological time were analyzed. Thrombus disruption was determined as a sharp decrease in the number of platelets within the thrombus. We analyzed the following parameters of thrombus dynamics (Fig. 1): thrombus surface (S_{max} and S_{min}) and thrombus heights (h_{max} and h_{min}) before and after disruption, respectively. We also analyzed the relative size of disruptions, defined as $[S_{max} - S_{min}]/S_{max}$. The statistics for these parameters with the number of disruption events is given in the Tab. 1.

Table 1. Thrombus dynamics parameters inferred from *in silico* and *in vivo* experiments. Data presented as mean ± standard error of mean

Parameter type	Model with platelets of equal size (12 simulations, 27 disruptions)	Model with platelets of the different size (12 simulations, 24 disruptions)	Experimental data(1 experiment, 3 disruptions) [11]
S_{max}, μm^2	609 ± 19	666 ± 16	540 ± 40
S_{min}, μm^2	320 ± 16	415 ± 22	380 ± 40
ΔS, μm^2	289 ± 22	251 ± 25	163 ± 14
$\Delta S/S_{max}$	0.47 ± 0.04	0.37 ± 0.04	0.30 ± 0.05
h_{max}, μm	15.2 ± 0.4	14.6 ± 0.3	16 ± 1
h_{min}, μm	7.4 ± 0.2	9.4 ± 0.6	10 ± 1

Our *in silico* results demonstrate that taking into account platelet size heterogeneity results in thrombus dynamics which better reproduces experimental data. Interestingly, this result indicates that variations in platelet sizes lead to decrease in both absolute and relative size of the disrupting parts (embolus size) of the thrombus and thus seem to increase its stability against the flow.

3. Evaluation and Analysis of Application Behavior

The described approach was implemented in C++ language using OpenFOAM, gmsh and boost libraries. The experiments were conducted on the Lomonosov-2 supercomputer in *compute* partition (based on Intel Xeon E5-2697 v3 processors with 14 cores). For each parameter set the computations were performed on one node. In order to get the statistics, 14 simulations with different random seeds were initialized, each at a single thread mode, therefore utilizing all available physical cores in the node.

To estimate the increase of calculation time associated with quasi-steady hydrodynamics computations, we compared the mean time required to calculate 60 seconds of physiological time in a stationary model (when the flow is stationary and no CFD simulations are performed) and quasi-steady model. The obtained values for n=10 runs are the following: for stationary model, the calculation time (M ± SE) is 560 ± 50 min.; the time for quasi-steady model is 1960 ± 180 min.

In order to find out the most computationally intensive parts of our implementation (that are mostly responsible for the significant increase in execution time), we have performed program profiling using Intel VTune Amplifier [5]. It should be noted that in these experiments we calculated a much smaller time period — only 500 milliseconds of physiological time. This was done due to two reasons: 1) the profiling process requires a large number of experiments and therefore is very time-consuming; 2) we have chosen an interval of physiological time that reflects a representative as well as computationally expensive part of the simulation.

The results have expectedly shown that the CFD module tends to consume most of the execution time (∼60% in general), leaving only 40% to the platelet dynamics module. The most computationally intensive function in CFD module is responsible for infering the indexes of the edges which surround the given point in space for further calculation of the interpolated values of pressure and velocity — it takes up to 9.4% of the overall execution time. Next time-consuming steps in CFD part are marking particles which belong to the aggregate as well as calculation of distance between particles, since they consume ∼4% and 3.5%, correspondingly.

The interesting part was to discover that a significant part of the execution time is spent on performing useful arithmetic operations — sin/cos, simple arithmetic and exponent calculations (15.1%, 8.7% and 5.1% of execution time, correspondingly). Moreover, performing sin/cos operations is the overall most computationally intensive function. All these operations are called from one major function that calculates the interactions between particles within the aggregate, which makes this function one of the primary candidates for further optimization. Another way of optimization, which seems to be more preferred in our case, is to try parallelizing this program. According to the analysis results, it seems that the most suitable opportunity for parallelization is to perform a recalculation of particle dynamics in parallel (e.g. using OpenMP), since this part is performed in a nested loop with almost independent iterations where the program spends 40% of execution time.

Conclusions

In this paper we describe a novel *in silico* model of thrombus formation and its implementation, which allows studying thrombus dynamics on the physiologically relevant timescales of several minutes. Using this new instrument we demonstrate the importance of platelet size heterogeneity for describing experimental data on thrombus dynamics and embolization. To perform mass experiments based on this model, we have used computing resources of the Lomonosov-2 supercomputer. In order to find out the performance bottlenecks of our implementation, we have performed program profiling using Intel Vtune software. The analysis results showed us the most computationally intensive parts of our program and allowed us to identify possible directions for optimization.

Acknowledgements

The work described in all sections except section 3 (in part of program profiling and analysis) was supported by the Russian Federation President Grant to Young Scientists MD-229.2017.4 to MP and MK-2706.2017.4 to DN. The results described in section 3 concerning program profiling and analysis were obtained in the Lomonosov Moscow State University with the financial support of the Russian Science Foundation (agreement N 17-71-20114). The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University supported by the project RFMEFI62117X0011.

References

1. Belyaev, A.V., Dunster, J.L., Gibbins, J.M., Panteleev, M.A., Volpert, V.: Modeling thrombosis in silico: Frontiers, challenges, unresolved problems and milestones. Physics of life reviews (2018), DOI: 10.1016/j.plrev.2018.02.005
2. Belyaev, A.V., Panteleev, M.A., Ataullakhanov, F.I.: Threshold of Microvascular Occlusion: Injury Size Defines the Thrombosis Scenario. Biophysical Journal 109(2), 450–456 (2015), DOI: 10.1016/j.bpj.2015.06.019
3. Coburn, L.A., Damaraju, V.S., Dozic, S., Eskin, S.G., Cruz, M.A., McIntire, L.V.: GPIbα-vWF Rolling under Shear Stress Shows Differences between Type 2B and 2M von Willebrand Disease. Biophysical Journal 100(2), 304–312 (2011), DOI: 10.1016/j.bpj.2010.11.084
4. Geuzaine, C., Remacle, J.F.: Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. International Journal for Numerical Methods in Engineering 79(11), 1309–1331 (2009), DOI: 10.1002/nme.2579
5. Intel Vtune Amplifier home page. `https://software.intel.com/en-us/vtune`, accessed: 2018-09-30
6. Jackson, S.P., Nesbitt, W.S., Westein, E.: Dynamics of platelet thrombus formation. Journal of Thrombosis and Haemostasis 7(s1), 17–20 (2009), DOI: 10.1111/j.1538-7836.2009.03401.x
7. Jackson, S.P.: Arterial thrombosisinsidious, unpredictable and deadly. Nature Medicine 17, 1423 (2011), DOI: 10.1038/nm.2515
8. Nguyen, T.H., Palankar, R., Bui, V.C., Medvedev, N., Greinacher, A., Delcea, M.: Rupture Forces among Human Blood Platelets at different Degrees of Activation. Scientific Reports 6, 25402 (2016), DOI: 10.1038/srep25402
9. OpenFOAM User Guide, Version 6, `https://cfd.direct/openfoam/user-guide`, accessed: 2018-09-30
10. Paulus, J.M.: Platelet size in man. Blood 46(3), 321–336 (1975)
11. Stalker, T.J., Traxler, E.A., Wu, J., Wannemacher, K.M., Cermignano, S.L., Voronov, R., Diamond, S.L., Brass, L.F.: Hierarchical organization in the hemostatic response and its relationship to the platelet-signaling network. Blood 121(10), 1875–1885 (2013), DOI: 10.1182/blood-2012-09-457739
12. Verlet, L.: Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. Physical Review 159(1), 98–103 (1967), DOI: 10.1103/PhysRev.159.98
13. Woldhuis, B., Tangelder, G.J., Slaaf, D.W., Reneman, R.S.: Concentration profile of blood platelets differs in arterioles and venules. American Journal of Physiology-Heart and Circulatory Physiology 262(4), H1217–H1223 (1992), DOI: 10.1152/ajpheart.1992.262.4.H1217

DOI: 10.14529/jsfi180407

New Binding Mode of SLURP Protein to α7 Nicotinic Acetylcholine Receptor Revealed by Computer Simulations

***Igor D. Diankin*[1], *Denis S. Kudryavtsev*[2], *Arthur O. Zalevsky*[1,2,3], *Victor I. Tsetlin*[2], *Andrey V. Golovin*[1,3,4]**

SLURP-1 is a member of three-finger toxin-like proteins. Their characteristic feature is a set of three beta strands extruding from hydrophobic core stabilized by disulfide bonds. Each beta-strand carries a flexible loop, which is responsible for recognition. SLURP-1 was recently shown to act as an endogenous growth regulator of keratinocytes and tumor suppressor by reducing cell migration and invasion by antagonizing the pro-malignant effects of nicotine. This effect is achieved through allosteric interaction with α7 nicotinic acetylcholine receptors (alpha-7 nAChRs) in an antagonist-like manner. Moreover, this interaction is unaffected by several well-known agents specifically alpha-bungarotoxin.

In this work, we carry out the conformational analysis of the SLURP-1 by a microsecond-long full-atom explicit solvent molecular dynamics simulations followed by clustering, to identify representative states. To achieve this timescale we employed a GPU-accelerated version of GROMACS modeling package. To avoid human bias in clustering we used a non-parametric clustering algorithm Affinity Propagation adapted for biomolecules and HPC environments. Then, we applied protein-protein molecular docking of the ten most massive clusters to α7-nAChRs in order to test if structural variability can affect binding. Docking simulations revealed the unusual binding mode of one of the minor SLURP-1 conformations.

Keywords: molecular dynamics, gromacs, clustering, affinity propagation, protein docking, biomolecules.

Introduction

Three-finger proteins of the Ly6 family have multiple functions across the organism: from lignd-binding domains of growth factors receptors (myostatin receptor) to regulation of nicotinic receptor (nAChR) expression and function in the brain (Lynx1). Most mammalian Ly6 proteins have a GPI anchor at the C-terminus attaching them to the membrane while others do not have it and are secreted. Among the latter is SLURP-1 functioning as a water-soluble paracrine/autocrine messenger molecule which binds nicotinic acetylcholine receptors and regulates keratinocyte growth and angiogenesis. Such properties of SLURP-1 make it a valuable object to study as an endogenous cholinergic ligand similar to snake venom neurotoxins that played a crucial role in the nAChR research for decades [8, 10].

The purpose of the research was to find out alternative SLURP-1 complexes with nAChRs conformation patterns exist. Can there be different options for binding? It was previously shown that SLURP-1 does not compete with alpha-Bgt for the binding to α7 nAChRs. There were data, including NMR data, on the binding of SLURP-1 variants without N- and C-terminal tags and labels. But there is still no evidence about the binding mechanism. SLURP-1 fusion constructs bearing unnatural tags and labels show properties drastically different from those described for recombinant SLURP-1 which has only one additional N-terminal Met residue (PDB ID: 2MUO). Noteworthy, no experimental spatial structure of fusion SLURP-1 proteins

[1]Lomonosov Moscow State University, Moscow, Russian Federation
[2]Shemyakin-Ovchinnikov Institute of Bioorganic Chemistry, Russian Academy of Sciences, Moscow, Russian Federation
[3]I.M. Sechenov First Moscow State Medical University, Moscow, Russian Federation
[4]Faculty of Computer Science, Higher School of Economics, Moscow, Russian Federation

were described. Fusion proteins demonstrated positive allosteric modulation of $\alpha 7$ nAChR while recombinant SLURP-1 act as an inhibitor of the receptor [8].

Since no competition with alpha-bungarotoxin is observed, some alternative binding sites of SLURP-1 should exist. In this work, the hypothesis was tested by molecular modeling methods. Molecular dynamics simulations were used to detect stable SLURP-1 conformations, and protein-protein docking was performed to identify prospective binding sites. In the course of the study, several characteristic patterns appeared that did not always correspond to NMR data. Changes in the secondary structure of one of the fingers were observed. Also, more than one variant of nACHRs binding model was found, and they depended on the conformation of SLURP.

Without experimental verification, it is difficult to determine the correctness of the prediction and unequivocally confirm or disprove the hypothesis.

1. Methods

Molecular dynamics simulation of SLURP-1 NMR conformation (1st model) (PDB ID: 2MUO) was performed in GROMACS [7]. Protein part was described with amber99sb-ildn forcefield [6] and explicit water model tip3p was used as a solvent. Computational resources of Lomonosov supercomputer in conjunction with GPU acceleration allowed us to reach total trajectory length of 2 mks.

The trajectory was clusterized with non-parametric clustering algorithm affinity propagation [5]. The secondary structure was predicted with DSSP [4, 9].

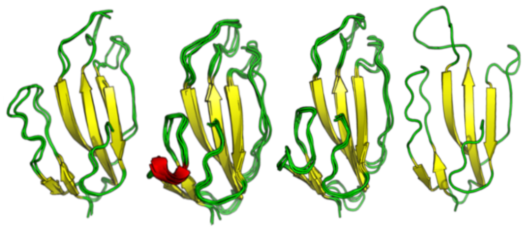
Protein-protein docking for 10 largest clusters was performed with ZDOCK [3] and replicated 10 times. The number of output complexes was set to 1000 and sorted by ZRANK. Top 10 hits from each replicate were selected producing 100 in total. Contacts at a distance less than 3.6 angstroms were calculated, and several models were selected for visual analysis with PyMol [2].

2. Results

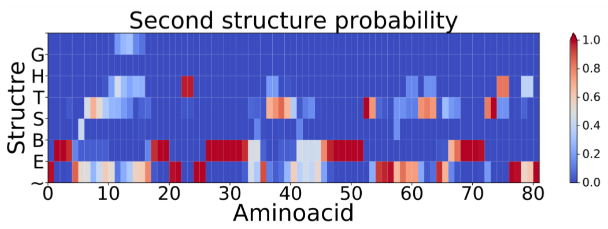
2.1. Conformational Landscape of SLURP-1

During the visual analysis of the simulation of molecular dynamics, SLURP-1 underwent a change in the course of the backbone during the molecular dynamics toward the formation of the alpha helix like motif by the residues SER12, ALA13 and SER14. DSSP analysis of secondary structure over the whole trajectory confirmed formation 3-10 alpha helix for these residues Fig. 1 (a). Alpha helices are colored in red, beta-strands in yellow and unstructured areas in green. On Fig. 1 (b) the probability of participation of each amino acid in the formation of a certain secondary structure is presented, "$\sim$" – unstructured sections, "E" remains involved in the formation of beta strands, "B" – the situation where only one amino acid forms a hydrogen bond characteristic of the beta-strand, "S" and "T" – turns, "H" – alpha helix, "G" – 3-10 alpha helix.

With clustering analysis, 54 clusters were found and only 10 can be formally designated to two groups: with and without 3-10 alpha helix. Additionally, 10 replicas of the molecular dynamics simulation were performed for 100 nanoseconds each, showing reproducibility of the clusters.



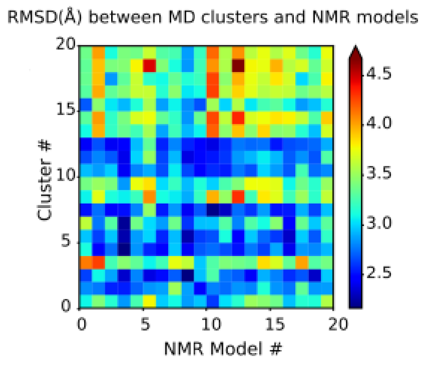
(a) The difference observed in loops and the presence or absence of a 3-10 alpha-helix site



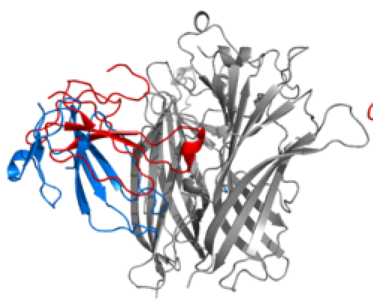
(b) Secondary structure distribution over amino acids

Figure 1. SLURP-1 can form 3-10 alpha-helix by 12-17, amino acids and its loops dynamics

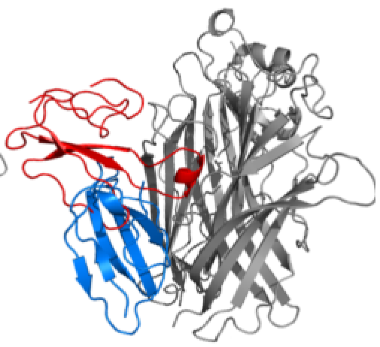
RMSD values were measured for all possible pairs of 20 simulation clusters and 20 NMR models Fig. 2. Remarkably, simulation data generally reproduced the conformation set from the NMR experiment.



(a) Comparison of MD clusters with NMR models



(b) Typical position of SLURP-1 (blue) and bungarotoxin (red)



(c) Atypical position of SLURP-1

Figure 2. Heatmap of MD clusters vs NMR models. Positions of SLURP-1 and bungarotoxin with two subunits of α7 nAChRs (grey)

2.2. Protein-Protein Docking of SLURP-1 to 7 nAChRs

Well-known ligand of α7 nAChRs receptor – alpha-bungarotoxin binds at the interface between subunits in pentamer. Results of protein docking of SLURP-1 conformations to α7 receptor presented two major binding modes. The first mode overlaps with the alpha-bungarotoxin binding site. Ligand interacts with the receptor by "fingers"– loop II and III regions without secondary structure involving residues 134-144 and 157-167 Fig. 2. The second binding mode does not overlap the alpha-bungarotoxin site, and SLURP-1 continued to have stable contact through the "fingers".

Disussion and Conclusions

According to simulation results SLURP-1 has two binding modes for various conformations. First binding mode overlaps with bungarotoxin site while the second one does not. Our data provide clear directions for experimental support of newly proposed mode of interaction of SLURP-1 with α7 nAChRs.

Acknowledgments

The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University [1] supported by the project RFMEFI62117X0011.

