Enhancing the efficiency of animal-alternative in-silico drug cardiotoxicity prediction through CUDA-based parallel processing

Iga Narendra Pramawijaya[1*], Aroli Marcellinus[1] , Ali Ikhsanul Qauli[1] , Ki Moo Lim[1]
Department of IT Convergence Engineering, Kumoh National Institute of Technology, Korea
*iga@kumoh.ac.kr

Abstract

Cardiovascular diseases (CVDs) remain the leading cause of mortality worldwide. Current drug discovery pipelines face a significant challenge: the heavy reliance on animal testing. This "species translation" issue can lead to misleading results. In response to these limitations, in-silico (computer-based) methods for drug cardiotoxicity prediction have emerged as promising alternatives. In this research, we propose a novel approach to address the computational bottleneck associated with the calculation of complex in silico models under multi-sample scenario, by leveraging CUDA (Compute Unified Device Architecture)-based parallel processing on GPUs. This approach has the potential to significantly accelerate the execution of the model up to 10 times faster when faced with a population sample for drug-testing, making them more efficient and practical for real-world drug discovery applications.

1. Introduction

Cardiovascular diseases are the leading global causes of death, which emphasizes the importance of effective methods for drug discovery. Traditionally, drug cardiotoxicity prediction is achieved using animal testing, which is controversial and has several drawbacks. Modern in-silico or computer-based methods for drug cardiotoxicity prediction show promising results as an animal-alternative alternative. Nevertheless, some of them are computationally inefficient due to large amount of sample it needs to compute. This paper presents an innovative method to optimize currently available in-silico simulation for drug cardiotoxicity testing. Our program acts as computational acceleration for in-silico methods by conducting Nvidia's CUDA (Compute Unified Device Architecture)-based parallel programming on Graphics Processing Units (GPU) [1]. By combining the strengths of in-silico prediction with the computational power of parallel processing, this work aims to contribute to the development of a more efficient, ethical, and reliable drug toxicity evaluation process.

Parallelisation in computational biology is not an entirely new concept. The Cells in Silico (CiS) framework presented by Berghoff et al. (2020) [cite] offers a tool for simulating the growth and development of biological tissues. The modular and parallel design of CiS allows for flexible configuration of different model assumptions, making it applicable to a wide range of research questions. As demonstrated by the example of a 10003 voxel-sized cancerous tissue simulation at sub-cellular resolution, CiS can be used to explore complex biological processes at a high level of detail.

Utilisation of GPU in biological cell computing has been explored in previous researches. One of them is from Miguel, et al [cite] in 2020. Miguel, et al. explored an adaptive parallel simulator to solve performance loss in massive parallel membrane computing devices known as membrane systems or P systems. The paper demonstrates the effectiveness of this approach by extending an existing simulator for Population Dynamics P systems. Experimental results show that this adaptive simulation can significantly improve performance, up to 2.5x on both GPUs and multicore processors.

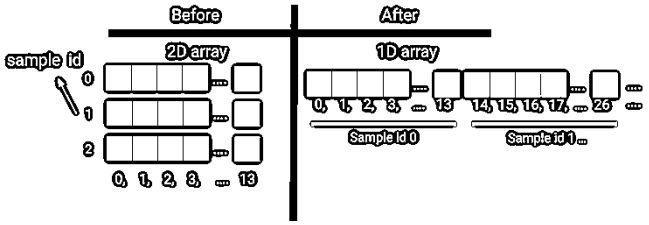

Figure 3. 1 Main difference in variable storing paradigm after CUDA-parallelisation

Related to drug toxicity and discovery, other researchers tried to approach and optimise drug development process using parallel computing approach as well. Previously, McIntosh-Smith et, al. developed a in-silico drug screening method on multiple core processors. McIntosh-Smith et, al. developed BUDE (Bristol University Docking Engine), a drug discovery tool, simulating molecular docking. To speed up calculations on powerful processors with multiple cores, BUDE has been adapted to work with OpenCL, a common language for parallel programming [cite]. As a result, McIntosh-Smith et, al. achieved of 46% at peak, or 1.43 TFLOP/s on a single Nvidia GTX 680.

