Enhancing the efficiency of in-silico drug cardiotoxicity prediction using sample-optimised CUDA-based parallel processing

Iga Narendra Pramawijaya[1*], Aroli Marcellinus[1], Ariyadi[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

Current in-silico drug discovery pipelines face a significant bottleneck, when the sample size increases, the complexity of the calculations grows, resulting in longer processing time. In this research, we propose the updated approach to address the computational bottleneck associated with the calculation of complex in silico models under multi-sample scenario and inter-individual variability, by optimising CUDA (Compute Unified Device Architecture)-based parallel processing even more compared to previous approaches. This approach has the potential to significantly accelerate the execution of the model up to 50 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

Modern in-silico or computer simulation-based methods for drug discovery process such as cardiotoxicity prediction show promising results as an alternative to animal-testing method. However, many in-silico methods encounter significant computational challenges, primarily due to the vast amount of sample data for accurate representation of naturally occurred biological variations. As the sample size increases, the complexity of the calculations grows, resulting in longer processing times and reduced efficiency. This limitation makes it difficult for traditional computational approaches to handle large-scale simulation (such that uses multi-sample scenario or inter-individual variations) within a reasonable timeframe. This research introduces an updated solution to address the computational inefficiencies of current in-silico drug cardiotoxicity simulations. By implementing Nvidia's CUDA (Compute Unified Device Architecture)-based parallel programming on Graphics Processing Units (GPU) [1], our method significantly accelerates overall computational process, enabling faster handling of large-scale simulations. By leveraging the power of parallel processing, this approach not only enhances the in-silico simulation but also ensures that drug toxicity evaluations are both more practical and accurate, paving the way for broader and more ethical applications in real-world drug testing.

2. Method

This study develops a CUDA-based parallelization model based on the human cardiac electrophysiology framework established by O'Hara et al. [2], aiming to improve the efficiency of multi-sample calculations. Additionally, we utilize the findings from O'Hara et al. for validation purposes. The primary focus is on assessing drug effects at the cellular level, which involves the use of 7 IC50, option to add 18 values of conductance variability, and 7 Hill coefficients per sample [4]. The main outputs generated from this approach include crucial drug toxicity biomarkers and time-series data for each simulated channel, all provided in CSV format, offering valuable insights for future drug discovery initiatives.

This computational model is based on algebraic calculations and dynamic functions in the form of Ordinary Differential Equations (ODEs), was adapted to leverage CUDA-based parallel computing [2]. By developing a semi-analytical approach, we transformed the computational process, allowing the parallelization to efficiently handle the processing of multiple samples simultaneously, rather than focusing on solving multiple equations individually. The ODE solver in the CUDA-based model operates similarly to the Euler method, where the next value is calculated by adjusting the previous value with the rate of change and the time difference. Additionally, we implemented a function to dynamically update the time intervals at each step, minimizing errors and further enhancing computational accuracy.

3. Results

CUDA-based parallel programming is built using C/C++ syntax, though the source files are saved in `.cu` and `.cuh` formats, analogous to traditional `.c` and `.h` files. While CUDA supports certain C++ object-oriented features, its capabilities are limited, particularly when managing vector data types and multi-dimensional arrays in shared memory. To overcome this, we restructured the simulation output, which was initially a dynamically sized 2D array. The data was flattened into a single-dimensional array with specific offset indexing to efficiently access elements during computation. This modification enhanced compatibility with GPU architecture, improving memory management.

In addition, we further optimized memory usage, allowing for better parallelization of a larger number of samples. By reducing memory overhead, we were able to parallel-process even more samples simultaneously, maximizing the potential of the GPU's computational resources. This optimization plays a crucial role in handling large-scale simulations while maintaining high efficiency, ensuring that the CUDA-based parallelization can manage larger datasets with minimal performance degradation.

a. Time Performance Comparison

To begin, we needed to determine the most optimal GPU core allocation per computing block. For this optimization trial, we utilized an Nvidia RTX 4090 with 24GB of GPU memory. Overall, we tested the model using 8000 samples.

