DETERMINISTIC SIMULATIONS OF LARGE-SCALE MODELS OF CELLULAR PROCESSES ACCELERATED ON GRAPHICS PROCESSING UNITS

Andrea Tangherloni[(1)], Paolo Cazzaniga[(2,4)], Marco S. Nobile [(1,4)], Daniela Besozzi[(3,4)], Giancarlo Mauri[(1,4)]

(1) Università degli Studi di Milano-Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Viale Sarca 336, 20126 Milano, Italy
a.tangherloni@campus.unimib.it; nobile/mauri@disco.unimib.it

(2) Università degli Studi di Bergamo
Dipartimento di Scienze Umane e Sociali
Piazzale S. Agostino 2, 24129 Bergamo, Italy
paolo.cazzaniga@unibg.it

(3) Università degli Studi di Milano
Dipartimento di Informatica
Via Comelico 39, 20135 Milano, Italy
besozzi@di.unimi.it

(4) SYSBIO Centre of Systems Biology
Piazza della Scienza 2, 20126 Milano, Italy

Keywords: GPU computing, deterministic simulation, Runge-Kutta, biochemical reaction network, large-scale model.

Abstract. The computational investigation of mathematical models of cellular processes represents an essential methodology in Systems Biology, complementary to conventional experimental biology, to understand the emerging behavior of biological systems. The simulation of the dynamics of these models, usually required for computational studies such as parameter estimation or sensitivity analysis, can become burdensome if the corresponding biochemical reaction network is characterized by hundreds or thousands of different molecular species and reactions. In this work, we introduce a novel GPU-powered fine-grain deterministic simulator of large-scale models of biochemical reaction networks, and test its computational performances on a set of randomly generated synthetic models of increasing size. We show that our parallel simulator, running on a GPU Nvidia GeForce GTX Titan Z, outperforms the sequential version, running on a CPU Intel i7-4790K 4.00GHz, achieving up to $7.8\times$ speed-up.

1 Scientific Background

Thanks to the availability of efficient high performance computing architectures, *in silico* investigation of biological systems can nowadays be strongly supported by effective computational strategies, which allow to achieve an accurate understanding of the behavior and the emerging properties of biochemical reaction networks (BRNs).

Given a stochastic or deterministic model of a BRN, typical computational methods used in Systems Biology—e.g., parameter estimation or identifiability, parameter sweep analysis, sensitivity analysis, reverse engineering [2]—usually require the execution of large numbers of simulations, possibly resulting in prohibitive running times on Central

Processing Units (CPUs). In addition, when the BRNs are characterized by a large number of different molecular species or reactions, these computational tasks can become excessively burdensome.

To overcome these limitations, Graphics Processing Units (GPUs) can be adopted as an alternative approach to classic parallel architectures (e.g., computer clusters) for the parallelization and the speed-up of the simulation of large-scale models of BRNs. Indeed, GPUs are cheaper than classic CPU cluster architectures, as they allow the execution of demanding computational analyses also on a standard desktop computer, and they also provide an effective mean to sustainable computing by drastically reducing the power consumption.

Anyway, the development of methods that fully exploit the GPU's peculiar architecture is a challenging task, since specific programming skills and a complete redesign of the algorithms are necessary. In order to make GPU-powered simulators available to the Systems Biology community, different GPU-based versions of the most efficient numerical integration methods have been introduced so far. Specifically, both *coarse-grain* and *fine-grain* parallel implementation have been developed. On the one hand, coarse-grain parallelization allows to simultaneously run a massive number of simulations of the same model, each one characterized by a different parameterization; on the other hand, fine-grain parallelization permits to distribute all the calculations required by a single simulation on the cores of the GPU.

An example of coarse-grain parallelization of deterministic simulations of BRNs was presented by Ackermann *et al.* [1]. In this work, a SBML model of a biochemical system is automatically converted into CUDA code, compiled and linked with the parallel simulator and the host code, producing a final executable file ready to simulate the dynamics. The performances of this method were assessed with a NVIDIA GeForce 9800 GX2, which shown a speed-up between $28\times$ to $63\times$ with respect to a CPU Xeon 2.66 GHz. Following a similar approach, Nobile *et al.* presented a parallel simulator named cupSODA [9], which allows to automatically generate the system of ODEs and the corresponding Jacobian matrix from a reaction-based mechanistic model of a biological system, described according to the mass-action kinetics. Differently from Ackermann's solution based on Euler's method, cupSODA relies on the LSODA numeric integration algorithm, achieving a higher accuracy in the simulation of the dynamics, and accelerating the computation in case of systems characterized by stiffness. cupSODA allows a $86\times$ speed-up with respect to COPASI, used as a reference CPU-based LSODA simulator. Another strategy to accelerate deterministic simulations by means of LSODA was presented by Zhou *et al.* [10]. This simulator, named cuda-sim, performs just in time compilation by converting a SBML model into CUDA code, and allows to achieve a $47\times$ speed-up with respect to a CPU implementation written in Python.