Barth et, al. developed a parallelisation on biochemical simulation of metabolic pathways in their high level computational simulation. This method allows Barth et, al. to run simulations with more complex models, featuring a greater number of chemicals and reactions. Hence, Barth et, al. can achieve more realistic, lifelike outcomes while using less computing time [cite].

2. Method

a. Simulation Protocol

This research builds CUDA-based parallelisation model upon existing human cardiac electrophysiology model proposed by O'Hara et. al. [2] to enhance its multi-sample calculation performance. We will also leverages the results from O'Hara et. al. for validation. Our focus is on drug effects at the cellular level, requiring 7 IC50 and 7 Hill coefficient [4] per sample. Key outputs include drug toxicity biomarkers and time-series data from each simulated channel, both in CSV format,

valuable for future drug discovery efforts.

b. Solving Ordinary Differential Equation (ODE)

The model built upon algebraic calculations and dynamic functions served in the form of ODEs. We were able to trace the computational procedures and create semi-analytical method to be implemented in the CUDA-based model [2]. This transformation is required to simplify the computational process, and let the parallelisation process focuses on processing multiple samples instead of multiple equations. ODE solver in CUDA-based model is quite similar to Euler style of solving, where the next value determined by adding the previous value with rate of change multiplied by time difference. We also provided function that dynamically update the time difference in each pace to minimize error from this method.

3. Results

a. Developing GPU-based Parallel Process

CUDA-based parallel programming is mainly using C/C++ style but in a different file format. Written in C style, the code is being saved using .cu and .cuh format, these are equivalent to .c and .h respectively. Programming in CUDA accepts C++ style object-oriented programming but in a very limited way. One of the them is the lacking the ease to use vector data type and multi-dimensional arrays in a live-shared (pooling) data. Vector data type encoded as 2D array required in the previous model to store simulation result as the result's length might vary. Multi-dimensional array limitation requires us to convert everything into a single dimensional array (flatten), and use a particular offset number such as in Figure 3.1.

b. Time Performance Comparison

Initially, we need to pick the most optimal GPU core usage per GPU computing block. We used an Nvidia RTX 3080 Ti with 8GB of GPU memory for this optimization trials. Figure 3.2 shows all of block configuration we tried, highlighting extremites and most optimal configuration. In total, we tested the model for 2000 samples. As we are creating the parallelisation based on each sample, each sample has their own "computing core", making the configuration as factors of 2000 (2 blocks = 1000 core/block; 10 blocks = 200 core/blocks, etc.). All of the 2000 samples were simulated for 1000 times (pacing). There are only two configurations that were skipped, 2000 cores/block, and 1000 cores in 2 blocks. These were skipped due to their invalid result, generating only zeroes in the output file. Significant trials bring us to a conclusion where each computing block optimally consist of 20 cores.

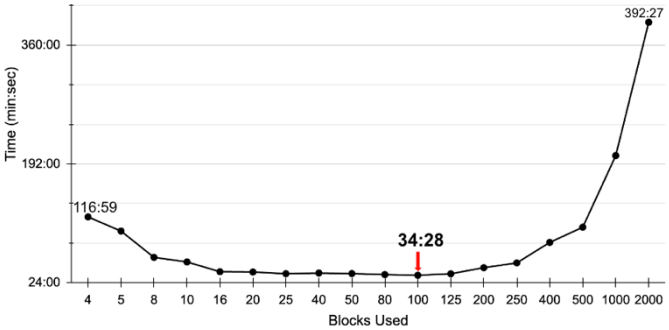

Figure 3. 2 Comparison of Time and Block Used in GPU-based Simulation

In theory, GPU cores are less powerful compared to Central Processing Unit (CPU) cores, making CPU cores as obvious choice for single sample simulation. Upper part of Figure 3.3 shows computing speed of single sample calculation in different resources for 1000 pacing. CPU calculation time should be linear with the sample size and pacing. This linearity makes CPU computation time grow as sample grows. In GPU computing unit, this linearity does not affect the computing speed due to GPU parallelisation. In other words, the time it takes to compute 1 sample will be similar to any number of samples.

