Research Proposal:

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

CUDA기반 병렬처리를 통한 동물대체 인실리코 약물 심독성 예측 효율성 증대

Pramawijaya, Iga Narendra

20236132

Background

- Current drug discovery methods, which rely heavily on animal testing, face **ethical concerns** and **limitations in accuracy** to predict human drug safety.
- In recent years, **in-silico (computer-based) methods** have emerged as a **promising alternative.**
- However, their efficiency is often hindered by the complexity of simulating biological processes, and **large samples that being processed**.

Proposed Method

This research proposes a novel approach to address this challenge by leveraging the power of **CUDA-based parallel processing on Graphics Processing Units (GPUs) to simulate large samples in parallel.** 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.

This research will develop a CUDA-parallel processing simulation based on **O'Hara-Rudy* cardiac cell model.** The research will explain the simulation protocol, the cardiac cell model, and how to solve ordinary differential equation used in the cardiac cell model. Due to different hardware architecture in GPU, some modification will be introduced.

In the end, the whole modifications will be delivered in containerisation using docker.

*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

Simulation Protocol

Overview

Since this research based from O'Hara-Rudy cardiac cell model, simulation protocol will be similar to them, but this research validate our model differently from the previous one. While theirs relied on human data, ours leverages results from O'Hara et al.

Our focus is on how drugs affect individual cells, requiring 14 drug-related parameters per sample, consist of 7 IC50 samples, and 7 Hill coefficient^. We provide two key outputs in a user-friendly format (CSV): drug toxicity markers (biomarkers) and detailed cell response data, both valuable for future drug cardiotoxicity prediction.

^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

Simulation Protocol

How to Solve the ODE & ORd Model

Cardiac cell model uses ordinary differential equations (ODEs) problem that needs to be solved in this research. We were able to trace the computational procedures and create semi-analytical method to be implemented in the CUDA-based model, making the computation faster.

CUDA-based implementation prioritises processing multiple samples over 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.

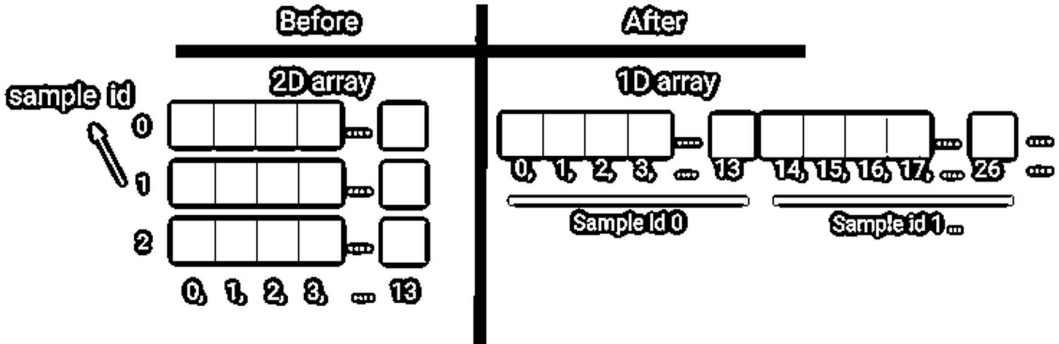
We also provided function that dynamically update the time difference in each pace to minimize error from this method.

Developing CUDA Codes

CUDA programming adopts a C/C++ style, saved in .cu and .cuh formats, akin to .c and .h files. While it allows for C++ style.

Vector data type encoded as 2D array required in the previous model to store simulation result as the result length might vary, this does not apply in CUDA programming.

We proposed a shared pool memory between GPU cores without using 2D array by converting into 1D array with sample id information as offset. Using this approach, each GPU core will handle exactly one sample of cardiac cell parameter set, while maintaining value sharing capability



Containerisation Using Docker^^

Docker containerization is a way to **package an application with all its dependencies** into a standardized unit called a **container**. This container can then be easily run **on any machine** with Docker installed, ensuring the application runs consistently regardless of the environment.

In conclusion, the encapsulation of all code within Docker containers will facilitate seamless modification and progression to subsequent stages of code utilization.

^^Dirk Merkel. 2014. Docker: lightweight Linux containers for consistent development and deployment. Linux J. 2014, 239, Article 2 (March 2014).