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

Iga Narendra Pramawijaya1\*, Aroli Marcellinus1 , Ali Ikhsanul Qauli1 , Ki Moo Lim1

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 a novel 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 on GPUs compared to previous approches. 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**

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, 1**8 values of conductance variability [cite?]** 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**

**a. Developing GPU-based Parallel Process**

CUDA-based parallel programming relies heavily on a C/C++ syntax, though the source files are saved in the `.cu` and `.cuh` formats, analogous to the traditional `.c` and `.h` file types. While CUDA supports some C++ object-oriented programming features, its implementation is notably restricted. For example, handling vector data types and multi-dimensional arrays in shared memory environments is challenging. In particular, the model's simulation output, originally stored as a dynamically sized 2D array, had to be restructured. Due to the limitations of CUDA in managing multi-dimensional arrays, all arrays were flattened into a single dimension, with specific offset indexing applied to access the correct elements during computation. This transformation enables the parallelization process to efficiently manage large datasets, ensuring compatibility with the GPU's architecture..

**b. 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. Figure 3.2 presents the various block configurations we tested, showcasing both extreme and optimal setups. Overall, we tested the model using 8000 samples. 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. Through extensive testing, we concluded that the most efficient configuration was 32 cores per block.

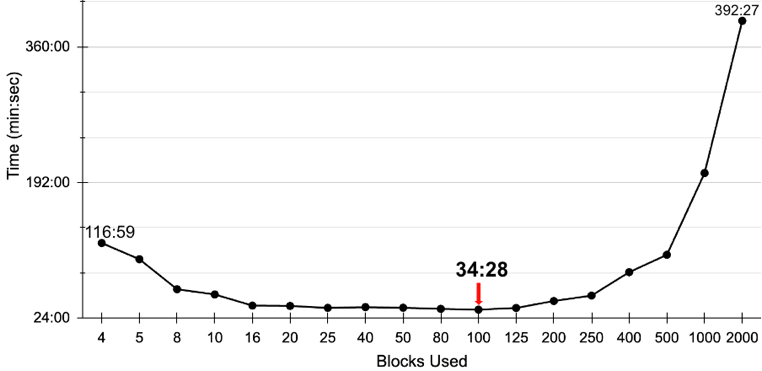


Figure 3. 2 Comparison of Time and Block Used in GPU-based Simulation

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.

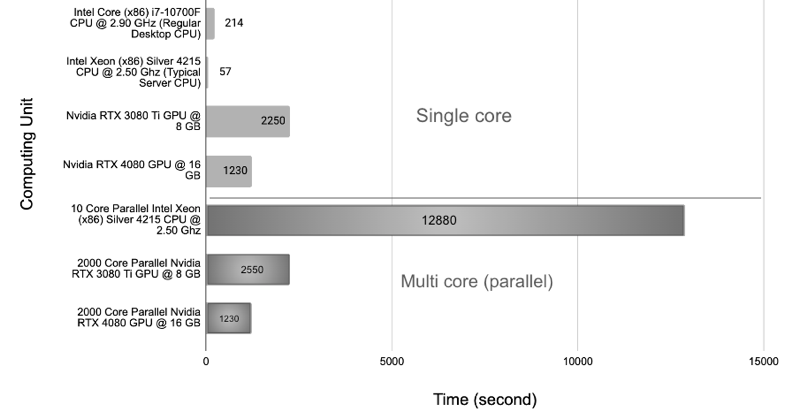


Figure 3. 3 Computational Speed of Different Resources for single and multi core

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.

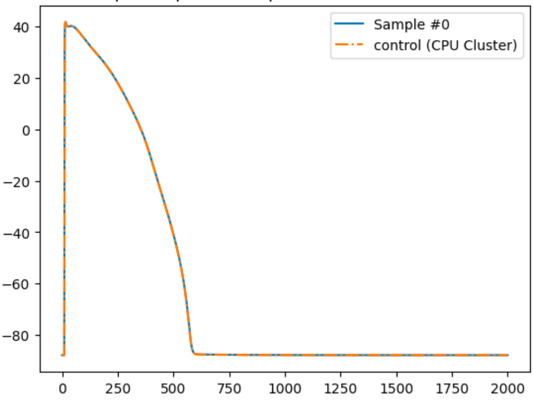


Figure 3. 5 Action Potential Shape of both CPU and GPU Simulation Result

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.

**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