Differently from the solutions previously presented in literature, in this work we introduce a novel GPU-based simulator, named LASSIE (LArge-Scale SImulator), specifically designed to accelerate *fine-grain* deterministic simulations of *large-scale* mechanistic models of BRNs.

2 Materials and Methods

In this work we assume that a mechanistic model of a BRN is specified by giving the set of the N different molecular species $\{S_1, \ldots, S_N\}$ that are involved, either as reactants or products, in a set of M biochemical reactions $\{R_1, \ldots, R_M\}$. The amount (concentration) of species S_i is denoted by X_i, for $i = 1, \ldots, N$, while the kinetic constant associated with reaction R_j is denoted by k_j, for $j = 1, \ldots, M$.

The system of ordinary differential equations (ODEs)—which describes how the concentration of each species occurring in the BRN varies in time—can be easily derived by assuming the law of mass action. This basic biochemical kinetic law states

that, in a dilute solution, the rate of an elementary reaction (i.e., a reaction with a single mechanistic step) is proportional to the product of the concentration of its reactants raised to the power of the corresponding stoichiometric coefficient.

As an example, consider the BRN described by 3 reactions among 4 species:

$$S_1 + S_2 \underset{k_2}{\overset{k_1}{\rightleftarrows}} S_3 \overset{k_3}{\rightarrow} S_4, \tag{1}$$

and let us derive the ODE for species S_3, that is,

$$\frac{dX_3}{dt} = k_1 X_1 X_2 - k_2 X_3 - k_3 X_3. \tag{2}$$

Eq. 2 is given by the sum of the rates of production and degradation of S_3, determined by multiplying the concentrations (raised to the power of 1, in this case) of all species involved in Eq. 1 together with the corresponding kinetic constants. Note that, in this ODE, the right-hand side consists of 3 terms, corresponding to the number of reactions where S_3 is involved in. By repeating the same procedure for all species in the BRN, we can derive the whole system of ODEs and proceed with its numerical integration by using some ODE solver.

Briefly, LASSIE implements the following workflow:

1. given as input a mechanistic model of a BRN, it automatically derives the system of ODEs, as described above;

2. it determines an ordering of the various ODEs;

3. it solves the systems of ODEs by exploiting the Runge-Kutta (RK4) method.

The novel feature of this tool consists in the ODE-ordering step. The rationale behind this step is twofold. On the one hand, if the BRN consists in a number of species and a number of reactions that are in the order of hundreds of thousands, than also the corresponding system of ODEs will be very large (measured both in the number of ODEs and in the number of terms per ODE). In this case, standard numerical integration methods on CPU might require a long running time, therefore demanding alternative parallel solutions, as general-purpose GPU computing. On the other hand, to implement fast numerical integration methods on GPUs, one should also take into account their peculiar architecture.

To better explain this issue, it is worth considering that GPUs are based on a SIMD (Same Instruction Multiple Data) programming model, combined with multi-threading: all threads execute the same code, accessing different memory areas. In LASSIE, this feature is used to launch multiple threads, so that each thread solves the ODE related to a specific molecular species, by using the RK4 method. All threads do not run simultaneously, but are organized in blocks, which are distributed on the available streaming multiprocessors (SMs). SMs organize the threads within a block into sub-groups of 32 threads, named warps, which are executed in a lockstep. However, GPUs allow the temporary divergence of the execution flow of warps, so that a part of the threads can execute different portions of code. When this situation occurs (e.g., due to an IF/THEN/ELSE statement), a part of the threads get stalled, waiting for reconvergence. Although this mechanism provides the developer with the freedom of temporarily abandoning the SIMD paradigm, it can potentially lead to the complete serialization of a warp execution, strongly impairing the performance. For this reason, conditional branches should be avoided as much as possible. Since all ODEs can have a different

structure, they consider different variables (i.e., concentrations of the molecular species) which might have different exponents, all threads can, in principle, take a divergent thread and lead to serialization.