References

1. Sadovnichy, V., Tikhonravov, A., Voevodin, V., Opanasenko, V.: “lomonosov”: Supercomputing at Moscow State University. In: Contemporary High Performance Computing: From Petascale toward Exascale. pp. 283–307. Chapman & Hall/CRC Computational Science, Boca Raton, United States, Boca Raton, United States (2013)

2. Schrödinger, LLC: The pymol molecular graphics system, version 1.8 (2015)

3. Pierce, B.G., Wiehe, K., Hwang, H., Kim, B.H., Vreven, T., Weng, Z.: ZDOCK server: interactive docking prediction of protein-protein complexes and symmetric multimers. Bioinformatics 30(12), 1771–1773 (2014), DOI: 10.1093/bioinformatics/btu097

4. Kabsch, W., Sander, C.: Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. Biopolymers 22(12), 2577–2637 (1983), DOI: 10.1002/bip.360221211

5. Reshetnikov, R.V., Stolyarova, A.V., Zalevsky, A.O., Panteleev, D.Y., Pavlova, G.V., Klinov, D.V., Golovin, A.V., Protopopova, A.D.: A coarse-grained model for dna origami. Nucleic Acids Research 46(3), 1102–1112 (2017), DOI: 10.1093/nar/gkx1262

6. Lindorff-Larsen, K., Piana, S., Palmo, K., Maragakis, P., Klepeis, J.L., Dror, R.O., Shaw, D.E.: Improved side-chain torsion potentials for the amber ff99sb protein force field. Proteins: Structure, Function, and Bioinformatics pp. NA–NA (2010), DOI: 10.1002/prot.22711

7. Abraham, M.J., Murtola, T., Schulz, R., Páll, S., Smith, J.C., Hess, B., Lindahl, E.: GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX 1-2, 19–25 (2015), DOI: 10.1016/j.softx.2015.06.001

8. Durek, T., Shelukhina, I.V., Tae, H.S., Thongyoo, P., Spirova, E.N., Kudryavtsev, D.S., Kasheverov, I.E., Faure, G., Corringer, P.J., Craik, D.J., Adams, D.J., Tsetlin, V.I.: Interaction of synthetic human slurp-1 with the nicotinic acetylcholine receptors. Scientific Reports 7(1) (2017), DOI: 10.1038/s41598-017-16809-0

9. Joosten, R.P., te Beek, T.A.H., Krieger, E., Hekkelman, M.L., Hooft, R.W.W., Schneider, R., Sander, C., Vriend, G.: A series of pdb related databases for everyday needs. Nucleic Acids Research 39(Database), D411–D419 (2010), DOI: 10.1093/nar/gkq1105

10. Lyukmanova, E.N., Shulepko, M.A., Buldakova, S.L., Kasheverov, I.E., Shenkarev, Z.O., Reshetnikov, R.V., Filkin, S.Y., Kudryavtsev, D.S., Ojomoko, L.O., Kryukova, E.V., Dolgikh, D.A., Kirpichnikov, M.P., Bregestovski, P.D., Tsetlin, V.I.: Water-soluble lynx1 residues important for interaction with muscle-type and/or neuronal nicotinic receptors. Journal of Biological Chemistry 288(22), 15888–15899 (2013), DOI: 10.1074/jbc.m112.436576

DOI: 10.14529/jsfi180408

Supercomputer Simulations of Fluid-Structure Interaction Problems Using an Immersed Boundary Method

Natalya S. Zhdanova[1], *Andrey V. Gorobets*[1], *Ilya V. Abalakin*[1]

The paper describes a supercomputer application in simulations of fluid-structure interaction problems. A compressible flow solver based on a high-accuracy scheme for unstructured hybrid meshes is considered. It combines an immersed boundary method with a dynamic mesh adaptation method in order to represent motion of solid objects in a turbulent flow. The use of immersed boundaries allows you to dynamically adapt the mesh resolution near moving solid surfaces without changing the mesh topology. Multilevel MPI + OpenMP parallelization of these components fits well with the architecture of modern cluster systems. The proposed implementation can engage thousands of CPU cores in one simulation efficiently. An example application is presented in which a high-speed turbulent flow around a cavity with a deflector is simulated.

Keywords: Parallel CFD, immersed boundary method, unstructured mesh, turbulent flow, MPI+OpenMP.

Introduction

Mathematical modeling of the fluid-structure interaction (FSI) problems is crucial for fluid mechanics engineering, as well as for design of structures and reliability control. At the same time scale-resolving simulation of turbulent flows in dynamic geometrically complex configurations imposes high computing demands.

An immersed boundary condition (IBC) method is used in combination with a dynamic mesh adaptation method in order to reduce computing costs and increase efficiency of simulation as compared to traditional body-fitted approaches. Originally IBC methods were created for modeling of flows around arbitrary-shaped obstacles using structured cartesian grids. In the present work the Brinkman penalization method [1] is used within a high-accuracy compressible flow solver on unstructured hybrid meshes. The dynamic mesh adaptation technique based on variational principles [2] significantly reduces the required number of mesh nodes and improves the accuracy of the solid surface representation.

In contrast to a body-fitted approach, when solid surfaces are represented by exterior mesh faces, the use of IBC allows to easily track a solid surface by adding an external force field. Thanks to the ability of nodes to pass through solid objects, the mesh adaptation becomes much easier, affecting only coordinates of the nodes while the mesh topology remains constant. This makes recalculation of control volumes much cheaper. Furthermore, in this case there is no workload imbalance, which would otherwise appear when adding and removing mesh nodes.

1. Mathematical Model and Numerical method

The mathematical model is based on the compressible Navier–Stokes (NS) equations. At the fluid-solid interface the no-slip condition is imposed using the Brinkman penalization method [1], which does not require matching of the mesh nodes to the solid boundary. The special penalty functions are added as source terms into the NS system. These functions differ from zero only at the mesh nodes inside obstacle. In order to model the FSI problems, the NS system is coupled with the obstacle motion equation.

[1]Keldysh Institute of Applied Mathematics, Russian Academy of Sciences, Moscow, Russian Federation

The NS system is descretized on an unstructured hybrid mesh using the high-accuracy edge-based scheme EBR [3]. This scheme provides high accuracy (up to 5-th order on translationally invariant meshes) at about the same computing cost as a basic low-order scheme.

The time integration is performed using an implicit second-order scheme based on the Newton linearization. The corresponding Jacobi system of linear equations is solved using a preconditioned Bi-CGSTAB solver [4].

The algorithm of the time integration step consists of several stages. Firstly, the motion equation is solved in order to update the coordinates and velocities of the mass centers of the solid bodies. Then the positions of solid surfaces are updated and tracked by the mesh adaptation method. Finally, new penalty functions are determined, and the penalized NS system is solved.

2. Parallel Implementation

The proposed method was implemented in the CFD code NOISEtte [5]. It has MPI+OpenMP parallelization for supercomputers made of multi- and manycore processors. A multilevel mesh partitioning is used for workload distribution among supercomputer nodes and among CPU cores inside nodes. A detailed description of the parallel algorithm can be found in [5].

The immersed boundary penalization method was implemented following the same parallelization approach. The calculation of penalty functions produces no significant load imbalance since it is not computing intensive.

The dynamic mesh adaptation follows MPI+OpenMP parallelization as well, but it introduces notably more complexity to the parallel algorithm. It involves a different parallel iterative solver [6] for a linear system of equations represented with another sparse matrix format.

3. Verification and Validation

Various test cases with available experimental and numerical data were considered in order to validate the accuracy of FSI modeling using Brinkman penalization method coupled with dynamic mesh adaptation.

The oscillations of 2D and 3D obstacles, such as a cylinder or a sphere, caused either by a given external force or by vortex-induced forces were investigated. The computational results are in a good agreement with the reference data.

A flow around a flapping foil represents more complex geometry and motion law. In this test case the NACA0012 foil oscillates with plunging and pitching mechanisms. The instantaneous vorticity magnitude for different phase angles is shown in Fig. 1. An example of the automatic mesh adaptation that tracks the surface of the foil is shown in Fig. 2.

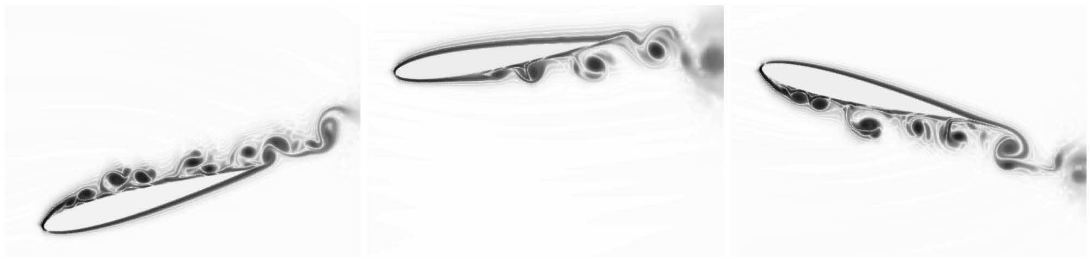


Figure 1. Instantaneous vorticity magnitude for different phase angles of the flapping foil

A good agreement between numerical and experimental results was observed in comparison of the measured and calculated mean thrust and power coefficients. Further details on comparison with experimental data and body-fitted numerical results can be found in [7].

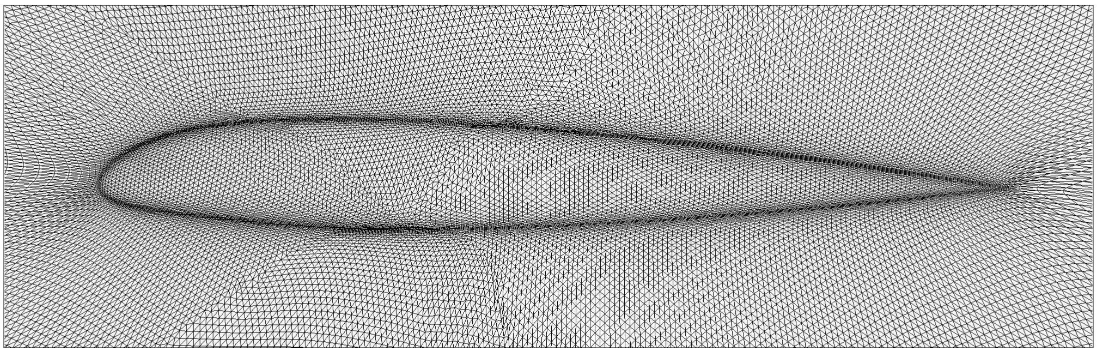
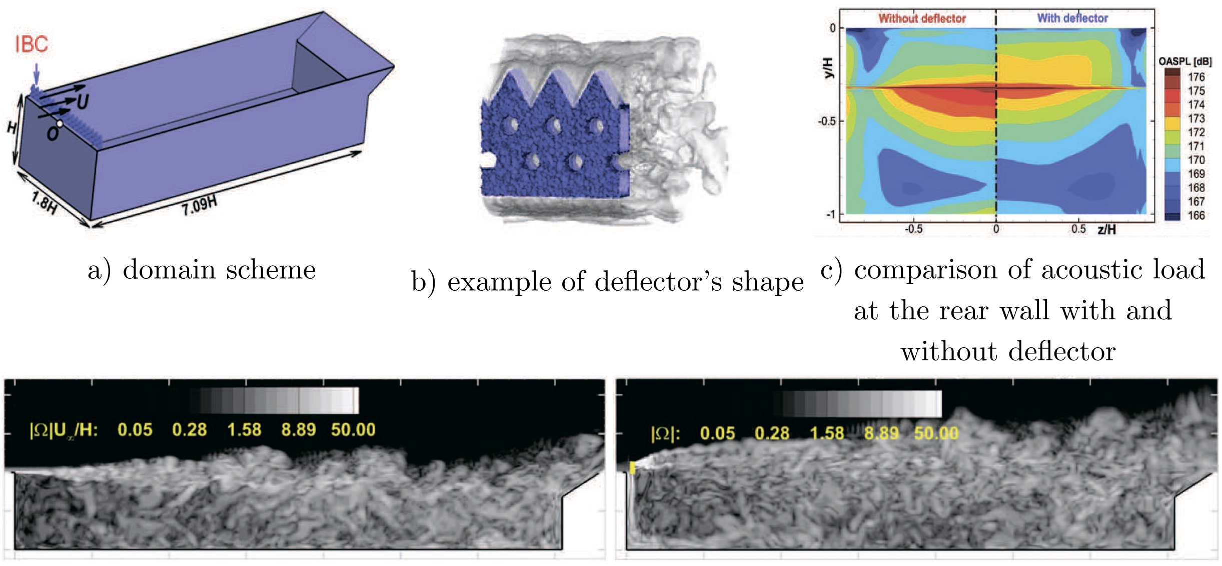


Figure 2. Example of automatic mesh adaptation that tracks motion of the foil

4. Industry-Oriented Applications

The immersed boundary method can be efficiently combined with a body-fitted approach. Static objects that require accurate boundary layer resolution can be represented with a body-fitted mesh. At the same time, moving objects or static objects that allow less accurate resolution can be represented with IBC. For instance, if we study a helicopter main rotor using a body-fitted mesh and want to account for effect of the helicopter fuselage on the flow, then we can simulate the fuselage using IBC. Deployment of spoilers on a wing, opening doors of a gear bay, deployment of landing gears, release of a payload, etc., can be simulated with this combined approach.

The proposed method was applied in a simulation of a flow over a deflector mounted at the upstream side of a cavity (Fig. 3).



a) domain scheme

b) example of deflector's shape

c) comparison of acoustic load at the rear wall with and without deflector

d) instantaneous turbulent flow fields without deflector

e) instantaneous turbulent flow fields with deflector

Figure 3. Flow over a cavity with a deflector

The flow parameters correspond to a flight regime of a high-speed aerial transport vehicle. The aim of the study was to reduce acoustic and vibration load on the surface of the cavity by improving the shape of the deflector. Multiple configurations of the complex shape deflector were numerically studied using IBC. The use of IBC allowed us to easily change the shape with a given surface parametrization, and to use one spatial mesh for all the configurations. The computational results demonstrated that the deflector considerably reduces the mean pressure coefficient and the overall pressure fluctuation level at the cavity bottom and on its rear wall. Further details on this study can be found in [8]. IBC applicability tests were performed for static meshes with up to one billion elements using up to 10,240 CPU cores of Lomonosov supercomputer with parallel efficiency of doubling the number of cores above 90%.

Conclusions

The developed approach can be applied for supercomputer modeling of FSI problems on unstructured meshes. The immersed boundary method allows to efficiently handle moving obstacles. Parallel dynamic mesh adaptation provides it with necessary resolution near solid surfaces. All the components fit well with MPI+OpenMP parallelization and can be easily implemented in the existing parallel CFD codes.

Acknowledgments

The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University supported by project RFMEFI62117X0011. The work was partially funded by the Council on grants of the President of Russia, project MD-5968.2018.1 (reduction of computing costs using mesh adaptation).

References

1. Angot, Ph., Bruneau, C-H., Fabrie, P.: A penalization method to take into account obstacles in incompressible viscous flows. Numerical Mathematics and Scientific Computation 81(4), 497–520 (1999), DOI: 10.1007/s002110050401

2. Garanzha, V.A., Kudryavtseva L.N., Utyuzhnikov S.: Variational method for untangling and optimization of spatial meshes. Journal of Computational and Applied Mathematics 269, 24–41 (2014), DOI: 10.1016/j.cam.2014.03.006

3. Abalakin, I., Bakhvalov, P., Kozubskaya, T.: Edge-based reconstruction schemes for unstructured tetrahedral meshes. International Journal for Numerical Methods in Fluids 81(6), 331–356 (2016), DOI: 10.1002/fld.4187

4. Van der Vorst, H.A.: Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. SIAM Journal on Scientific and Statistical Computing 13(2), 631–644 (1992), DOI: 10.1137/0913035

5. Gorobets, A.V.: Parallel Algorithm of the NOISEtte Code for CFD and CAA Simulations. Lobachevskii Journal of Mathematics 39(4), 524–532 (2018), DOI: 10.1134/S1995080218040078

6. Kaporin, I.E., Milyukova, O.Yu.: MPI+OpenMP parallel implementation of explicitly preconditioned conjugate gradient method. KIAM Preprint #8, Moscow, 2018, DOI: 10.20948/prepr-2018-8

7. Abalakin, I., Zhdanova, N., Kozubskaya, T.: Immersed boundary penalty method for compressible flows over moving obstacles. Proceedings of the 6t-h European Conference on Computational Mechanics, ECCOMAS ECCM6, Glasgow, UK, June 11–15, 2018, http://www.eccm-ecfd2018.org/admin/files/filePaper/p1797.pdf, accessed: 2018-12-21

8. Duben' A. P., Zhdanova N. S., Kozubskaya T. K.: Numerical investigation of the deflector effect on the aerodynamic and acoustic characteristics of turbulent cavity flow. Fluid Dynamics 52(4), 561–571 (2017), DOI: 10.1134/s001546281704010x

DOI: 10.14529/jsfi180409

Test of Computational Approaches for Gold-Thiolate Clusters Calculation using Lomonosov Supercomputer

Nadezhda N. Nikitina[1], *Daria A. Pichugina*[1], *Alexander V. Oleynichenko*[1], *Oxana N. Ryzhova*[1], *Kirill E. Kopylov*[1], *Vladimir V. Krotov*[1], *Nikolay E. Kuzmenko*[1]

High-level procedures (MP2, CCSD, CCSD(T)) and reliable experimental data have been used to assess the performance of a variety of exchange-correlation functionals for the calculation of structures and energies of small models of thiolate-protected gold clusters. Clusters represent rather complicated objects for examination, therefore the simple models including Au_2, AuS were considered to find an appropriate method to calculate Au-Au and Au-S interactions in protected clusters. The mean unsigned errors of the quantum chemical methods were evaluated via reliable experimental bond distances and dissociation energies of Au_2 and AuS. Based on the calculation, the SVWN5, TPSS+D3, PBE96+D3, and PBE0+D3 were found to give the most reliable results and can be recommended for calculation of the structure and properties of thiolate-protected gold clusters. The influence of the relativistic corrections calculated in Dirac-Coulomb-Breit framework and inclusion of dispersion corrections on the structure and energy of thiolate-protected gold clusters have been analyzed.

Keywords: density functional theory, parallel calculation, cluster, gold, dispersion correction, relativistic effects, computational chemistry.

Introduction

Supercomputer simulation based on quantum chemical methods is effectively applied to predict and examine the structure and properties of different compounds including nanoclusters, materials, metal-organic systems, and drugs [1, 2]. Thiolate-protected gold clusters, $Au_n(SR)_m$, are popular objects in theoretical and nano science. The quantum chemical study of clusters has a number of challenges: *(i)* To describe Au-Au and Au-S interactions correctly, the approach should take into account relativistic effects, dispersion interactions, and others features; *(ii)* High-level *ab initio* procedures are not feasible for $Au_n(SR)_m$, since they become very resource-intensive; *(iii)* Application of the $Au_n(SR)_m$ data in solid or liquid phases as a benchmark of theoretical methods is difficult due to that a majority of quantum chemical methods are related to a gas phase. The problem of an extensive global search of potential energy surface to locate global minima can be solved by the genetic algorithm with DFT calculation [3].

We propose new insights to choose the quantum chemical protocol of calculation of gold protected clusters and similar metal-organic systems performing benchmark study of the simple fragments of cluster. Clusters represent rather complicated objects for examination, therefore the simple models including Au_2, AuS were considered.