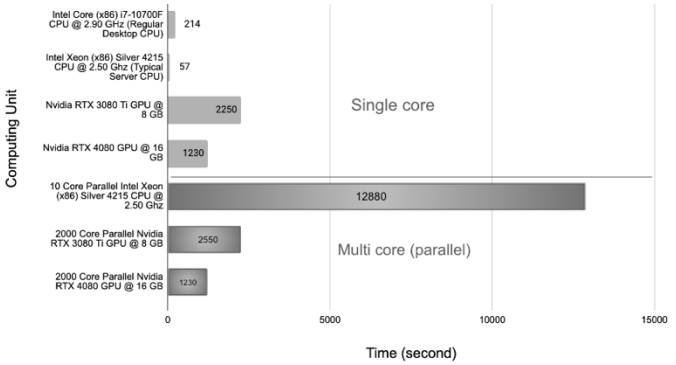

Figure 3. 3 Computational Speed of Different Resources for single and multi core

We compared our CUDA-based GPU approach with a parallel computing CPU using OpenMPI [2]. For 2000 samples simulation, Lower part of Figure 3.3 shows GPU processing achieved up to a 10x speedup, demonstrating significant efficiency gains. All results shown simulated under Bepridil drug effect, with concentration of 99.0 mMol. Experiment shows little to no performance difference between under drug and no-drug simulation.

c. Time-Series Result Validation

Result inaccuracy can cause invalid drug cardiotoxicity prediction. The simplest way to validate the result is by

comparing both of action potential shape from CPU and GPU simulation.

Figure 3.5 shows action potential curve from both CPU and GPU simulation. As shown, little to no difference from both of the result, indicating a valid result from the GPU-based simulation. Promising more efficient in-silico drug cardiotoxicity prediction.

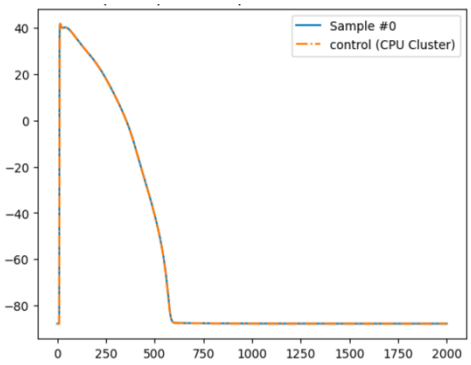

Figure 3. 5 Action Potential Shape of both CPU and GPU Simulation Result

4. Acknowledgements

This research was partially supported by the Ministry of Food and Drug Safety (22213MFDS3922), the NRF (National Research Foundation of Korea) under the Basic Science Research Program (2022R1A2C2006326), and the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2022-2020-0-01612) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation).

5. References

[1] Jason Sanders and Edward Kandrot. 2010. CUDA by Example: An Introduction to General-Purpose GPU Programming (1st. ed.). Addison-Wesley Professional.

[2] O'Hara T, Virág L, Varró A, and Rudy Y (2011) "Simulation of the Undiseased Human Cardiac Ventricular Action Potential: Model Formulation and Experimental Validation". PLoS Comput Biol 7(5): e1002061. https://doi.org/10.1371/journal.pcbi.1002061

[3] R. L. Graham, G. M. Shipman, B. W. Barrett, R. H. Castain, G. Bosilca and A. Lumsdaine, "Open MPI: A High-Performance, Heterogeneous MPI," 2006 IEEE International Conference on Cluster Computing, Barcelona, Spain, 2006, pp. 1-9, doi: 10.1109/CLUSTR.2006.311904.

[4] Mirams G. R., Cui Y., Sher A., Fink M., Cooper J., Heath B. M., et al. (2011). Simulation of multiple ion channel block provides improved early prediction of compounds' clinical torsadogenic risk. Cardiovasc. Res. 91 (1), 53–61. doi:10.1093/cvr/cvr044