In order to mitigate this problem, before the numerical integration takes place, LASSIE reorganizes the ODEs according to the structure of their expressions. Specifically, LASSIE sorts the ODEs by their length, that is, according to the number of terms that appear in the right-hand side of each ODE. The goal of this heuristic is to reduce the serialization of warps, increasing the overall performances.

3 Results

The computational performance of LASSIE was evaluated by simulating a set of synthetic models of BRNs of increasing size, generated with the methodology used in [8]. These BRN models are characterized by a number of species N and of reactions M equal to $(64 \times 64), (128 \times 128), (256 \times 256), (512 \times 512), (704 \times 704), (960 \times 960), (1216 \times 1216), (1472 \times 1472), (1728 \times 1728), (1984 \times 1984), (2240 \times 2240), (2496 \times 2496), (2752 \times 2752), (3008 \times 3008), (5760 \times 5760), (10000 \times 10000)$. For each model, the kinetic constants were randomly sampled with uniform distribution in $(0, 1)$.

Figure 1 shows the running time necessary to carry out a deterministic simulation of the various synthetic models by using a CPU Intel i7-4790K 4.00GHz, a dual-GPU Nvidia GeForce GTX Titan Z equipped with $2 \times 2880 = 5760$ CUDA cores and 12GB of memory, and a Nvidia Tesla K20c equipped with 2496 CUDA cores and 5GB of memory. As shown in the plot, the running time of the CPU linearly increases with the size of the models. On the contrary, both GPU video cards are characterized by an almost constant running time, irrespective of the size of the models when the number of ODEs is fewer than the available cores, as shown by the tailored test with 10000 chemical species. In particular, the GeForce GTX Titan Z allows to obtain the best results—thanks to its higher number of CUDA cores with respect to the Tesla K20c—with a speed-up of $7.8\times$ with respect to the CPU. This is due to the fact that ODEs are outnumbered by the available cores, so that calculations can be completely executed in a parallel fashion, without any stalling due to blocks scheduling. It is worth noting that the break-even between the CPU and the GeForce GTX Titan Z is observed when the number of ODEs to be integrated is around 1200.

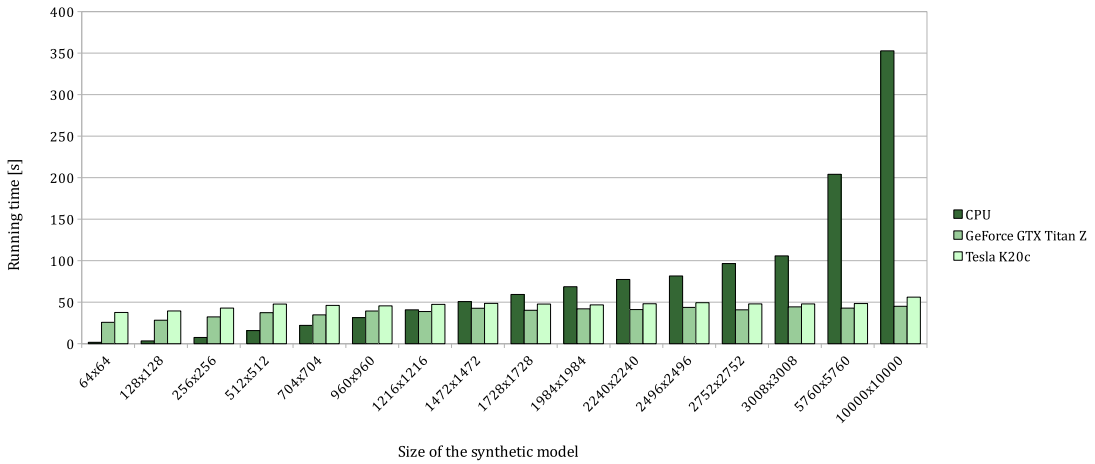

Figure 1: Comparison of the running times achieved by LASSIE on CPU Intel i7-4790K 4.00GHz, Nvidia GeForce GTX Titan Z and Nvidia Tesla K20c for the simulation of synthetic models of increasing size, having a number of species and of reactions $N \times M$ as specified on the x-axis.