1. Calculation Details

Calculated interatomic distances (R_e) and bond energies (D_0) of the fragments were compared to available experimental spectroscopic data [4, 5] and values obtained by high-level *ab initio* methods (MP2, CCSD, CCSD(T)) to assess the performance of exchange-correlation functional (LSDA, GGA, meta-GGA, global-hybrid GGA, global-hybrid GGA, global-hybrid meta-

[1]Department of Chemistry, M.V. Lomonosov Moscow State University, Moscow, Russian Federation

GGA, range-separated hybrid GGA functionals). Effective core potential formalism using the cc-pVDZ/cc-pVDZ-PP [6, 7], 6-31G*/SBKJC [8], 6-31G*/LANL2DZ [9] (further referred to BS-1, BS-2, and BS-3) was applied.

The Dirac-Coulomb-Breit Hamiltonian (DCB) with separate spin-free and spin-dependent components [10] was employed. The energy-optimized extended Gaussian basis set of triple-polarized quality of the large component, and the corresponding kinetically balanced basis for the small component [11] was used (BS-4). The dispersion corrections with the Becke-Johnson damping (DFT+D3) [12] was also calculated.

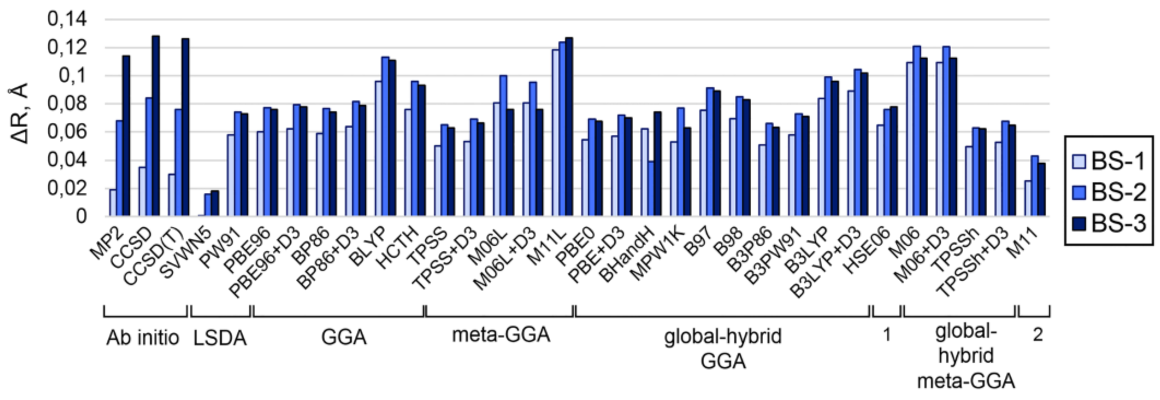
For each protocol mean unsigned errors (MUEs) were established. The DFT MUEs were compared with the MP2, CCSD, and CCSD(T) MUEs and divided into three groups: *(i)* accurate procedure if MUE is less than CCSD(T)/BS-1 MUE; *(ii)* less accurate procedure if MUE is in the range between CCSD(T)/BS-1 and CCSD(T)/BS-2; and *(iii)* erroneous procedure if MUE is more than MUE of CCSD(T)/BS-2.

All calculations were performed with NWChem [13] and PRIRODA [14] codes. Server 1: HP DL160G6 server with 2 Intel Xeon E5504 (quad-core, 2.0 GHz) processors, 24 GiB DDR3 ECC RAM, 1.8 TiB HP P410/512MB based RAID 1+0 of 4 Seagate ST1000NM0033-9ZM disks. Servers 2 and 3: virtual machines with 8 processor cores (2.53 GHz), 24 GiB RAM and 300 GB VHDD, running on VMware vSphere 5.5 platform (M.V. Lomonosov Moscow State University Datacenter). All servers are running CentOS 6.7 Linux 64-bit, NWChem 6.6 64-bit built with Intel Compilers, MKL and Intel MPI from Intel Parallel Studio 2016 Cluster Edition). The calculation capacity offered by Supercomputing Center of M.V. Lomonosov Moscow State University has been also used [15].

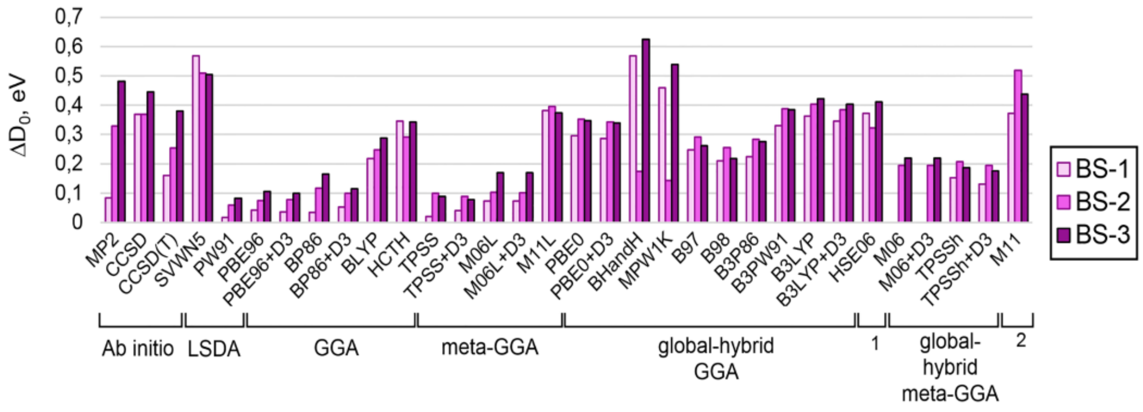
2. Results and Discussion

The calculated R_e values of Au_2 molecule are compared to the experimental data 2.4719 [4]; the MUEs are shown on Fig. 1. Evidently, CCSD(T)/BS-1 and MP2/BS-1 protocols give the most accurate Au-Au distance among*ab initio* methods. For all *ab initio* methods, the MUEs increase in the order BS-1 <BS-2 <BS-3. Thus, the expanding of a basis set slightly improve the distance calculated at CCSD(T) level only. In contrast, the Re calculated by different functionals slightly varies from the basis sets. The first group of the high accurate functionals contains SVWN5/BS-1, SVWN5/BS-2, SVWN5/BS-3, and M11/BS-1. The homogeneous electron gas approximation to XC energy could be provided the advance of SVWN5. The second group is represented by PW91, PBE96, BP86, TPSS, PBE0, BHandH, MPW1K, B3P86, B3PW91, HSE06, TPSSh. Other functionals including BLYP, B3LYP, M06, M06L, M11L have high MUEs, more than 0.08 , and failed to describe the experimental Au-Au distance. It should be noted that the majority of protocols tend to overestimate R_e in Au_2, except MP2/BS-1 and SVWN5/BS-1. Dispersion corrections slightly effect the R_e. The Au-Au distance calculated by DCB framework is equal to 2.530 ; MUE=0.058 .

Among *ab initio* methods, CCSD(T)/BS-1 and MP2/BS-1 successfully predict the dissociation energy of Au_2 (Fig. 1). The errors of MP2/Def2-TZVPP and CCSD(T)/Def2-TZVPP are large and equal to 0.13 eV and 0.23 eV, respectively. Some of the functionals have less MUEs than *ab initio* methods. The PW91, PBE96, BP86, TPSS, M06L form the first group of accurate methods (MUE ¡ 0.1 eV). The SVWN5 and BHandH functionals, which have exactly predicted R_e, calculate $D_0(Au_2)$ with large MUE, as well as HCTH, B3PW91, B3LYP, HSE06, M11, M06.



a



b

Figure 1. Mean unsigned errors in calculation of ΔR (A) and ΔD_0 (B) of Au_2 (a – range-separated hybrid GGA functional; b – range-separated hybrid meta-GGA functional)

The dispersion corrections slightly influence the results. Relativistic DCB dissociation energy (2.30 eV) is completely in line with the experimental ones.

The equilibrium distance and dissociation energy of AuS are calculated using CCSD, CCSD(T), MP2, DFT, DCB. The experimental value of R_e, 2.156 [5] was used as a benchmark. MP2/BS-1 accurately predicts the Au-S distance among the considered *ab initio* methods. For MP2, CCSD, CCSD(T) the error increases in the order BS-1 <BS-2 <BS-3. The Au-S distance calculated using DFT also depends on the basis set and a functional type. The first group of accurate procedure (with MUEs less than 0.06) contains all considered functionals with BS-1 excluding BLYP, HCTH, M11L, BHandH, B97, B98, B3P86, B3LYP, M06. The dispersion and relativistic corrections slightly influence R_e value. For instance, the Au-S distance calculated in DCB approximation has the same error (0.07) as the first group of the DFT methods.

All *ab initio* methods accurately predict a dissociation energy of AuS excluding MP2/BS-2 procedure (MUE = 0.41 eV). The errors of MP2/Def2-TZVPP and CCSD(T)/Def2-TZVPP are larger than BS-1 and equal to 0.56 eV and 0.63 eV, respectively. Among DFT, MPW1K/BS-1 protocol calculates D0 more accurate than CCSD(T)/BS-1. For PBE96+D3, BP86+D3, HCTH, TPSS+D3, M06L, M06L+D3, M11L, PBE0, PBE0+D3, B97, B98, B3P86, B3PW91, B3LYP, B3LYP+D3, HSE06, M06, M06+D3, TPSSh, TPSSh+D3 with Lanl2DZ, MUEs are less then MUEs obtained by SBK and cc-pVDZ basis sets. It should be noted that PBE96/BS-

3, TPSS/BS-1, M11L/BS-3, PBE0/BS-3, MPW1K/BS-1, B98/BS-3, B3LYP/BS-3, M06/BS-3 could be recommended for AuS calculations. The dispersion corrections improve the calculated value in most cases excluding M06. Dissociation energy calculated by DCB approximation is 2.67 eV (MUS=0.08 eV) and in line with the experimental data.

The calculation of Au-Au and Au-S distances in Au_2 and AuS reveals the functionals which are close in accuracy to high-level *ab initio* procedures and experimental data. Due to SVWN5, B3PW91, PBE96, PBE0, B3LYP, TPSS, TPSSh, M06, M06L, M11, M11L, PW91, BP86 procedure have the least errors in the predicted properties of Au_2 and AuS, they are further applied to benchmark calculation of the cyclic gold-thiolate complexes.

Finally, the $Au_{20}(SR)_{16}$ structure is calculated by SVWN5, PBE96, TPSS, PBE96+D3, TPSS+D3, PBE0, PBE0+D3 methods using BS-2, BS-3 basis set and by DCB framework with BS-4 basis set to understand the accuracy and time of the DFT procedure to predict the interatomic distances in real thiol protected gold cluster. The main interatomic distances and average calculation time of one optimization step of calculation are collected in Tab. 1.

Table 1. The main interatomic distances () in $Au_{20}(SR)_{16}$ and average calculation time of one optimization step (t, min)

Method	Basis set	Au(1)-Au(2)	Au(2)-Au(3)	Au(3)-Au(4)	Au(2)-Au(5)	t
X-ray	-	2.717	2.870	2.982	3.120	-
SVWN5	BS-2	2.720	2.854	2.795	2.996	95
	BS-3	2.727	2.880	2.860	3.013	61
TPSS	BS-2	2.776	2.883	2.960	3.090	76
	BS-3	2.774	2.899	2.959	3.150	55
TPSS+D3	BS-2	2.750	2.849	2.836	2.965	83
	BS-3	2.740	2.872	2.845	3.013	67
PBE96	BS-2	2.810	2.967	3.123	3.232	79
	BS-3	2.801	2.944	3.034	3.256	60
PBE96+D3	BS-2	2.774	2.830	2.905	3.120	86
	BS-3	2.771	2.898	2.946	3.153	64
PBE0	BS-2	2.792	2.901	2.991	3.081	75
	BS-3	2.765	2.909	2.968	3.142	63
PBE0+D3	BS-2	2.772	2.894	2.951	3.098	78
	BS-3	2.749	2.882	2.934	3.156	69
PBE96+DCB	BS-4	2.810	2.952	3.063	3.204	142

The structure obtained at TPSS/BS-2 method proved to be more close to the experimental one than the structure predicted in PBE96/BS-2 and SVWN5/BS-2 approaches. It should be noted that dispersion corrections improve convergence of the calculated interatomic distances and the experimental ones using the PBE96 functional. The account of relativistic effects in DCB framework also improves the geometrical parameters, but the average calculation time of one optimization step is sufficiently longer than time of PBE96/BS-2 and PBE96/BS-3 protocols.

Conclusions

The structures and energies of small models of thiolate-protected gold clusters were calculated by ab initio methods (MP2, CCSD, CCSD(T)) and different exchange-correlation functionals including dispersion and relativistic corrections. To find the accurate method for the study of Au-Au and Au-S interactions, the diatomic Au_2 and AuS molecules were considered.

All calculated Au-Au bonds in the $Au_{20}(SR)_{16}$ are slightly larger than the corresponding values in crystal structure. The account of dispersion corrections and relativistic effects improves the geometrical parameterizes. So, SVWN5, TPSS+D3, PBE96+D3, PBE0+D3 using BS-2 and BS-3 could be recommended for $Au_n(SR)_m$ calculation, because they describe Au-Au and Au-S interactions more accurately.

The obtained results illustrated the complications of $Au_n(SR)_m$ for theoretical investigation and provide new information to theorists and chemists studying structure and properties of protected gold clusters or other complicated chemical systems including self-assembled monolayers, ligand-protected metal clusters, and organometallic complexes.

Acknowledgments

The present work was supported by RFBR Grant N 17-03-00962. The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University supported by the project RFMEFI62117X0011

References

1. Polyakov, I.V., Moskovsky, A.A., Nemukhin, A.V.: Multi-Scale Supercomputing of Large Molecular Aggregates: A Case Study of the Light-Harvesting Photosynthetic Center. Supercomputing Frontiers and Innovations 2, 48–54 (2015), DOI: 10.14529/jsfi150403

2. Usher, W., Wald, I., Knoll, A., Papka, M., Pascucci, V.: In Situ Exploration of Particle Simulations with CPU Ray Tracing. Supercomputing Frontiers and Innovations 3, 4–18 (2016), DOI: 10.14529/jsfi160401

3. Liu, Y., Tian, Z., Cheng, L.: Size evolution and ligand effects on the structures and stability of (AuL)n (L = Cl, SH, SCH3, PH2, P(CH3)2, n = 113) clusters. RSC Advances. 6, 4705–4712 (2016), DOI: 10.1039/C5RA22741K

4. Bishea, G.A., Morse, M.D.: Spectroscopic studies of jetcooled AgAu and Au_2. The Journal of Chemical Physics 95, 5646–5659 (1991), DOI: 10.1063/1.461639

5. Kokkin, D.L., Zhang, R., Steimle, T.C.: AuS Bonding Revealed from the Characterization of Diatomic Gold Sulfide, AuS. TheJournal of Physical Chemistry A A 119, 11659–11667 (2015), DOI: 10.1021/acs.jpca.5b08781

6. Dunning Jr.T.H.: Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen. The Journal of Chemical Physics 90, 1007–1023 (1989), DOI: 10.1063/1.456153

7. Peterson, K.A., Puzzarini, C.: Systematically convergent basis sets for transition metals. II. Pseudopotential-based correlation consistent basis sets for the group 11 (Cu, Ag, Au) and 12 (Zn, Cd, Hg) elements. Theoretical Chemistry Accounts 114, 283–296 (2005), DOI: 10.1007/s00214-005-0681-9

8. Stevens, W.J., Krauss, M., Basch, H., Jasien, P.G.: Relativistic compact effective potentials and efficient, shared-exponent basis sets for the third-, fourth-, and fifth-row atoms. Canadian Journal of Chemistry 70, 612–630 (1992), DOI: 10.1139/v92-085

9. Hay, P.J., Wadt, W.R.: Ab initio effective core potentials for molecular calculations. Potentials for the transition metal atoms Sc to Hg. The Journal of Chemical Physics 82, 270–283 (1985), DOI: 10.1063/1.448799

10. Dyall, K.G.: An exact separation of the spinfree and spindependent terms of the Dirac-CoulombBreit Hamiltonian. The Journal of Chemical Physics 100, 2118–2127 (1994), DOI: 10.1063/1.466508

11. Laikov, D.: A new class of atomic basis functions for accurate electronic structure calculations of molecules. Chemical Physics Letters 416, 116–120 (2005), DOI: 10.1016/j.cplett.2005.09.046

12. Grimme, S., Antony, J., Ehrlich, S., Krieg H.: A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu. The Journal of Chemical Physics 132, 154101(1–19) (2010), DOI: 10.1063/1.3382344

13. Valiev, M., Bylaska, E.J., Govind, N., Kowalski, K., Straatsma, T.P., Van Dam, H.J.J., Wang, D., Nieplocha, J., Apra, E., Windus, T.L., De Jong, W.A.: NWChem: A comprehensive and scalable open-source solution for large scale molecular simulations. Computer Physics Communications 181, 1477–1489 (2010), DOI: 10.1016/j.cpc.2010.04.018

14. Laikov, D.N.: PRIRODA-04: a quantum-chemical program suite. New possibilities in the study of molecular systems with the application of parallel computing. Russian Chemical Bulletin, International Edition 54, 820–826 (2005), DOI: 10.1007/s11172-005-0329-x

15. Sadovnichy, V., Tikhonravov, A., Voevodin, Vl., Opanasenko, V.: Lomonosov: Supercomputing at Moscow State University. In Contemporary High Performance Computing: From Petascale toward Exascale (Chapman & Hall/CRC Computational Science), Boca Raton, USA, CRC Press. 283–307 (2013)

DOI: 10.14529/jsfi180410

Supercomputer Modeling of Dual-Site Acetylcholinesterase (AChE) Inhibition

***Sofya V. Lushchekina*[1], *Galina F. Makhaeva*[2], *Dana A. Novichkova*[1,3], *Irina V. Zueva*[4], *Nadezhda V. Kovaleva*[2], *Rudy J. Richardson*[5]**

Molecular docking is one of the most popular tools of molecular modeling. However, in certain cases, like development of inhibitors of cholinesterases as therapeutic agents for Alzheimer's disease, there are many aspects, which should be taken into account to achieve accurate docking results. For simple molecular docking with popular software and standard protocols, a personal computer is sufficient, however quite often the results are irrelevant. Due to the complex biochemistry and biophysics of cholinesterases, computational research should be supported with quantum mechanics (QM) and molecular dynamics (MD) calculations, what requires the use of supercomputers. Experimental studies of inhibition kinetics can discriminate between different types of inhibition—competitive, non-competitive or mixed type—that is quite helpful for assessment of the docking results. Here we consider inhibition of human acetylcholinesterase (AChE) by the conjugate of MB and 2,8-dimethyl-tetrahydro-γ-carboline, study its interactions with AChE in relation to the experimental data, and use it as an example to elucidate crucial points for reliable docking studies of bulky AChE inhibitors. Molecular docking results were found to be extremely sensitive to the choice of the X-ray AChE structure for the docking target and the scheme selected for the distribution of partial atomic charges. It was demonstrated that flexible docking should be used with an additional caution, because certain protein conformational changes might not correspond with available X-ray and MD data.