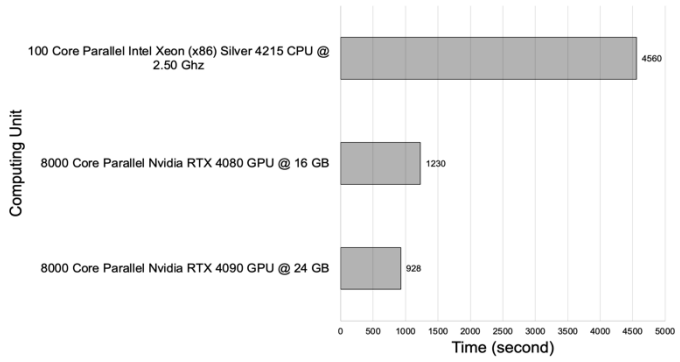

Figure 3. 1 Computational Speed of Each Parallel Computing Hardware

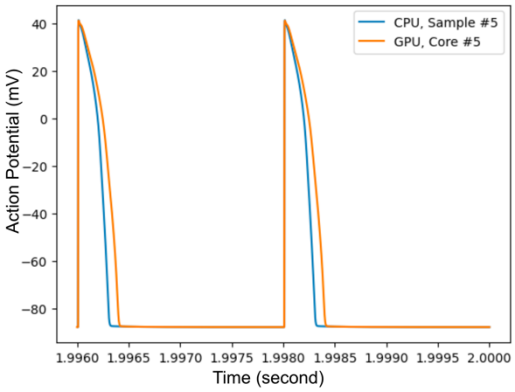

Figure 3. 5 Action Potential Shape of both CPU and Stress-Tested GPU Simulation Result

Since the parallelization was applied per sample, each sample was assigned its own "computing core," with configurations as multiples of 8000 (e.g., 2 blocks = 4000 cores/block; 10 blocks = 800 cores/block, etc.). All 8000 samples were run for 1000 iterations (pacing). Some configurations, such as 8000 cores/block and 4000 cores across 2 blocks, were excluded due to generating invalid results with zero output. Compared to previous approach, we optimise the computing core allocation algorithm. Through extensive testing, we concluded that the most efficient configuration was 32 cores per block.

In theory, GPU cores operate at lower clock speeds, making them inherently less powerful than Central Processing Unit (CPU) cores, which is why CPUs are typically preferred for single-sample simulations. The computation time for CPUs increases linearly with the sample size and pacing, meaning that as the number of samples grows, so does the time require for computation. In contrast, GPU parallelization eliminates this linear growth; the time it takes to compute one sample is nearly the same regardless of how many samples are processed, thanks to its parallel computing architecture.

We compared the performance of our CUDA-based GPU approach to that of a CPU-based parallel computing system using OpenMPI [2]. For a simulation involving 8000 samples, as shown in Figure 3.1, the GPU achieved a speedup of up to 4.91 times compared to a 100-core CPU parallelization and up to 50 times faster than a 10-core CPU setup. Given the high cost of providing 100-core parallelization, 10-core setups are more common in practice, making GPU parallelisation significantly more time-efficient for most simulations. All results were tested under the Bepridil drug effect with a concentration of 99.0 mMol, and the experiments showed no significant difference in performance between drug and no-drug simulations.

b. Time-Series Result Validation

Inaccurate results can lead to incorrect drug cardiotoxicity predictions, so validating the accuracy of the simulation is crucial. A straightforward method for validation is to compare the action potential shapes generated by both the CPU and GPU simulations, particularly when the GPU is subjected to stress tests, such as running in less-than-ideal conditions. In this study, we stressed the GPU by allowing it to operate under excessive heat. Figure 3.2 illustrates the action potential curves from both the CPU and the GPU under stress. As shown, there is minimal deviation between the two, even when the GPU was tested in suboptimal conditions. When both the CPU and GPU simulations were run under ideal conditions, no differences were observed in the action potential shapes. This consistency confirms the accuracy of the GPU-based simulation, supporting its potential for more efficient in-silico drug cardiotoxicity predictions..

4. Acknowledgements

This work was supported by Innovative Human Resource Development for Local Intellectualization program through the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (IITP-2024-RS-2020-II201612).

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