4 Conclusion

We compared the performances of CPU and GPU to execute fine-grain deterministic simulations of the dynamics of large-scale BRNs of synthetic models. Our results highlight that our GPU-powered tool outperforms the CPU to simulate mechanistic models consisting in more than 1200 ODEs. However, the speed-up could be smaller by using a multi-threaded CPU simulator: we plan to exploit OpenMP, in the future, for a fair comparison between the architectures. We also plan to replace the ODE-ordering step with a more sophisticated clustering technique, that we expect will further reduce the serialization of threads execution [6]. To be more precise, the ODEs clustering organizes ODEs with similar structures within the same warp, thus avoiding as much as possible the conditional branching, which impairs the performance of the simulator. Second, we plan to optimize the GPU code in order to fully exploit the GPU memory hierarchy and reduce the accesses to the global memory that strongly affect the performance. Third, we will implement on the GPU the Runge-Kutta-Fehlberg algorithm [7], an alternative numerical integration method characterized by adaptive step-size, which allows better performances and simulation quality with respect to classic RK4.

For a thorough assessment of the computational performance of LASSIE, we plan to apply it to simulate large-scale models of real biological systems, such as the ErbB model presented in [3], which consists in more than 1000 ODEs. In particular, we believe that LASSIE will be especially useful for advanced computational analysis of rule-based models [4]. In addition, considering the ongoing research in developing whole-cell models at a detailed level of molecular description [5], we hope that our tool will provide a parallel solution to simulate real models of ever increasing size.

Finally, we plan to include LASSIE within the GPU-based toolbox that we are currently developing. We expect that this innovative, efficient and usable toolbox will obtain a beneficial adoption by the Systems Biology community, and help researchers to easily predict the still unknown behaviors of biological systems and to design more focused experiments in a very reduced time with respect to standard CPU analyses.

References

[1] J. Ackermann, P. Baecher, T. Franzel, M. Goesele, K. Hamacher. “Massively-parallel simulation of biochemical systems”. Proceedings of Massively Parallel Computational Biology on GPUs, Jahrestagung der Gesellschaft für Informatik e.V, pp. 739–750, 2009.

[2] B.B. Aldridge, J.M. Burke, D.A. Lauffenburger, P.K. Sorger. “Physicochemical modelling of cell signalling pathways”. Nature Cell Biology, vol. 8, no.11, pp.1195-1203, 2006.

[3] W.W. Chen, B. Schoeberl, P.J. Jasper, M. Niepel, U.B. Nielsen, D.A. Lauffenburger, P.K. Sorger. “Input-output behavior of ErbB signaling pathways as revealed by a mass action model trained against dynamic data”. Molecular Systems Biology, vol. 5:239, 2009.

[4] L.A. Chylek, L.A. Harris, C.-S. Tung, J.R. Faeder, C.F. Lopez, W.S. Hlavacek. “Rule-based modeling: a computational approach for studying biomolecular site dynamics in cell signaling systems”. WIREs Systems Biology and Medicine, vol. 6, no.1, pp.13–36, 2014.

[5] J.R. Karr, J.C. Sanghvi, D.N. Macklin, M.V. Gutschow, J.M. Jacobs, B. Bolival Jr., N. Assad-Garcia, J.I. Glass, M.W. Covert. “A whole-cell computational model predicts phenotype from genotype”. Cell, vol. 150, no.2, pp.389–401, 2012.

[6] S. Lei, A.Y. Allidina, K. Malinowski. “Clustering technique for rearranging ODE systems”. In: M. G. Singh, A. Y. Allidina, B. K. Daniels eds. “Parallel Processing Techniques for Simulation” pp. 31–43, Springer US, 1986.

[7] J.H. Mathews, K.K. Fink. “Numerical Methods Using Matlab”, 4th Edition, Prentice-Hall Inc., Upper Saddle River, New Jersey, USA, 2004.

[8] M.S. Nobile, P. Cazzaniga, D. Besozzi, D. Pescini, G. Mauri. “cuTauLeaping: A GPU-powered tau-leaping stochastic simulator for massive parallel analyses of biological systems”. PLoS ONE, vol. 9, no.3:e91963, 2014.

[9] M.S. Nobile, D. Besozzi, P. Cazzaniga, G. Mauri. “GPU-accelerated simulations of mass-action kinetics models with cupSODA”. Journal of Supercomputing, vol. 69, no.1, pp.17–24, 2014.

[10] Y. Zhou, J. Liepe, X. Sheng, M.P.H. Stumpf, C. Barnes. “GPU accelerated biochemical network simulation”. Bioinformatics, vol. 27, pp. 874-876, no. 6, 2011.