Keywords: acetylcholinesterase, Alzheimer's disease, molecular docking, atomic charges.

Introduction

Therapy of Alzheimer's disease (AD) involves inhibition of brain AChE to restore acetylcholine (ACh) levels. [9]. In addition to hydrolyzing ACh, AChE promotes aggregation of β-amyloid peptide through its interaction with the AChE peripheral anionic site (PAS). Thus, dual-site inhibitors of the active and PAS sites are expected to be disease-modifying agents [10].

To develop dual-site anti-AD drugs, we combined two known pharmacophores, methylene blue (MB) and carbolines into single conjugates (**MBC**) [14], see Fig. 1, and demonstrated that they were effective inhibitors of AChE capable of displacing propidium from the AChE PAS [1].

Docking and other computational methods have been used in drug design for decades. However biophysical constraints can hamper the predictive power of these approaches [2]. For example, AChE contains a gorge with a midpoint constriction ("bottleneck") that separates the PAS and active site regions [15]. Consequently, inhibition is determined not only by geometric and interaction energy factors, but also by binding dynamics [7]. In the present work, we analyzed the results of different molecular docking approaches for **MBC** into AChE and compared them to kinetic data, demonstrating mixed-type inhibition (Fig. 2). Thus, the compound should bind competetively to the active site and noncompetitively to the PAS, and docking should provide poses of the ligand both above and below the bottleneck.

[1]Emanuel Institute of Biochemical Physics, Russian Academy of Sciences, Moscow, Russia
[2]Institute of Physiologically Active Compounds, Russian Academy of Sciences, Chernogolovka, Russia
[3]Lomonosov Moscow State University, Moscow, Russian Federation
[4]Arbuzov Institute of Organic and Physical Chemistry, FRC Kazan Scientific Center of RAS,Kazan, Russia
[5]Departments of Environmental Health Sciences and Neurology, University of Michigan, Ann Arbor, USA

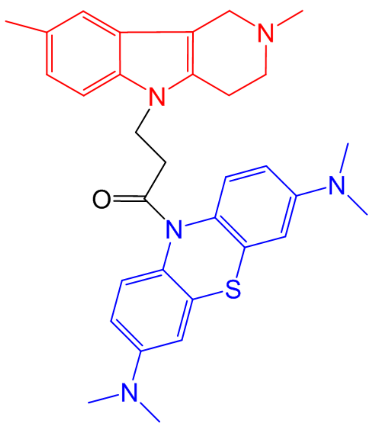


Figure 1. Structure of compound **MBC**, a conjugate of MB (colored blue) and 2,8-dimethyl-tetrahydro-γ-carboline (colored red)

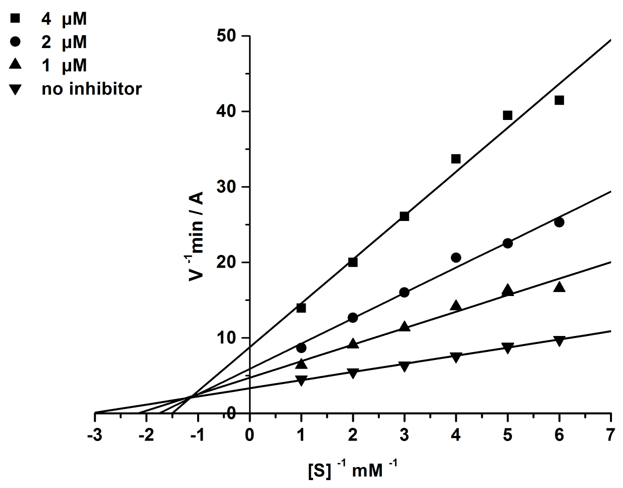
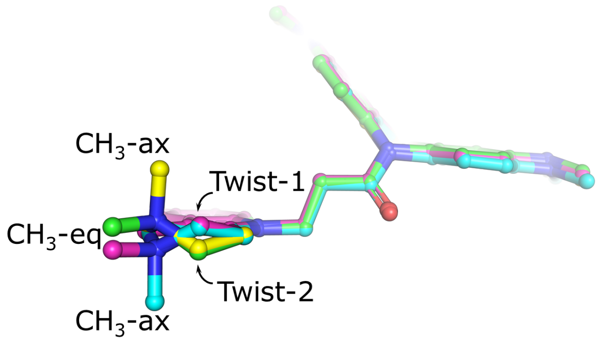


Figure 2. Steady state inhibition of AChE by this compound; Lineweaver-Burk double-reciprocal plots of initial velocity and substrate concentrations in the presence of the inhibitor, showing mixed-type inhibition

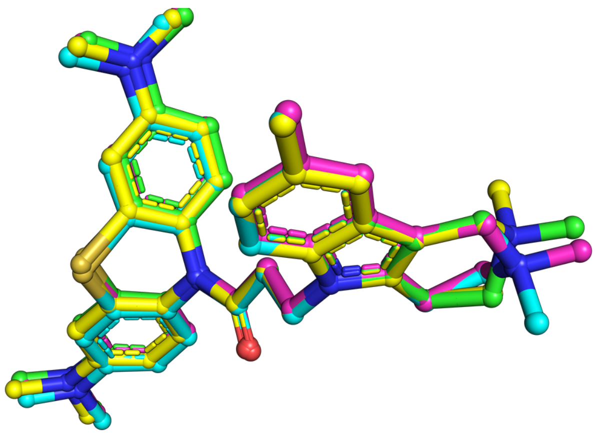
1. Methods

The carboline part of **MBC** contains a piperidine ring condensed with an aromatic system that implicates conformers and enantiomers. Using OpenEye OMEGA 2.5.1.4: OpenEye Scientific Software, Santa Fe, NM. `http://www.eyesopen.com` [5], 4 configurations of **MBC** were generated (Fig. 3).

pK_a values were calculated with Schrödinger Jaguar QM DFT pK_a module [16]. Geometries were optimized with Gamess-US [11] software (B3LYP/6-31G*). For docking, optimized ligand structures were used with Gasteiger partial atomic charges and those derived from QM results according to Mulliken and Löwdin schemes. Additionally, Schrödinger QM-Polarized Ligand Docking (PLD) [4] was used with extra precision docking and redocking; charges were calculated using the Jaguar accurate QM method. Five X-ray structures of human AChE (PDB IDs 4EY4-4EY8, [3]) were used for docking. Rigid docking was performed with AutoDock 4.2.6 [8] as described earlier [12]. For flexible docking, Schrödinger Glide Induced Fit [13] was used with AChE as a target. The docking volume included the entire gorge, and extra precision docking



(a) The carboline ring is perpendicular to the viewing plane



(b) A general side view

Figure 3. Overlaid configurations of the piperidine fragment of the γ-carboline ring of **MBC**

and scoring were employed. MD simulations were done in 0.15 M NaCl solution with previously published methods [6, 17].

2. Results

The PDB (`http://www.rcsb.org`) contains X-ray structures of *apo*-AChE (4EY4) and co-crystallized with different compounds: (-)-Huperzine A (4EY5); (-)-Galantamine (4EY6); Donepezil (4EY7); and fasciculin-2 (4EY8) [3]. The absence or presence of ligands affects the conformation of principal amino acids lining the gorge, e. g., Tyr337 and Tyr341 [15], see Fig. 4. We have previously reported significant differences in estimated binding energies for the same compounds with these targets [12]. Here, we show that the target X-ray structure determines whether or not ligand poses reflect mixed-type inhibition.

The QM-calculated pK_a value for the piperidine nitrogen was 7.84. Thus, under experimental conditions mimicking physiological pH 7.4, both protonated and non-protonated forms could be present. For this reason, both states were used for the docking study and analysis of results.

Partial atomic charges are crucial for docking results from the algorithms used in our study, as the charge distribution calculation scheme defines the estimated binding energies and geometries of complexes [4]. With respect to **MBC** docking into AChE, the influence of partial atomic charges was even more pronounced. In the case of *apo*-AChE as a target, poses below the bottleneck were obtained only for structures with partial charges derived from QM calculations according to the Löwdin scheme (Fig. 5). The results obtained with the Gasteiger scheme and derived from QM data according to the Mulliken scheme and Shrödinger QM PLD docking showed poorer occupation of the active site compartment for other targets (Fig. 5).

Only in the case of the AChE structure co-crystallized with Donepezil was **MBC** docked in full correspondence with experimental data (below and above the bottleneck) regardless of the partial atomic charges scheme (Fig. 5). This is ensured by the Tyr337 side chain, which forms the bottleneck, being rotated so that it does not block the gorge.

For AChE structures co-crystallized with Huperzine A and fasciculin-2 (Fig. 6), the **MBC** ligand could be found only in the PAS, which corresponds to non-competitive inhibition, and thus does not agree with experimental data.

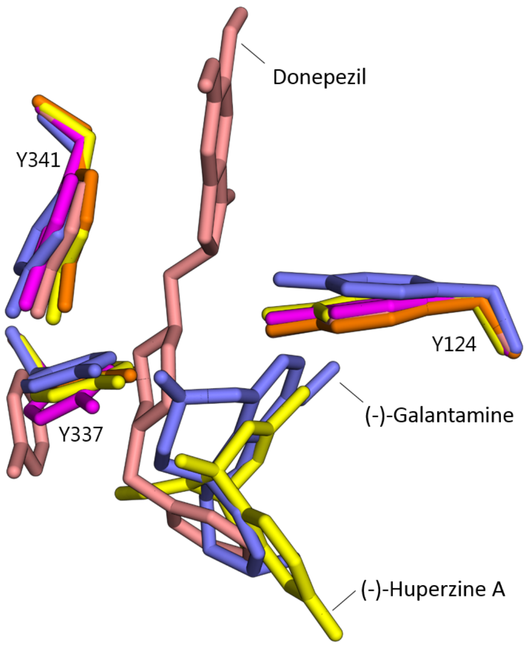


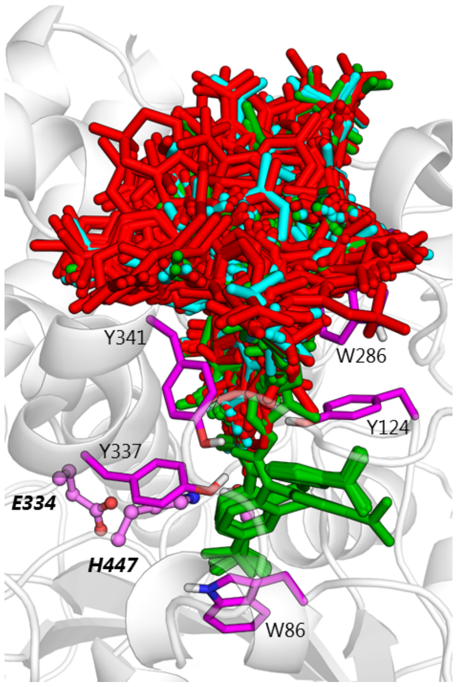
Figure 4. Overlay of X-ray structures of *apo*-state AChE (magenta); or AChE co-crystallised with (-)-Huperzine A (yellow), (-)-Galantamine (blue), Donepezil (salmon), and fasciculin-2 (orange). Principal amino acids of the gorge (Tyr341, Tyr337 and Tyr24) and ligands are shown

The Schrödinger Glide Induced Fit protocol for molecular docking of **MBC** provided positions similar to those obtained by rigid docking to 4EY7 as a target. The major difference in the compound's position was a flipped MB fragment, achieved through appreciable displacement of Phe297 and Tyr124, while conformational changes for other principal residues of the gorge were less significant (Fig. 7).

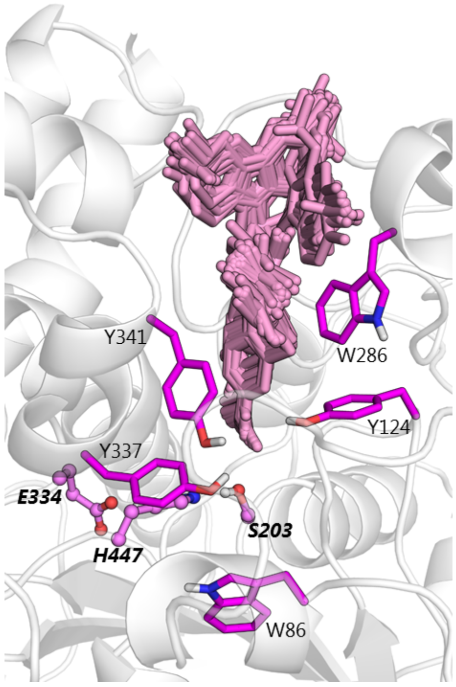
Conformations of Phe297 and Tyr124 side chains, namely, the χ_1 torsion angle for induced fit docking complexes, could be compared with conformations found in X-ray structures of human and mouse AChE and along MD trajectories for *apo*-AChE and AChE in complex with another bulky inhibitor [6]. We found side chain conformations from flexible docking different from those found in X-ray data or during MD simulations even with a bulky inhibitor in the gorge (Fig. 8). This suggests that results of the Induced Fit protocol of Glide should be compared with other available data and certain torsion angles should be fixed for redocking.

Conclusions

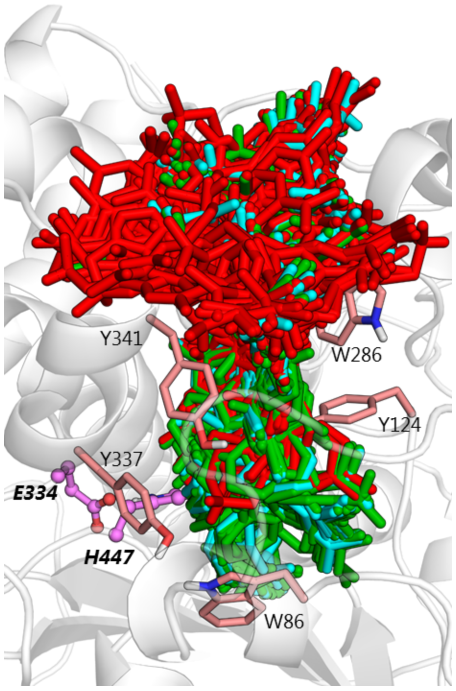
The results of kinetic and docking studies demonstrate the importance of choosing the right target structures. For bulky ligands, the structure of AChE co-crystallized with Donepezil (4EY7) gave the best agreement with experimental data. The use of different partial atomic charges also leads to markedly different docking results; the use of charges derived from QM calculations is advisable. Induced-fit docking should be used with caution; conformational changes of protein residues should be related to protein dynamics data (X-ray and MD) to avoid artifacts. Overall, to achieve reliable results, docking studies require the support of computationally demanding QM and MD calculations, as afforded by supercomputing facilities.



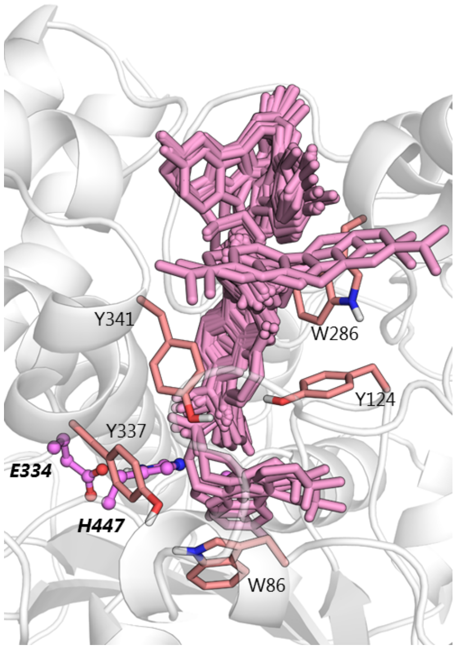
(a) Target 4EY4, Gasteiger/Mulliken/Löwdin



(b) Target 4EY4,QM-PLD

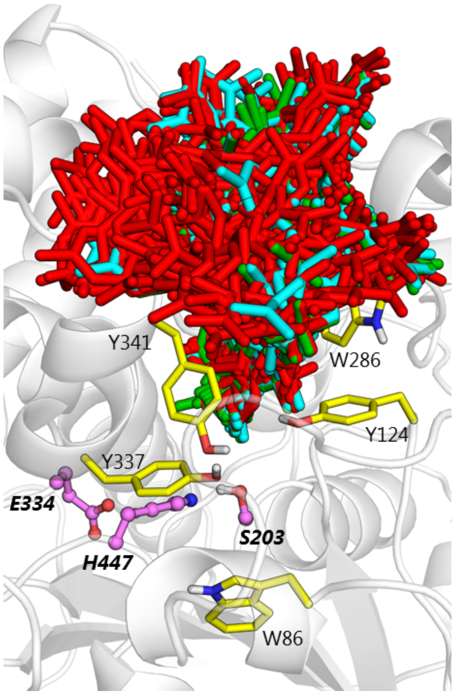


(e) Target 4EY7, Gasteiger/Mulliken/Löwdin

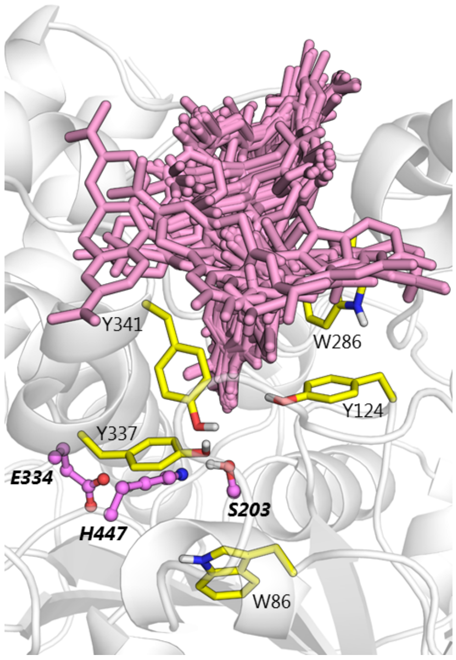


(f) Target 4EY7,QM-PLD

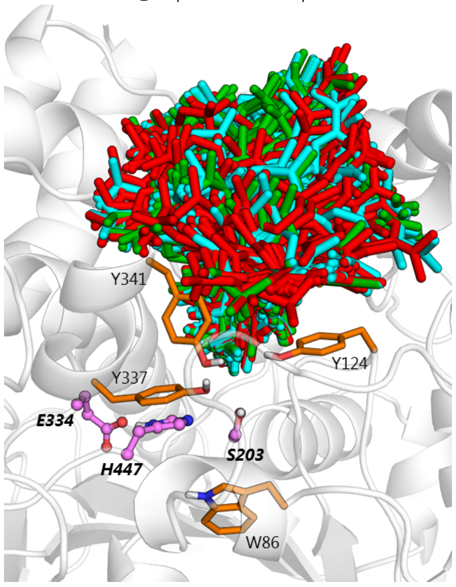
Figure 5. Molecular docking results of conjugate **MBC** into AChE, corresponding to experimental data. Carbon atoms of the target AChE amino acids are colored according to Fig. 4; catalytic residues are colored violet. In the left columns, cyan color shows poses obtained with partial atomic charges derived from the Gasteiger scheme, red—derived from QM calculations according to the Mulliken scheme, and green—derived from QM calculations according to the Löwdin scheme. Results of Schrödinger QM-PLD for each X-ray AChE structure are shown separately in the right column—ligand poses are colored pink



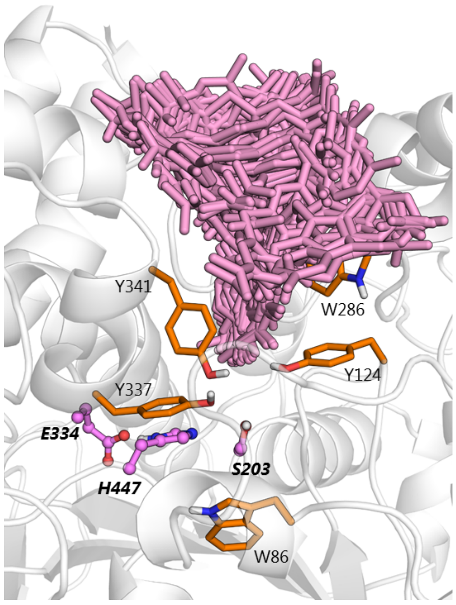
(c) Target 4EY5, Gasteiger/Mulliken/Löwdin



(d) Target 4EY5,QM-PLD



(g) Target 4EY8, Gasteiger/Mulliken/Löwdin



(h) Target 4EY8,QM-PLD

Figure 6. Molecular docking results of conjugate **MBC** into AChE, not reflecting experimental data. Coloring according to Fig. 5

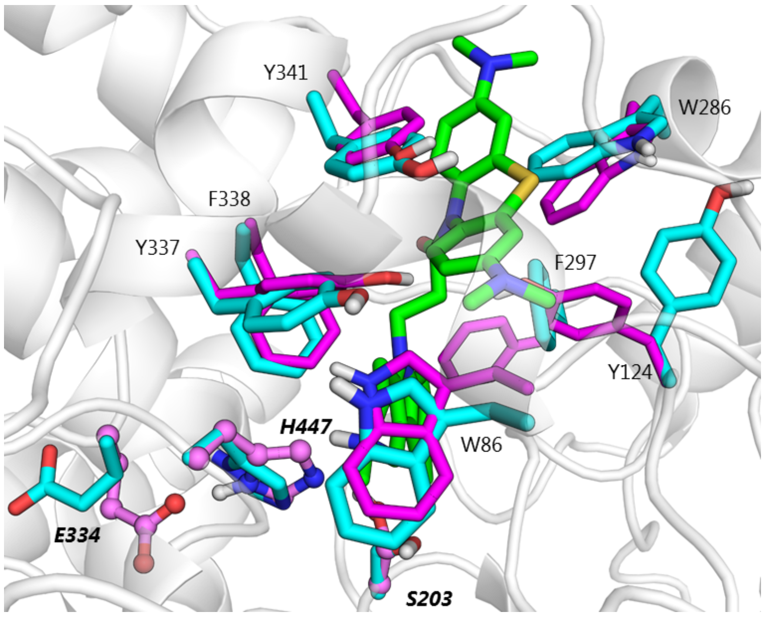


Figure 7. Protein-inhibitor complex obtained as a result of Induced Fit procedure (Schrödinger/Glide). The **MBC** ligand carbon atoms are green and protein carbon atoms are cyan. The docked complex is overlaid with the *apo*-AChE X-ray structure (carbon atoms are magenta)

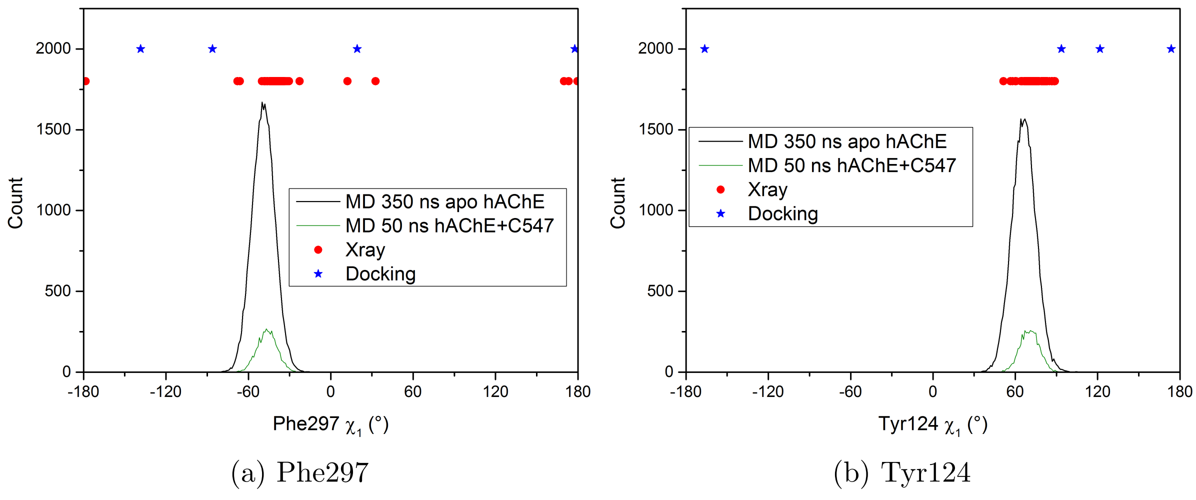


Figure 8. Distribution of values of χ_1 torsion angle over MD trajectories for *apo*-AChE (black line), total length 350 ns and 50 ns with the bulky inhibitor C-547 [6] (green line). Corresponding values for X-ray structures of human and mouse AChE available in the PDB are overlaid on the distribution plot with red points, and Induced Fit docking results are overlaid with blue stars

Acknowledgements

The study was partly supported by the Russian Science Foundation (project №14-23-00160P) and Russian State assignment №0090-2017-0019. The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University supported by the project RFMEFI62117X0011. Support for RJR's contribution was provided by the University of Michigan.

References

1. Bachurin, S.O., Makhaeva, G.F., Shevtsova, E.F., Boltneva, N.P., Kovaleva, N.V., Lushchekina, S.V., Rudakova, E.V., Dubova, L.G., Vinogradova, D.V., Sokolov, V.B., Aksinenko, A.Y., Richardson, R.J., Aliev, G.: Conjugates of methylene blue with γ-carboline derivatives as new multifunctional agents for the treatment of neurodegenerative diseases. Sci Rep in press (2019)

2. Chen, Y.C.: Beware of docking! Trends Pharmacol Sci 36(2), 78–95 (2015), DOI: 10.1016/j.tips.2014.12.001

3. Cheung, J., Rudolph, M.J., Burshteyn, F., Cassidy, M.S., Gary, E.N., Love, J., Franklin, M.C., Height, J.J.: Structures of human acetylcholinesterase in complex with pharmacologically important ligands. J Med Chem 55(22), 10282–10286 (2012), DOI: 10.1021/jm300871x

4. Cho, A.E., Guallar, V., Berne, B.J., Friesner, R.: Importance of accurate charges in molecular docking: Quantum mechanical/molecular mechanical (qm/mm) approach. J Comp Chem 26(9), 915–931 (2005), DOI: 10.1002/jcc.20222

5. Hawkins, P.C., Skillman, A.G., Warren, G.L., Ellingson, B.A., Stahl, M.T.: Conformer generation with omega: algorithm and validation using high quality structures from the protein databank and cambridge structural database. J Chem Inf Model 50(4), 572–584 (2010), DOI: 10.1021/ci100031x

6. Kharlamova, A.D., Lushchekina, S.V., Petrov, K.A., Kots, E.D., Nachon, F., Villard-Wandhammer, M., Zueva, I.V., Krejci, E., Reznik, V.S., Zobov, V.V., Nikolsky, E.E., Masson, P.: Slow-binding inhibition of acetylcholinesterase by an alkylammonium derivative of 6-methyluracil: mechanism and possible advantages for myasthenia gravis treatment. Biochem J 473(9), 1225–1236 (2016), DOI: 10.1042/BCJ20160084

7. Masson, P., Lushchekina, S.V.: Slow-binding inhibition of cholinesterases, pharmacological and toxicological relevance. Arch Biochem Biophys 593, 60–68 (2016), DOI: 10.1016/j.abb.2016.02.010

8. Morris, G.M., Huey, R., Lindstrom, W., Sanner, M.F., Belew, R.K., Goodsell, D.S., Olson, A.J.: Autodock4 and autodocktools4: Automated docking with selective receptor flexibility. J Comput Chem 30(16), 2785–2791 (2009), DOI: 10.1002/jcc.21256

9. Mufson, E.J., Counts, S.E., Perez, S.E., Ginsberg, S.D.: Cholinergic system during the progression of alzheimer's disease: therapeutic implications. Expert Rev Neurother 8(11), 1703–1718 (2008), DOI: 10.1586/14737175.8.11.1703

10. Rouleau, J., Iorga, B.I., Guillou, C.: New potent human acetylcholinesterase inhibitors in the tetracyclic triterpene series with inhibitory potency on amyloid beta aggregation. Eur J Med Chem 46(6), 2193–2205 (2011), DOI: 10.1016/j.ejmech.2011.02.073

11. Schmidt, M.W., Baldridge, K.K., Boatz, J.A., Elbert, S.T., Gordon, M.S., Jensen, J.H., Koseki, S., Matsunaga, N., Nguyen, K.A., Su, S.J., Windus, T.L., Dupuis, M., Montgomery, J.A.: General atomic and molecular electronic-structure system. J Comp Chem 14(11), 1347–1363 (1993), DOI: 10.1002/jcc.540141112

12. Semenov, V.E., Zueva, I.V., Mukhamedyarov, M.A., Lushchekina, S.V., Kharlamova, A.D., Petukhova, E.O., Mikhailov, A.S., Podyachev, S.N., Saifina, L.F., Petrov, K.A., Minnekhanova, O.A., Zobov, V.V., Nikolsky, E.E., Masson, P., Reznik, V.S.: 6-methyluracil derivatives as bifunctional acetylcholinesterase inhibitors for the treatment of alzheimer's disease. ChemMedChem 10(11), 1863–1874 (2015), DOI: 10.1002/cmdc.201500334

13. Sherman, W., Day, T., Jacobson, M.P., Friesner, R.A., Farid, R.: Novel procedure for modeling ligand/receptor induced fit effects. J Med Chem 49(2), 534–553 (2006), DOI: 10.1021/jm050540c

14. Sokolov, V.B., Aksinenko, A.Y., Epishina, T.A., Goreva, T.V., Grigoriev, V.V., Gabrel'yan, A.V., Bachurin, S.O.: Synthesis and biological activity of n-substituted tetrahydro-γ-carbolins bearing bis(dimethylamino)phenothiazine moiety. Russ Chem Bull 64(3), 718–722 (2015), DOI: 10.1007/s11172-015-0925-3

15. Varfolomeev, S.D., Lushchekina, S.V., Nemukhin, A.V., Kulakova, A.M., Kots, E.D., Makhaeva, G.F., Delacour, H., Lockridge, O., Masson, P.: Molecular polymorphism of human enzymes as the basis of individual sensitivity to drugs. supercomputer-assisted modeling as a tool for analysis of structural changes and enzymatic activity of proteins. Russ Chem Bull 65(6), 1592–1607 (2017), DOI: 10.1007/s11172-016-1487-8

16. Yu, H.S., Watson, M.A., Bochevarov, A.D.: Weighted averaging scheme and local atomic descriptor for pka prediction based on density functional theory. J Chem Inf Model 58(2), 271–286 (2018), DOI: 10.1021/acs.jcim.7b00537

17. Zueva, I.V., Lushchekina, S.V., Masson, P.: Water structure changes in oxime-mediated reactivation process of phosphorylated human acetylcholinesterase. Biosci Rep 38(3), BSR20180609 (2018), DOI: 10.1042/BSR20180609

DOI: 10.14529/jsfi180411

Supercomputer Simulations of Dopamine-Derived Ligands Complexed with Cyclooxygenases

Valentina D. Maslova[1]***, Roman V. Reshetnikov***[2,6]***, Vladimir V. Bezuglov***[3]***, Igor I. Lyubimov***[4]***, Andrey V. Golovin***[1,2,5,6]

An *in silico* approach was adopted to identify potential cyclooxygenase inhibitors through molecular docking studies. Four potentially active molecules were generated by fusion of dopamine with ibuprofen or ketorolac derivatives. The binding mode of the considered ligands to cyclooxygenase-1 and cyclooxygenase-2 isoforms was described using Autodock Vina. Preliminary docking to full cyclooxygenase isoforms structures was used to determine possible binding sites for the described dopamine-derived ligands. The following more accurate docking iteration to the described binding sites was used to achieve better conformational sampling. Among the studied molecules, IBU-GABA-DA showed preferable binding to cyclooxygenase active site of cyclooxygenase-1, while IBU-DA bound to peroxidase site of cyclooxygenase-1, making these ibuprofen-comprising ligands a base for further research and design of selective cyclooxygenase-1 inhibitors. Keterolac-derived ligands KET-DA and KET-GABA-DA demonstrated binding to both cyclooxygenase isoforms at a side pocket, which does not relate to any known functional site of cyclooxygenases and needs to be further investigated.

Keywords: molecular docking, non-steroidal anti-inflammatory drugs, ibuprofen, dopamine, cyclooxygenase.

Introduction

Cyclooxygenases (COX), or prostaglandin-endoperoxide synthases, are a family of membrane-bound isozymes located on the lumenal surfaces of the endoplasmic reticulum and on the inner and outer membranes of the nuclear envelope. There are two human COX isoforms, COX-1 and COX-2, which mediate basic housekeeping functions in various tissues and play key roles in inflammation process. Non-steroidal anti-inflammatory drugs (NSAIDs) are COX inhibitors that exhibit analgesic, antipyretic, and anti-inflammatory actions [1]. Most NSAIDs are known to inhibit COX enzymes by binding at the cyclooxygenase active site, but several NSAIDs have alternative binding locations on COX surface [3]. Development of selective COX-1 inhibitors might be highly relevant for diseases, such as neuro-inflammation, atherosclerosis and gastrointestinal toxicity, while COX-2 selective NSAIDs are needed for treatment of rheumatoid arthritis and as a preventative agent for colon cancer [1].

Dopamine, one of the major neurotransmitters in the central nervous system, is involved in regulation of the immune system and host defense. Dopamine-derived drugs are a largely unexplored but promising class of mediators involved in the regulation of neuroinflammation, exhibiting reduction of prostaglandin E2 level in primary microglial cells without alteration of COX-2 gene expression [9].

[1]Lomonosov Moscow State University, Moscow, Russian Federation
[2]First Moscow State Medical University, Institute of Molecular Medicine, Moscow, Russian Federation
[3]Shemyakin-Ovchinnikov Institute of Bioorganic Chemistry, Russian Academy of Sciences, Moscow, Russian Federation
[4]Gurus BioPharm, LLC
[5]Faculty of Computer Science, Higher School of Economics, Moscow, Russian Federation
[6]Apto-Pharm LLC, Moscow, Russian Federation

An established technique of developing new COX inhibitors is modification of existing non selective inhibitors, such as ibuprofen. Ibuprofen and other NSAIDs esterified from their carboxyl group showed higher binding affinity and good selectivity for COX-2 in both *in silico* and *in vitro* studies [2]. In this study we investigate the binding mode to COX isozymes for four potential dopamine-derived COX inhibitors designed by ibuprofen and ketorolac esterficaton (Fig. 1). The preferable binding sites were determined by the estimated binding affinities using molecular docking approach. IBU-GABA-DA is selected as a lead molecule for further design of COX-1 selective inhibitors.

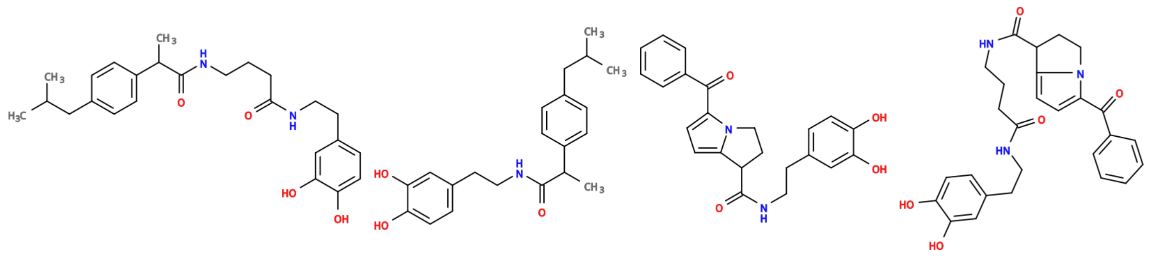


Figure 1. Chemical structures of chosen dopamine-derived COX inhibitors on base of ibuprofen From left to right: IBU-DA, IBU-GABA-DA, KET-DA, KET-GABA-DA

1. Methods

3D structures for four dopamine-derived ligangs (DDLs), the cyclooxygenase substrate arachidonic acid (AA) and NSAID ibuprofen (IBU) were generated from SMILES strings using Open Babel 2.3.2 [5]. The AutoDock Tools 1.5.6 software was used to assign atomic partial charges [4].

Three crystal structures of COX-1 and COX-2 bound to AA (PDB ID 1DIY for COX-1, 3HS5 for COX-2), ibuprofen (1EQG for COX-1, 4PH9 for COX-2) and other inhibitors (flurbiprofen - 2AYL for COX-1, naproxen - 3NT1 for COX-2) were used to sample ligand binding to different COX conformations. The macromolecules were treated to be rigid.

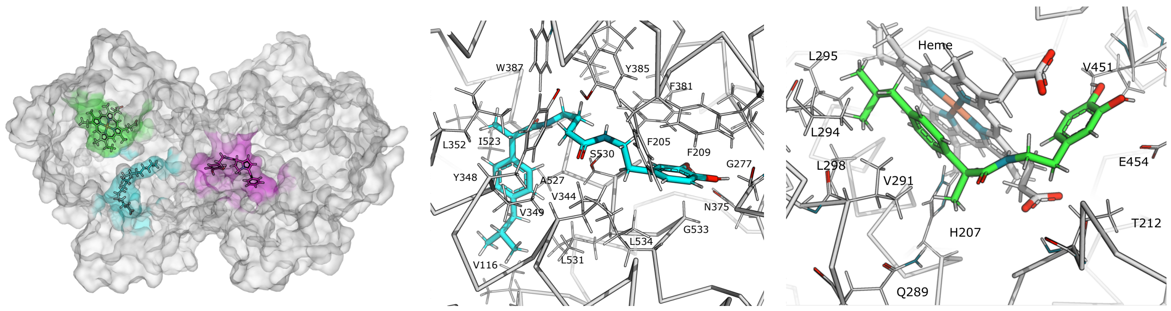
We first performed 140 docking iterations per COX structure of AA to box containing the whole protein structure (95.21Å x 96.96Å x 121.05Å) with `exhaustiveness`=8, using Autodock Vina [8]. AA:COX complexes were clustered with Affinity Propagation method implemented in Affbio python package [6]. The centroids of the resulting clusters were used as new centers of docking boxes (20Å x 20Å x 20Å). 140 docking iterations with `exhaustiveness`=256 per docking box per ligand per protein structure were performed.

2. Results

As NSAIDs are known to target various binding sites, we performed docking of AA to search space covering the whole COX isoform structure. Despite the fact that AA poses bound to active sites in both COX isoforms had highest binding scores (-7.9 kcal/mol in COX-1, -8.2 kcal/mol in COX-2), they represent only 4.8% and 6.7% of poses for COX-1 and COX-2. Thus, we performed docking of DDLs to smaller boxes formed around AA-located sites in order to achieve better pose sampling.

As a control study, we compared poses for AA and IBU with crystal structures by docking to all the selected boxes. The best binding energy was observed at active sites. The root mean square deviation (RMSD) to the crystal structures were 0.98Å and 0.92Å in COX-1 and COX-2 for AA, 0.56Å and 1.27Å for IBU, indicating the obtained poses correspond to actual conformations.

All four DDLs were then docked to all the selected docking boxes. For COX-1, IBU-GABA-DA demonstrated the best binding energy (-9.4 kcal/mol) at the subunit A cyclooxygenase active site (Fig. 2 (a)). Most interactions in this binding had hydrophobic character, except for hydrogen bonds with ASN-375 (see Fig. 2 (b)). For COX-2, IBU-GABA-DA had the best binding energy (-10.1 kcal/mol) at the side pocket on the subunit B lumenal surface. The pose of IBU-DA with the best binding energy (-9.3 kcal/mol) binds closely to peroxidase (POX) site of subunit A in COX-1 (Fig. 2 (a, c)). In COX-2, IBU-DA best binding site (-9.6 kcal/mol) was located on the subunit B membrane surface. In COX-2 both KET-DA and KET-GABA-DA bound better (-10.6 kcal/mol and -11.6 kcal/mol, for KET-DA and KET-GABA-DA) at the same side pocket on the subunit B lumenal surface as IBU-GABA-DA, and in COX-1 they demonstrated the highest binding score (-10.6 kcal/mol and -10.4 kcal/mol) at the corresponding site.



(a) Possible binding sites on COX-1 surface, green - POX site of subunit A, cyan - active site of subunit A, magenta - a pocket on the subunit B lumenal surface

(b) IBU-GABA-DA bound at active site of COX-1

(c) IBU-DA bound at the POX site of COX-1

Figure 2. Possible binding sites of DDLs on COX-1 surface

3. Discussion

Molecular docking is a powerful instrument for prediction of binding conformations for enzyme inhibitors. Autodock Vina is one of the most widely used tools for this task. Note that while the software provides a valuable insight into the geometry of intermolecular interactions, its binding affinity estimates should be treated with care [8].

Among the DDLs considered in this study, only IBU-GABA-DA shows preferable binding to one of the cyclooxygenase active sites of COX-1. The preferable pose interacts with the same set of COX-1 aminoacids as AA in the crystal structure, yet the estimated binding energy for IBU-GABA-DA is lower, -9.4 kcal/mol versus -7.9 kcal/mol for AA. As it does not demonstrate preferable binding to the active site of COX-2, we may suggest that IBU-GABA-DA can serve as a base for further design of COX-1 selective inhibitors.

The predicted IBU-DA pose interacts with Heme molecule bound to the POX site of COX-1. Such interaction may affect the activity of COX-1, as several other NSAIDs, such as resveratrol, bind at the POX site [3]. The binding energies for the POX site, active site and a pocket on the lumenal surface of COX-1 have a difference of 0.2 kcal/mol, indicating that IBU-DA may have different modes of action. The predicted binding mode of this ligand to COX-2 may not be functional, as the top scoring pose interacts with the membrane part of COX-2.

KET-DA and KET-GABA-DA bind to both COX isoforms at a side pocket, which does not relate to any known functional site. Despite the fact that the binding energy of these potential inhibitors is lower than the other DDLs' considered in this study, we can not state whether this binding would affect the activity of any COX isoform.

Conclusion

In this study we have described the binding mode of four dopamine-derived potential COX inhibitors using molecular docking. Among the candidates, IBU-GABA-DA is predicted to bind selectively at the active site of COX-1, making it a possible target for further drug development. IBU-DA is predicted to have a set of equivalent target sites on COX-1. A putative functional site was located with KET-DA and KET-GABA-DA docking experiments, however further research is needed to prove its significance.

Acknowledgments

The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University supported by the project RFMEFI62117X0011 [7]. The development of docking tools in this study was supported by the Ministry of Science and Education (RFMEF157617X0095).

References

1. Blobaum, A.L., Marnett, L.J.: Structural and functional basis of cyclooxygenase inhibition. Journal of Medicinal Chemistry 50(7), 1425–1441 (2007), DOI: 10.1021/jm0613166

2. Hegazy, G.H., Ali, H.I.: Design, synthesis, biological evaluation, and comparative cox1 and cox2 docking of p-substituted benzylidenamino phenyl esters of ibuprofenic and mefenamic acids. Bioorganic & Medicinal Chemistry 20(3), 1259–1270 (2012), DOI: 10.1016/j.bmc.2011.12.030

3. Kümmerle, A.E., da Silva, G.M.S., Sant'Anna, C.M., Barreiro, E.J., Fraga, C.A.: A proposed molecular basis for the selective resveratrol inhibition of the PGHS-1 peroxidase activity. Bioorganic & Medicinal Chemistry 13(21), 5981–5985 (2005), DOI: 10.1016/j.bmc.2005.07.028

4. Morris, G.M., Huey, R., Lindstrom, W., Sanner, M.F., Belew, R.K., Goodsell, D.S., Olson, A.J.: AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. Journal of Computational Chemistry 30(16), 2785–2791 (2009), DOI: 10.1002/jcc.21256

5. O'Boyle, N.M., Banck, M., James, C.A., Morley, C., Vandermeersch, T., Hutchison, G.R.: Open babel: An open chemical toolbox. Journal of Cheminformatics 3(1), 33 (2011), DOI: 10.1186/1758-2946-3-33

6. Reshetnikov, R.V., Stolyarova, A.V., Zalevsky, A.O., Panteleev, D.Y., Pavlova, G.V., Klinov, D.V., Golovin, A.V., Protopopova, A.D.: A coarse-grained model for DNA origami. Nucleic Acids Research 46(3), 1102–1112 (2017), DOI: 10.1093/nar/gkx1262

7. Sadovnichy, V., Tikhonravov, A., Voevodin, V., Opanasenko, V.: "Lomonosov": Supercomputing at moscow state university. In: Contemporary High Performance Computing: From Petascale toward Exascale. pp. 283–307. Chapman & Hall/CRC Computational Science, Boca Raton, United States, Boca Raton, United States (2013)

8. Trott, O., Olson, A.J.: Autodock vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. Journal of Computational Chemistry 31(2), 455–461 (2010), DOI: 10.1002/jcc.21334

9. Yoo, J.M., Park, E.S., Kim, M.R., Sok, D.E.: Inhibitory effect of n-acyl dopamines on IgE-mediated allergic response in RBL-2h3 cells. Lipids 48(4), 383–393 (2013), DOI: 10.1007/s11745-013-3758-6

DOI: 10.14529/jsfi180412

High Performance Computing of Magnetized Galactic Disks

***Sergey A. Khoperskov*[1], *Yulia A. Venichenko*[2], *Sergey S. Khrapov*[3], *Evgenii O. Vasiliev*[4,5]**

A parallel implementation of the magneto-hydrodynamical code for global modeling of the galactic evolution is reported. The code is parallelized by using MPI interface, and it shows ideal scaling up to 200–300 cores on Lomonosov supercomputer with fast interconnect. In the benchmarking of this code, we study the dynamics of a magnetized gaseous disk of a galaxy with a bar. We run a high-resolution 3D magnetohydrodynamic simulation taking into account the Milky Way-like gravitational potential, gas self-gravity and a network of cooling and heating processes in the interstellar medium. By using this simulation the evolution of morphology and enhancement of the magnetic field are explored. In agreement to hydrodynamical models, when the bar is strong enough, the gas develops sharp shocks at the leading side of the bar. In such a picture we found that when typically the magnetic field strength traces the location of the large-scale shocks along the bar major axis, the magnetic field pressure weakens the shocks and reduces the inflow of gas towards the galactic center.

Keywords: magnetohydrodynamics, galactic dynamics, galactic magnetic field, parallelization.

Introduction

In recent observations it has been established that magnetic fields in galaxies have very complex structure [1]. Moreover, the energy enclosed in galactic magnetic fields is of the same order as the thermal energy of the interstellar gas. Then, magnetic fields are believed to play a significant role in global galactic evolution [2]. Magnetic fields are mainly locked in the plane of the galactic disk, where the magnetic induction is maximum. However, they embrace the entire disk of the galaxy and reach galactic halos due to galactic fountains and winds. In spiral galaxies like our Galaxy, the magnetic fields are mainly frozen into the gas and follow its flows. So that to understand the energy exchange between magnetic fields and gas in galaxies, a self-consistent modeling of galactic evolution is required. A significant advance in numerical techniques closely related to a rapid growth of computational resources allow to construct three-dimensional high-resolution numerical simulations of global galactic magnetic field evolution in more realistic conditions.

In this paper, we consider the global evolution of magnetized galactic disks using our MPI-parallelized magneto-hydrodynamical code and study the parallel efficiency of this code on Lomonosov supercomputer [11].

1. Methods and Model

To investigate the global gas dynamics in magnetized spiral galaxies, we have conducted a set of the numerical simulations using our three-dimensional code based on TVD MUSCL (Total Variation Diminishing Multi Upstream Scheme for Conservation Laws) scheme. For magnetic field divergence cleaning, we adopt the constrained transport technique for magnetic field transport through the computational domain [5]. In this approach the magnetic field strength is

[1] Institute of Astronomy, Russian Academy of Sciences, Moscow, Russia
[2] Heidelberg University, Heidelberg, Germany
[3] Volgograd State University, Volgograd, Russia
[4] Southern Federal University, Rostov-on-Don, Russia
[5] Special Astrophysical Observatory of Russian Academy of Sciences, Nizhnii Arkhyz, Russia

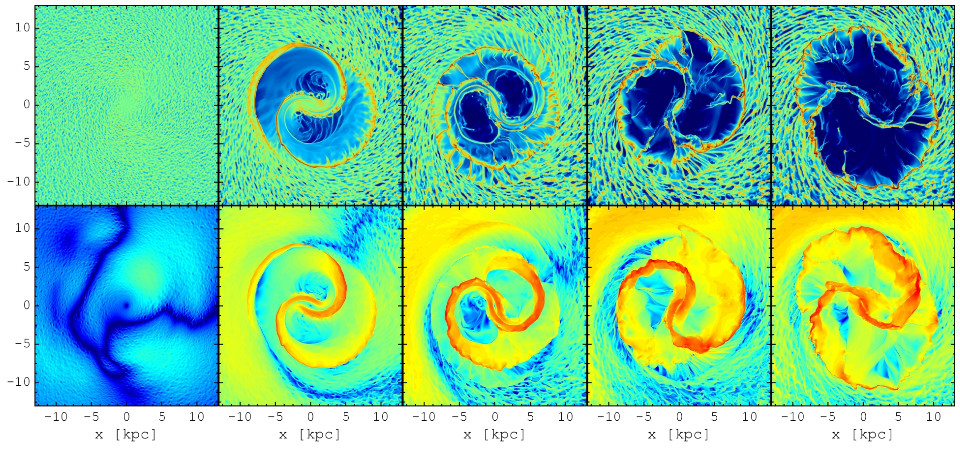


Figure 1. Evolution of gas surface density (top row) and magnetic field strength (bottom row) at 5, 50, 100, 150, 200 Myr

defined at faces of a cell, while the other gas dynamical variables are calculated at the center of a cell [9]. The code has successfully passed the standard tests for magnetic gas dynamics and has been already utilized for several galactic-scale simulations [6, 8].

We carry out 3D hydrodynamic simulations (Cartesian geometry) of a galactic disk in the computational box of size $20 \times 20 \times 4$ kpc with spatial resolution of $4,096 \times 4,096 \times 256$ grid zones in the x-, y-, and z-directions, respectively, that corresponds to a physical cell size of ≈ 4 pc. Such cell size is high enough to resolve molecular clouds at galactic scales.

To parallelize our code, the Message Passing Interface (MPI) software library is applied. The parallelization strategy is based on two-dimensional domain decomposition into cubic blocks, which are distributed across nodes. In our code, we use the third-order approximation of primitive gas dynamical variables, then two ghost cells are exchanged between nodes for boundary conditions. Despite the fact that our method of solving of the Poisson equation requires inefficient all-to-all communication, fast interconnect implemented on the Lomonosov cluster allows to reach good scaling on several tens of nodes or several hundred cores. To gain better scaling in the Riemann solver, we combine several scalar values into vector ones, that is favorable for using Advanced Vector Extensions (AVX2) instructions without complication of the code.

The gas density radial profile is distributed exponentially with a radial scale of 6 kpc and central density value equal to $10 M_\odot \ pc^{-2}$. The gaseous disk in the vertical direction is set in the hydrostatic equilibrium with a scale height of ≈ 100 pc. The equilibrium state of the gaseous disk is found according to the radial balance between the gas rotation versus radial gradient of gas pressure, gravitational forces (external potential and self-gravity) and magnetic field pressure. To mimic the turbulent structure of the magnetic field in the disk plane, its components are established as a superposition of two modes with (pseudo-) random location in the disk and various amplitudes [7].

2. Results

Global galactic disk evolution is driven by self-gravity, thermal instability, and rotation of the bar. These processes inevitably lead to the fragmentation of gaseous disk and formation of small-scale isolated clumps – giant molecular clouds, which may collide and merge with each other. A detailed description of this picture can be found in [3, 4, 10]. Figure 1 shows the distributions of

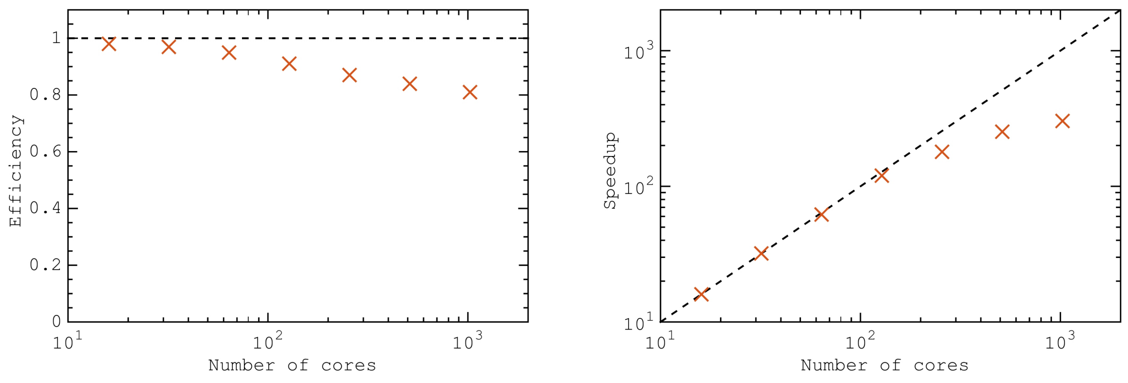


Figure 2. Parallel efficiency and speedup for the parallel code described in Section 1

gas density (top row) and magnetic field strength (bottom row). One can find that the magnetic field strength closely correlates with the gas density in the bar region, $r < 5$ kpc, where the field strength follows to gas adiabatically compressed when it moves through the spiral arms or bar. In our simulations the mean magnetic-field strength is consistent with that measured in the galactic disk, then we believe that the magnetic field inside clouds selected from the simulation should be similar to that in real giant molecular clouds. Mean magnetic energy is close to the equipartition with mean thermal and kinetic energies averaged over the whole galactic disk. Figure 1 demonstrates that the global magnetic field has a toroidal configuration.

The parallel code was benchmarked on Lomonosov supercomputer [11]. The number of cores in our runs has varied from 8 to 10^3 cores. We have rescaled our measurements in the way that the efficiency is equal to unity for a run on eight CPUs. Figure 2 shows how efficiency (left panel) and speedup (right panel) depend on the number of cores. One can find that both efficiency and speedup are ideal sat up to 200–300 cores. The efficiency remains higher than 0.8 till up to 10^3 cores. It is worth noting that the measured parallel efficiency depends on the load balance of processors, that in turn is determined by physical parameters of a simulated task.

Conclusions

In this paper we have studied the parallelization efficiency of the magneto-hydrodynamical code for global modeling of the galactic evolution. This code is parallelized by using MPI interface. A technique for more efficient use of AVX2 instructions has been applied. The code shows almost ideal scaling up to 200–300 cores. This code has been mainly developed to study the global evolution of disk galaxies. It can be used for modeling generation of spiral structure and evolution of interstellar medium taking into account self-gravity, thermal processes, magnetic fields, chemical kinetics, and so on. Here the evolution of the galactic magnetic field in barred galaxies is studied. In particular, we have found that the magnetic field strength closely correlates with the gas density in the bar region.

Acknowledgments

The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University supported by project RFMEFI62117X0011. SAK is supported by RFBR according to research project 16-32-60043.

SSK has been supported by the Ministry of Education and Science of the Russian Federation (government task No. 2.852.2017/4.6). EOV is thankful for support from the Russian Scientific Foundation (grant 14-50-00043).

References

1. Beck, R.: Galactic and Extragalactic Magnetic Fields. Space Science Reviews 99, 243–260 (2001)
2. Beck, R., Brandenburg, A., Moss, D., Shukurov, A., Sokoloff, D.: Galactic Magnetism: Recent Developments and Perspectives. Annual Review of Astronomy and Astrophysics 34, 155–206 (1996), DOI: 10.1146/annurev.astro.34.1.155
3. Dobbs, C.L., Burkert, A., Pringle, J.E.: Why are most molecular clouds not gravitationally bound? Monthly Notices of the Royal Astronomical Society 413, 2935–2942 (2011), DOI: 10.1111/j.1365-2966.2011.18371.x
4. Dobbs, C.L., Pringle, J.E.: The exciting lives of giant molecular clouds. Monthly Notices of the Royal Astronomical Society 432, 653–667 (2013), DOI: 10.1093/mnras/stt508
5. Evans, C.R., Hawley, J.F.: Simulation of magnetohydrodynamic flows – A constrained transport method. Astrophysical Journal 332, 659–677 (1988), DOI: 10.1086/166684
6. Khoperskov, S.A., Bertin, G.: Spiral density waves in the outer galactic gaseous discs. Monthly Notices of the Royal Astronomical Society 451, 2889–2899 (2015), DOI: 10.1093/mnras/stv1145
7. Khoperskov, S.A., Khrapov, S.S.: Global enhancement and structure formation of the magnetic field in spiral galaxies. Astromony and Astrophysics 609, 104–118 (2018), DOI: 10.1051/0004-6361/201629988
8. Khoperskov, S.A., Vasiliev, E.O.: A Kennicutt–Schmidt relation at molecular cloud scales and beyond. Monthly Notices of the Royal Astronomical Society 468, 920–926 (2017), DOI: 10.1093/mnras/stx532
9. Khoperskov, S.A., Vasiliev, E.O., Khoperskov, A.V., Lubimov, V.N.: Numerical code for multi-component galaxies: from N-body to chemistry and magnetic fields. Journal of Physics Conference Series 510(1), 012011 (2014), DOI: 10.1088/1742-6596/510/1/012011
10. Khoperskov, S.A., Vasiliev, E.O., Ladeyschikov, D.A., Sobolev, A.M., Khoperskov, A.V.: Giant molecular cloud scaling relations: the role of the cloud definition. Monthly Notices of the Royal Astronomical Society 455, 1782–1795 (2016), DOI: 10.1093/mnras/stv2366
11. Sadovnichy, V., Tikhonravov, A., Voevodin, V., Opanasenko, V.: "Lomonosov": Supercomputing at Moscow State University. In: Contemporary High Performance Computing. pp. 283–307. Chapman & Hall/CRC (2013)

DOI: 10.14529/jsfi180413

Regional Climate Model for the Lower Volga: Parallelization Efficiency Estimation

Alexander V. Titov[1], *Alexander V. Khoperskov*[1]

We have deployed the regional climate model (RCM) RegCM 4.5 for the Lower Volga and adjacent territories with a horizontal spatial resolution of 20 km. The problems of choosing the computational domain in the RCM RegCM version 4.5 are considered. We demonstrate the influence of this factor on the forecast of rainfall distribution in the numerical simulations. The study of rainfall and snowfall is a more demanding test in comparison with temperature or pressure distributions. We investigate dependencies of calculation time, parallel speedup and parallelization efficiency on the number of processes for different multi-core CPUs. Our analysis of the efficiency of parallel implementation of RegCM for various multi-core and multi-processor systems show a strong dependence of the simulation speed on the CPU type. The best effect is achieved when the number of CPU threads and the number of parallel processes are equal. The parallel code speedup is in the range of 1.8 – 11 for different CPUs.

Keywords: regional climate model, domain size, simulations, parallelization.

Introduction

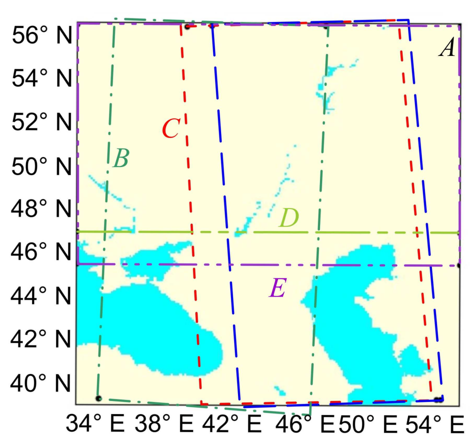
The solution to the problem of improving the accuracy of the climate changes forecast for a specific region is based on the massive use of regional climate models (RCMs) for calculations [7]. There are a number of reasons for the general interest in climate forecasts. In addition to the increasing global warming, which is investigated on the basis of general circulation models, we observe multidirectional trends of regional changes that are very important for engineering infrastructure, agricultural production, recreational projects, evaluation of the state of natural landscapes and especially river systems [2]. The study of extreme weather phenomena is at the forefront of climate sciences. RCMs allows to take into account the region-specific orographic features [1].

It is important to emphasize that it is not enough to increase only the spatial resolution of RCM in order to improve the quality of simulation results [6, 7]. We must successfully configure a large set of parameters describing heterogeneous subgrid processes for the study area. Parameterization of physical subprocesses, data reanalysis, radiation models, methods of meteorological parameters downscaling, boundary conditions, and choice of the calculation domain are crucial for the results of climate modeling [5]. In this work, our main efforts are aimed at analyzing the efficiency of parallel computing for the regional climate model RegCM 4.5. We examined several various multi-core processors for climate modeling.

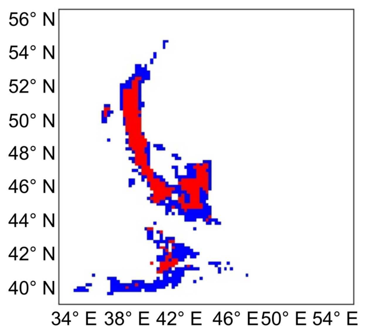
1. Regional Climate Model for Low Volga

When performing calculations, we used the standard set of parameters recommended by the Weather and Climate Physics Group of ICTP for the Caspian region (Fig. 1), and only the sizes and positions of the computational domain varied. We based our study on the hydrostatic core with the numerical grid resolution of 20 km and 18 vertical σ-levels and used the topographic data of GTOPO with resolution of 30 seconds, the data from the global climate model of the European Center for Medium-Range Weather Forecast's ERA-Interim (EIN15) for setting the

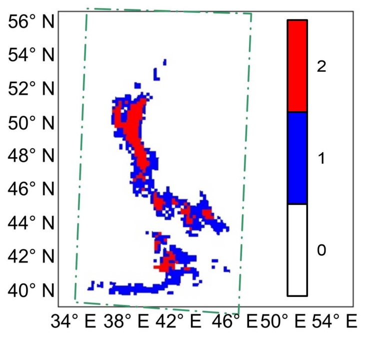
[1]Volgograd State University, Volgogorad, Russian Federation



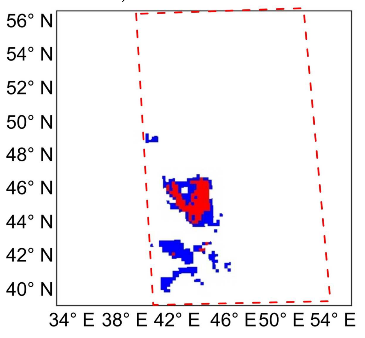
a) Computational domains for different models: A is the basic model with the size of $2,000\,\mathrm{km} \times 2,000\,\mathrm{km}$, the center of what is located in Volgograd City; models B, C, D, E were based on the about half the size of the model A



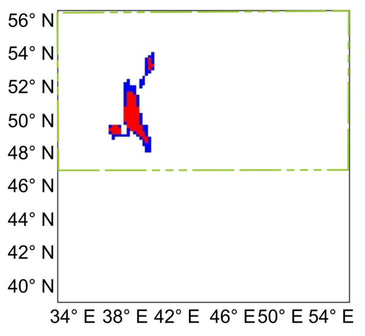
b) model A



c) model B



d) model C



e) model D

Figure 1. Examples of rainfall distributions are shown on the right for different computational domains

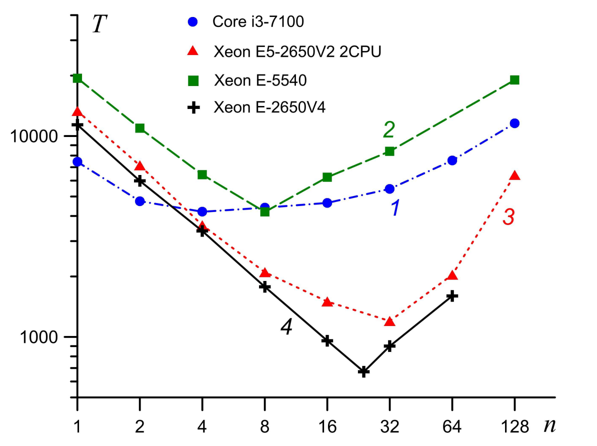
initial and boundary conditions in all our numerical experiments. The determination of the sea surface temperature is based on the reanalysis data of the Indian Ocean Sea Surface Temperature (IOSST). Our basic computational domain is located between 40°N – 56°N and 34°E – 54°E. The size of this domain (Fig. 1a) is typical for regional climate models.

We investigated the occurrence of special meteorological events, in particular, related to the rainfall formation. This approach seems more suitable, rather than considering the distributions of temperature or pressure [5]. In Fig. 1, we distinguish three types of events: very weak rainfall ($I < 0.86\,\mathrm{mm\,day}^{-1}$, symbol "0"), weak or moderate rainfall (symbol "1", blue color), heavy rainfall ($I > 24\,\mathrm{mm\,day}^{-1}$, symbol "2", red color). Fig. 1 $b-e$ show the spatial distributions of rainfall for the four computational domains on May 19, 2016 and for the initial state on May 1, 2016. The choice of the computational domain significantly affects the results of rainfall modeling. For the same area of the computational domain, we have quantitative and even qualitative differences in the meteorological situation when the center and the orientation of the domain change. Solving this problem requires expanding the modeling domain, and the computational resources is the limiting factor.

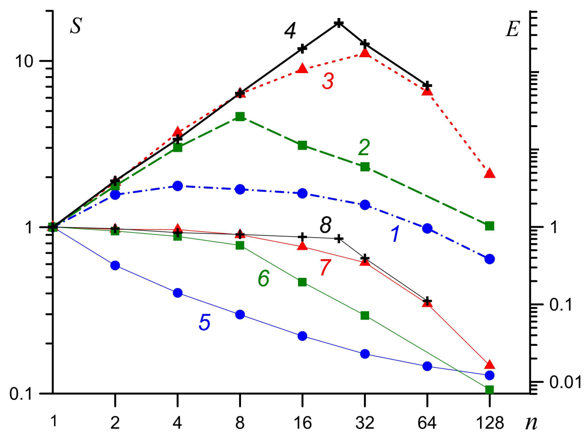
2. Parallelization Result

The climate modeling demands a large amount of calculation due to the need to vary a large number of parameters [4, 8]. The construction of very large data cubes is the basis for climate models. The dependence of the results in RCM on the computational domain requires the maximum possible sizes of the simulated area while maintaining the minimum scale of the numerical cell of about 10 km. And the ideal solution would be to use general circulation models with a resolution of up to 10 km for non-hydrostatic equilibrium. Let us consider the time of calculating T for modeling the climate system for 1 month with integration step of $\Delta t = 150\,\text{sec}$ to evaluate the parallelization efficiency in OpenMP. The time $T = T^{(cal)} + T^{(rw)}$ is determined by the time of pure integration of the hydrothermodynamics equations $T^{(cal)}$ and the time $T^{(rw)}$, which is necessary for reading and writing various data. The standard situation is to record the state of the climate system every 6 hours. We used the MPI2 library to run in parallel mode for multi-core systems.

Figure 2 shows the dependencies of T on the number of parallel processes n on different CPUs. Each processor has n_{cor} cores, for example, $n_{cor} = 2$ for Intel Core i3–7100, $n_{cor} = 16$ for Intel Xeon E5–2650V2 2CPU, $n_{cor} = 4$ for Xeon E–5540, $n_{cor} = 12$ for Xeon E–2650V4. Intel Hyper-threading technology doubles the number of logical processes, and we have $n_{th} = 2n_{cor}$ threads for a particular CPU type. The software using the MPI2 library allows you to specify an arbitrary number of processes n (Fig. 2), including $n \gg n_{th}$. The minimum time T is reached at $n = n_{th}$ for all the CPUs under investigation, and a very wide extremum is a characteristic feature. The transition from $n = 1$ to $n = 2$ gives the largest decrease of T. The fast growth of T with increasing n is observed in region $n > n_{th}$, so that $T((2 \div 4)n_{th}) \sim T(n = 1)$. These peculiar properties are due to the significant contribution of $T^{(rw)}$ to T.



a) Dependencies of calculation time T on the number of processes n for different multi-core CPUs



b) Parallel speedup $S = T(1)/T(n)$ (left axis, curves *1–4*) and parallelization efficiency $E = S/n$ (right axis, curves *5–8*) for the same computing systems

Figure 2. Parallelization result

Conclusions

Modeling of climatic changes for the territory of the Lower Volga region was carried out using the regional climate model RegCM version 4.5. We demonstrated the influence of the computational domain choice on the forecast of rainfall distribution in the numerical model. The study of rainfall and snowfall is a more demanding test in comparison with temperature or pressure distributions and requires a computational domain with a size of at least 3,000 km in the conditions of the Lower Volga.

The maximum speedup of parallel computing for OpenMP strongly depends on the CPU Type and varies from 1.8 to 11 for different CPUs. Mass transfer of regional climate models to GPUs is a priority task in accordance with the general trend of development of computational fluid dynamics [3].

Acknowledgments

AK has been supported by the Ministry of Science and Higher Education of the Russian Federation (government task No. 2.852.2017/4.6). AT is thankful to RFBR and Volgograd Region Administration (grant No. 18-47-340003). We thank the Weather and Climate Physics Group of ICTP for providing the RegCM-4.5. Thanks also to ECMWF, for free access to the ERA Interim reanalysis. The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University supported by project RFMEFI62117X0011.

References

1. Hui, P., Tang, J., Wang, S., Wu, J., Kang, Y.: Future climate projection under IPCC A1B scenario in the source region of Yellow River with complex topography using RegCM3. Journal of Geophysical Research: Atmospheres 119(19), 11,205–11,222 (2014), DOI: 10.1002/2014JD021992

2. Kalugin, A.S., Motovilov, Y.G.: Runoff Formation Model for the Amur River Basin. Water Resources 45(2), 121–132 (2018), DOI: 10.7868/S0321059618020013

3. Khrapov, S., Khoperskov, A.: Smoothed-Particle Hydrodynamics Models: Implementation Features on GPUs. Communications in Computer and Information Science 793, 266–277 (2017), DOI: 10.1007/978-3-319-71255-0_21

4. Kuhn, M., Kunkel, J., Ludwig, T.: Data Compression for Climate Data. Supercomputing Frontiers and Innovations 3(1), 75–94 (2016), DOI: 10.14529/jsfi160105

5. Politi, N., Nastos, P., Sfetsos, A., Vlachogiannis, D., Dalezios, N.: Evaluation of the AWR-WRF model configuration at high resolution over the domain of Greece. Atmospheric Research 208, 229–245 (2018), DOI: 10.1016/j.atmosres.2017.10.019

6. Raghavan, S.V., Liu, J., Nguyen, N.S., Vu, M.T., Liong, S.Y.: Assessment of CMIP5 historical simulations of rainfall over Southeast Asia. Theoretical and Applied Climatology 132(3), 989–1002 (2018), DOI: 10.1007/s00704-017-2111-z

7. Wang, Y., Leung, L.R., McGregor, J.L., Lee, D.K., Wang, W.C., Ding, Y., Kimura, F.: Regional Climate Modeling: Progress, Challenges, and Prospects. Journal of the Meteorological Society of Japan. Ser. II 82(6), 1599–1628 (2004), DOI: 10.2151/jmsj.82.1599

8. Wang, Y., Jiang, J., Zhang, J., He, J., Zhang, H., Chi, X., Yue, T.: An efficient parallel algorithm for the coupling of global climate models and regional climate models on a large-scale multi-core cluster. Journal of Supercomputing 74(8), 3999–4018 (2018), DOI: 10.1007/s11227-018-2406-6

DOI: 10.14529/jsfi180414

Performance Analysis of Different Computational Architectures: Molecular Dynamics in Application to Protein Assemblies, Illustrated by Microtubule and Electron Transfer Proteins

Vladimir A. Fedorov[1]***, Ekaterina G. Kholina***[1]***, Ilya B. Kovalenko***[1,2,3,4]***, Nikita B. Gudimchuk***[1,5]

All-atom molecular dynamics simulation represents a computationally challenging, but powerful approach for studying conformational changes and interactions of biomolecules and their assemblies of different kinds. Usually, the numbers of simulated particles in modern molecular dynamics studies range from thousands to tens of millions, while the simulated timescales span from nanoseconds to microseconds. For cost and computation efficiency, it is important to determine the optimal computer hardware for simulations of biomolecular systems of different sizes and timescales. Here we compare performance and scalability of 17 commercially available computational architectures, using molecular dynamics simulations of water and two different protein systems in GROMACS-5 package as computing benchmarks. We report typical single-node performance of various combinations of modern CPUs and GPUs, as well as multiple-node performance of "Lomonosov-2" supercomputer in molecular dynamics simulations of different protein systems in nanoseconds per day. These data can be used as practical guidelines for selection of optimal computer hardware for various molecular dynamics simulation tasks.

Keywords: molecular dynamics, tubulin, microtubule, plastocyanin-cytochrome f.

Introduction

Molecular dynamics (MD) is a powerful method to study conformational dynamics and interactions of biomolecules, including protein assemblies. Because of a big number of particles, which make up protein systems, and multiple computational steps, usually required to achieve meaningful results, MD simulations of proteins represent a major computational challenge. Therefore, indentification and use of optimal hardware for high efficiency calculations is important. In this work we systematically compare multiple currently available computer architechtures in their MD simulation performance, using two types of biomolecular systems as computing benchmarks: (i) water boxes of different sizes and (ii) two protein systems. The selected protein systems include different assemblies of tubulins, the building blocks of microtubules [5], and a photosynthetic electron-transfer complex of plastocyanin and cytochrome f proteins [4].

1. Methods

All-atom explicit solvent MD was used in all tests. Calculations were performed with the use of software package GROMACS-5 [3], which allows parallel computing on hybrid architecture with the CHARMM27 force field. All benchmarks were run for 15 minutes. TIP3P water model was employed. The protein structures were obtained from the Protein Data Bank. We

[1]Lomonosov Moscow State University, Moscow, Russia
[2]Federal Research and Clinical Center of Specialized Medical Care and Medical Technologies, Federal Medical and Biological Agency of Russia, Moscow, Russia
[3]Astrakhan State University, Astrakhan, Russia
[4]Scientific and Technological Center of Unique Instrumentation of the RAS, Moscow, Russia
[5]Center for Theoretical Problems of Physicochemical Pharmacology, RAS, Moscow, Russia

used higher plant plastocyanin-cytochrome f complex (PDB id 2PCF) and GMPCPP-bound tubulin structure (PDB id 3J6E). The size of the virtual cell was chosen in such a way that the distance from the protein surface to the nearest box boundary was no less than two nanometers. The particle mesh Ewald method was used for the long-range electrostatics. All-bond PLINKS constraints and mass rescaling were applied to the tested protein systems. Coulomb and Lennard-Jones cut-offs were both set to 1.25 nm. Specifications of MD systems used for benchmarking are summarized in (Tab. 1).

Table 1. Molecular dynamics systems used for benchmarking

MD systems	MD system name	Number of atoms	Box type	System size (nm)	Time step (fs)
Water box(WB)	WB-10	10,206	cube	4.7x 4.7x 4.7	1
	WB-80	80,232	cube	9.3x 9.3x 9.3	1
	WB-120	121,527	cube	10.7x10.7x10.7	1
	WB-160	159,780	cube	11.7x11.7x11.7	1
	WB-200	203,415	cube	12.7x12.7x12.7	1
Plastocyanin-cytochrome f	Pc-cyt	93,085	dodecahedron	11x11x11	5
Tubulin tetramer	Tub-4	315,718	cube	9.9x13.9x23.3	4
Tubulin 18-mer	Tub-18	1,119,458	cube	22.8x15.0x33.3	4

2. Results and Discussion

To begin with, we used our first benchmark, the water box, in order to examine performance of MD simulations as a function of the number of particles in the molecular system. We conducted MD simulations of water boxes of various sizes using 17 different single-node computer systems with various CPU/GPU architectures. Data summarized in (Tab. 2) suggest that an increase of MD system size leads to an unproportional decrease of computer performance. However, the relative extent of the decrease is almost equal for different computer systems. Not suprisingly, 2×Intel Xeon E5-2695 with four Tesla K80 GPUs shows the highest performance for all the tested MD systems. However, Intel Core i7-5930K with GTX 980 has the most optimal performance-price combination out of all hardware configurations we tested, consistent with conclusions of a previous study [2].

To further address the question of scalability, we used our second type of computing benchmark and established the dependence of the supercomputer "Lomonosov-2" performance in MD simulations on the number of computer nodes used. As expected, we could clearly see that for all three tested protein systems the performance grew as a function of the number of supercomputer nodes (Tab. 3). The relative rate of that growth did not significantly depend on the type and size of the biological system and slowed down gradually, roughly following Amdahl's law.

Table 2. Single-node performance (ns/day) depending on various combinations of CPUs and GPUs tested by MD GROMACS simulations of water boxes of different sizes. Time step is 1 fs

Processor	GPU	System name				
		WB-10	WB-80	WB-120	WB-160	WB-200
Intel Core i7-3820	GTX 680	57.8	8.1	5.3	4.1	3.2
	GTX 780	72.9	10.5	6.6	5.1	3.9
Intel Core i5-3570K	GTX 780	64.9	9.7	6.2	4.5	3.5
	GTX 980	82.5	10.4	7.1	4.9	3.1
Intel Core i7-4790K	GTX 780	85.9	12.3	7.3	5.7	4.3
	GTX 980	106.4	13.4	8.4	6.0	4.7
2×AMD Opteron 6168	GTX 980	61.1	9.9	6.7	5.1	4.0
AMD Phenom II X6 1100T	GTX 780	56.2	7.6	4.8	3.7	2.8
Intel Core i7-5930K	GTX 960	80.5	11.2	7.5	5.7	4.5
	GTX 970	103.1	15.0	10.2	7.7	6.1
	GTX 980	116.3	17.5	11.4	8.8	6.8
2×Intel Xeon E5-2695	1×Tesla K80	88.4	13.9	9.4	7.2	5.6
	2×Tesla K80	140.4	24.3	16.6	12.6	9.9
	4×Tesla K80	162.3	36.2	24.4	19.7	15.0
	no GPU	88.9	13.3	8.7	6.7	5.1
Intel Xeon 5670 (“Lomonosov-1”)	Tesla X2070	33.4	4.7	3.2	2.4	1.9
2×Intel Xeon E5-2697 (“Lomonosov-2”)	Tesla K40	87.5	13.5	9.0	6.9	5.5

Table 3. Performance of “Lomonosov-2” supercomputer (ns/day), dependending on the number of computing nodes for MD simulations of different protein systems. System size in atoms is given in brackets. Time step is 4 fs

	1	2	3	4	5	6	7	8
Pc-cyt (93085)	53.5	93.7	128.6	160.7	186.3	188.1	209.8	256.4
Tub-4 (315718)	16.1	27.2	34.1	43	48.7	51.3	53.4	61.6
Tub-18 (1119458)	4.5	5.4	8.5	12.1	16.3	18.4	21.4	22.5

Conclusion

Our comparative performance analysis suggests that for relatively small biomolecular systems, below 100,000 atoms, such as the complex of plastocyanin and cytochrome f proteins, it is quite practical to use “personal supercomputers”, i.e. single node workstations with a video accelerator. Such a computer can provide 100 ns/day performance for molecular dynamics calculation of a small biomolecular system with the size of about ten thousand atoms. For larger biomolecular systems, like a fragment of microtubule or a part of biological membrane with protein complexes, “personal supercomputers” are not currently fast enough, with typical performance of only several ns/day. Therefore, for large systems usage of modern supercomputers, like “Lomonosov-1” or “Lomonosov-2” with hybrid architecture is imperative [1]. By employing dozens of supercomputer nodes, such hardware systems are capable of accelerating calculations by an order of magnitude, providing up to 22 ns/day performance of GROMACS-5 MD simulation for a system sized more than one million particles.

Acknowledgments

This study was designed and conducted with support from the Russian Foundation for Basic Research, projects 16-34-60113 and 16-04-01862. The research was carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University supported by project RFMEFI62117X0011.

References

1. Sadovnichy, V., Tikhonravov, A., Voevodin, V., Opanasenko, V.: Lomonosov : Supercomputing at Moscow State Univer. Contemporary High Performance Computing: From Petascale toward Exascale (Chapman & Hall/CRC Computational Science) pp. 283–307 (2013)

2. Kutzner, C., Páll, S., Fechner, M., Esztermann, A., de Groot, B.L., Grubmüller, H.: Best bang for your buck: GPU nodes for GROMACS biomolecular simulations. Journal of computational chemistry 36(26), 1990–2008 (2015), DOI: 10.1002/jcc.24030

3. Abraham, M.J., Murtola, T., Schulz, R., Páll, S., Smith, J.C., Hess, B., Lindahl, E.: Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX 1, 19–25 (2015), DOI: 10.1016/j.softx.2015.06.001

4. Kovalenko, I., Khrushchev, S., Fedorov, V., Riznichenko, G.Y., Rubin, A.: The role of electrostatic interactions in the process of diffusional encounter and docking of electron transport proteins. In: Doklady Biochemistry and Biophysics. vol. 468, pp. 183–186. Springer (2016), DOI: 10.1134/S1607672916030066

5. Zakharov, P.N., Arzhanik, V.K., Ulyanov, E.V., Gudimchuk, N.B., Ataullakhanov, F.I.: Microtubules: dynamically unstable stochastic phase-switching polymers. Physics-Uspekhi 59(8), 773 (2016), DOI: 10.3367/UFNe.2016.04.037779

DOI: 10.14529/jsfi180415

Algorithm of the Parallel Sweep Method for Numerical Solution of the Gross–Pitaevskii Equation with Highest Nonlinearities

Andrey D. Bulygin[1,2]

In this paper, we for the first time introduce a numerical scheme the solution of a nonlinear equation of the Gross–Pitaevskii type (GP) or the nonlinear Schrodinger equation (NLSE) with highest nonlinearities, which provides implementation of a complete set of motion integrals. This scheme was parallelly implemented on a non-uniform grid. Propagation of a ring laser beam with non-zero angular momentum in the filamentation mode is studied using the implemented numerical scheme. It is shown, that filaments under exposure to centrifugal forces escape to the periphery. Based on a number of numerical experiments, we have found the universal property of motion integrals in the non-conservative case for a given class of equations. Research of dynamics of angular momentum for a dissipative case are also presented. We found, that angular moment, particularly normed by initial energy during filamentation process, is quasi-constant.

Keywords: nonlinear Schrodinger equation, fast parallel algorithm, fully conservative numerical scheme, motion integral.

Introduction

A nonlinear parabolic partial differential equation (or a system of such equations) occurs in many applications [3]. In such equations, their rigorous analytical solutions are often unknown. Generally, such equations are solved by numerical methods. Correctness of application of these numerical methods has to be controled, possibly, by comparing the solutions obtained with the known rigorous properties of these equations. Apparently, for the whole branch of these studies, such stage has been completed [2]. Indeed, inspite of the fact that for the GP (NLSE) equation with highest nonlinearities such properties are known, application checks of these properties were not carried out anywhere, except for [2]. One of special cases of this class of equations is the nonlinear Schrodinger equation with highest nonlinearities. In [2], a wide range of numerical schemes used for solution of the GP (NLSE) equation with highest nonlinearities was constructed by the example of a case with radial symmetry, and it was determined that the simplest and quite effective numerical method is the method of splitting by physical factors method. At the same time, discrete difference methods for solving the NLS equation are optimal for tracking and suppressing numerical imbalances, and the adaptive step along the evolutionary coordinate should be selected according to the conditions of preserving the Hamilton function on the numerical solution of the GP (NLSE) equation. Usually this step is significantly less compared to those offered in other works. In case of implementing this requirement on a grid in 2D+1 dimensions, development of methods for numerical solution on the non-uniform grids with the used parallelization methods becomes urgent. In this paper, we solved this problem.

1. The GP (NLSE) Equation and its Exact Properties

In this paper, we numerically explore the behavior of solutions of the complex Gross – Pitaevskii (GP) (or non-linear Schrodinger equation (NLSE)) equation with higher nonlineari-

[1]V.E. Zuev Institute of Atmospheric Optics SB RAS, Tomsk, Russian Federation
[2]Tomsk State University, Tomsk, Russian Federation

ties [1]:

$$\partial_t\psi + i(\epsilon(\psi) - \partial_\mu\partial^\mu - i\alpha(\psi)/2)\psi = 0, \tag{1}$$

where x^μ are transverse coordinates (in this article, the situation is considered as $\vec{x} \in \mathbb{R}^2$); $\psi(x)$ is complex-valued function, which, depending on the context, can have a different physical meaning; $\epsilon(\phi)$ is nonlinear function in the simplest case of a cubic in the field, but we will consider a more complicated situation with higher nonlinearities; $\alpha(\psi)$ is the function of nonlinear absorption. We will discuss three conservation laws that correspond to three global symmetries: the outer conservation is a shift in the evolutionary coordinate $t \to t + a$ (H-energy), and the inner law is the phase shift of the complex field $\psi \to \psi e^{ia}$ (the E-number of a particle), and the symmetry with respect to turn transformation ($y_\mu = A^\nu_\mu x_\nu$ Where $A \in so2$) (M-angular momentum). Due to the fact that we consider a model with dissipation of the ratio to be generalizing, the known conservation laws will take the following form:

$$\frac{\partial H}{\partial t} = -\int(i\alpha(\psi\Phi^*_\psi - \psi^*\Phi_\psi))d\vec{x}, \tag{2}$$

where $H \equiv \int(\partial_\mu\psi^*\partial^\mu)\psi + F_\epsilon)d\vec{x}$ is the Hamilton function; $\epsilon \equiv \delta F_\epsilon(\phi)/\delta\phi$ is the power in terms of mechanics or the nonlinear additive to the refractive index of the terms of optics; $\Phi_\psi = \delta H/(\delta\psi)$

$$\frac{\partial}{\partial t}\int\psi\psi^* d\vec{x} = -\int\alpha\psi\psi^* d\vec{x}, \tag{3}$$

$$\frac{\partial}{\partial t}\int m d\vec{x} = -\int\alpha m d\vec{x}, \tag{4}$$

where $m \equiv \vec{P} \times \vec{x}$ is the density of angular momentum; $P_\mu = (\phi\partial_\mu\psi^* - \psi^*\partial_\mu\psi)/2i$ is the Poynting vector. Implementation of the given exact relations in the numerical solution, even in the conservative case, imposes very strict conditions on the numerical grid, which makes the use of parallel algorithms to be relevant.

2. The Numerical Scheme

Taking into account the initial conditions, we construct an inhomogeneous grid which is origin-symmetric with the distance between the nodes increasing according to the law of geometric progression x_I. The same way, we will implement splitting of the orthogonal coordinate y_I. For simplicity, assume that each processor has the same number of points. The following relation is binding global index J to local index j, which is localized on the processor with number q:

$$J = (q-1) * m_l + j. \tag{5}$$

We will enable numerical implementation of a step of diffraction on a three-point “cross” scheme with the second order of accuracy of approximation by the Laplace operator on the non-uniform grid. In this case, we need to solve a system of equations of the form:

$$a_J\psi^{l+1}_{J-1} - c_J\psi^{l+1}_J b_J\psi^{l+1}_{J+1} = -f^l_J. \tag{6}$$

We generalize the technique of fast parallelizing [4] to a complex case.

STATEMENT: Solution of a complex-valued equation can be found by the following relation:

$$\psi_J = g_{q-1}u_j + g_q\nu_j + w_j. \tag{7}$$

Here $a_j u_{j-1} - c_j u_j + b_j u_{j+1} = 0$, where conditions are met at the borders $u_{(q-1)m_l} = (1,0); u_{qm_l} = (0,0)$, $a_j\nu_{j-1} - c_j\nu_j + b_j\nu_{j+1} = 0$ and $\nu_{(q-1)m_l} = (0,0); \nu_{qm_l} = (1,0)$. And finally, we give the equations for the internal field $a_j w_{j-1} - c_j w_j + b_j w_{j+1} = -f_j$. Where conditions are met at the borders: $w_{(q-1)m_l} = (0,0); w_{q(m_l)} = (0,0)$. Here, crosslinking coefficients g_q can be found by the following equation:

$$A_q g_{q-1} - C_q g_q + B_q g_{q+1} = -F_q, \tag{8}$$

$$-C_0 g_0 + B_0 g_1 = -F_0, \tag{9}$$

$$A_M g_{M-1} - C_M g_M = -F_M, \tag{10}$$

where $C_0 = c_0 - b_0 u_1$, $B_0 = b_0 * \nu_1$,$F_0 = f_0 - b_0 w_1$, $A_q = a_{qm_l} u_{q(m_l-1)}$, $C_q = -a_{qm_l}\nu_{q*(m_l-1)} + c_{qm_l} - b_{qm_l} u_{j(m_l+1)}$, $B_q = b_{qm_l} u_{q(m_l-1)}$, $F_q = f_{qm_l} + a_{qm_l} w_{q(m_l-1)} - b_{qm_l} w_{q(m_l+1)}$, $q = 1..M-1$, $A_M = a_{Mm_l} u_{M(m_l-1)}$, $C_M = -a_{Mm_l}\nu_{M(m_l-1)} - b_{Mm_l} u_{j(m_l+1)}$, $F_M = f_{Mm_l} + a_{Mm_l} w_{M(m_l-1)}$.

The proof of these results can be provided by direct substitution.

2.1. Results of Numerical Calculations

Results of numerical calculation for a beam with initial condition of the form are given below (in a radial coordinate system):

$$\psi_0(r,\varphi) = (\exp(-0.5(r^2)) - \exp(-2.5(r^2))) * \exp(-i\varphi m_t) * f_{noise}, \tag{11}$$

where m_t is the topological charge (in our case, $m_t = 2$); $n_2 = 35$, $K = 8$. The remaining nonlinearity parameters were chosen in the same way as in [5]. Where f_{noise} is the noise component, which violates the central symmetry.

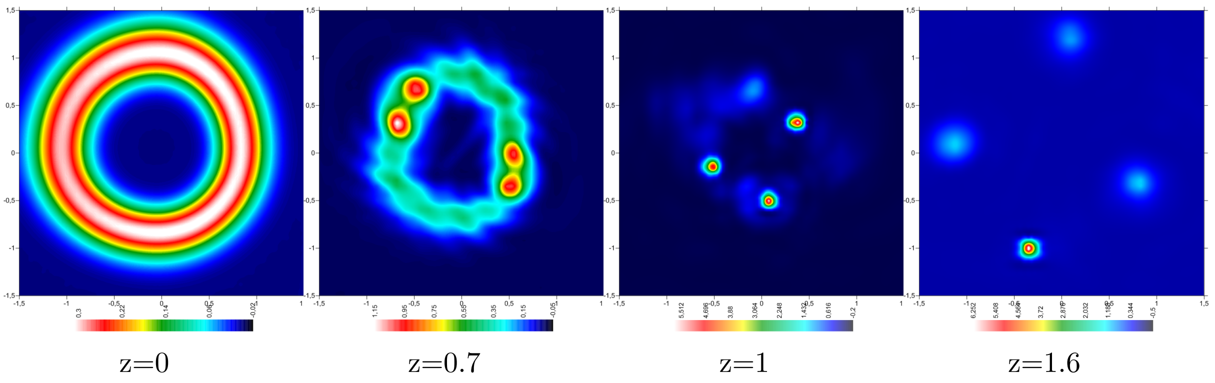


Figure 1. Distribution of the intensity field at different points of the propagation distance

Examples of distributions $|\psi|^2$ for different distances are shown in Fig.1. This example demonstrates that compression of beam in a result of the absorption effects has stopped, and, eventually, the beam's divergence is completely realized as it takes place and the similar radiative case [5]. A series of numerical experiments were made. The obtained results allow extending the findings in [5] on the general case. Namely, as we can see from numerical calculations, value ΔE is proportional to $\Delta\bar{H}$ with high degree of accurasy, i.e. the following relation is true:

$$H(z) = H(0) + \gamma\Delta E, \tag{12}$$

in addition, it was determined that

$$\bar{M} \equiv \int m d\vec{x}/E \approx const. \tag{13}$$

Conclusion

A parallel scheme for numerical solution of the NSE on an irregular grid using the accurate method for the system of solving of linear equations is proposed. This case is a tridiagonal system of linear algebraic equations which arises due to a discrete difference approximation of the GP (NLSE). Generalization of the Yanenko method to the complex case is proposed. By the example of numerical solution of the GP (NLSE) with topological charge, it is found that in case of absorption, the linear combination of the number of particles and energy is a constant value; the normalized number of particles of angular momentum is conserved with high accuracy.

Acknowledgments

The research has been performed using the equipment of the shared research facilities of the HPC computing resources at Lomonosov Moscow State University. The presented reported research was funded by the Russian Foundation for Basic Research and by the government of the Tomsk region of the Russian Federation, grant No. 18-41-703004 \18.

References

1. Balashov, A.D., Pergament A.Kh.: Mathematical modeling of femtosecond pulse propagation. Matem. Mod. 18(4), 3–18 (2006) (in Russian)

2. Bulygin, A.D., Zemlyanov, A.A.: Fully conservative numerical scheme for nonlinear Schrodinger equation with higher nonlinearities. Computational technologies 22(5), 3–13 (2017)

3. Tadmor E.: A review of numerical methods for nonlinear partial differential equations. Bulletin of the American Mathematical Society 49(4), 507–554 (2012), DOI: 10.1090/S0273-0979-2012-01379-4

4. Terekhov, A.V.: A fast parallel algorithm for solving block-tridiagonal systems of linear equations including the domain decomposition method. Parallel Computing 39(6–7), 245–258 (2013), DOI: 10.1016/j.parco.2013.03.003

5. Zemlyanov A.A., Bulygin A.D.: Analysis of Some Properties of the Nonlinear Schrodinger Equation Used for Filamentation Modeling. Russian Physics Journal 61(2), 357—363 (2018), DOI: 10.1007/s11182-018-1407-5