Parallel Pair-Wise Interaction for Multi-Agent Immune Systems Modelling

1st Mozhgan K. Chimeh
Dept. of Computer Science
University of Sheffield
Sheffield, UK
m.kabiri-chimeh@sheffield.ac.uk

2nd Peter Heywood
Dept. of Computer Science
University of Sheffield
Sheffield, UK
p.heywood@sheffield.ac.uk

3th Marzio Pennisi
Dept. of Mathematics and Computer Science
University of Catania
Catania, Italy
mpennisi@dmi.unict.it

4th Francesco Pappalardo
Dept. of Drug Sciences
University of Catania
Catania, Italy
francesco.pappalardo@unict.it

5th Paul Richmond
Dept. of Computer Science
University of Sheffield
Sheffield, UK
p.richmond@sheffield.ac.uk

***Abstract*—Agent Based Modelling (ABM), is an approach for modelling dynamic systems and studying complex and emergent behaviour. ABM approach is a very common technique in biological domain due to high demand for a large scale analysis tool to collect and interpret information to solve biological problems. However, simulating large scale cellular level models (i.e. large number of agents/entities) require a high degree of computational power which is achievable through parallel computing methods such as Graphics Processing Units (GPUs). The use of parallel approaches in ABMs is growing rapidly specifically when modelling in continuous space system (particle based). Parallel implementation of particle based simulation within continuum space where agents contain quantities of chemicals/substances is very challenging. Pair-wise interactions are different abstraction to continuous space (particle) models which is commonly used for immune system modelling.**

This paper describes an approach to parallelising the key component of biological and immune system models (pair-wise interactions) within an ABM model. Our performance results demonstrate the applicability of this method to a broader class of biological systems with the same type of cell interactions and that it can be used as the basis for developing complete immune system models on parallel hardware.

***Index Terms*—Agent Based Modeling, GPGPU, High-Performance Computing, Cellular Modelling, Computational modelling, Parallel simulation, FLAME GPU**

I. Introduction

To study and investigate biological systems, a hybrid approach that is the integration of experimental and computational research, is required. This hybrid approach has helped shaping novel hypotheses in research. In silico experiments, a.k.a simulation, attempts to capture the dynamics of the system as an alternative for studying biological systems. With the hybrid approach, the experiments that are not easily doable in a laboratory are achievable [1], [2].

Simulation and modelling has been used by researchers in various scientific domains as a tool to better understand and predict the behaviour of a system. Based on the characteristic of the model, a system can be represented using different design methods. A complex system can be represented as top-down by using sets of equations to model system level behaviour or bottom-up by modelling the individuals with the system as agents. Typically, an Agent Based Model (ABMs) contains an environment with a number of agents (self contained entities) and a set of rules describing their behaviours and interactions. Agent Based Modelling (ABM) is a method of studying behaviours of complex systems [3]. Multi-Agent Simulation (MAS) provides a natural modelling approach as often the individual levels behaviour are well understood. MAS is a common approach to simulate biological systems as it allows individual cells to be tracked throughout the simulation. Within ABM different levels of abstraction can be also applied. Agents such as cells may be modelled as points in continuous space or agents may represent discrete spatial areas containing quantities of chemicals and cells arranged in regular structures (e.g. as a square or hexagonal lattice). Hybrid approaches where discrete spatial areas perform reaction diffusion modelling but have well mixed collections of directly interacting individual cells with monte-carlo (pair-wise interaction) are also common within immune system and more general biological modelling.

Simulating a complex system, such as biological cellular system, as an ABM is computationally expensive when compared with a top down approach . Increasing the scale of the model to achieve natural sizes places a further computational burden which can impede modellers. A feasible solution would be the use of parallel computing resources to contain these requirement and achieve sensible simulation times when scaling complex systems.

Graphics Processing Units (GPUs) are specialised massively parallel processors containing hundreds of arithmetic processing units that can be utilised to achieve significant acceleration for computationally intensive scientific applications. GPUs al-

Authors receive a grant from Horizon 2020 research and innovation programme as well as EPSRC (See acknowledgement)

low a personal computer to be transformed into a personal supercomputer, providing up to 15 Trillion Floating Point Operations per Second (TFLOPS) in consumer hardware (NVIDIA TITAN V). While GPUs are computationally powerful, their hardware design is significantly different from modern CPUs. GPUs are known to be hard to program. One of the challenges of utilising high level of parallel performance using GPUs is the need for the programmer to have considerable knowledge of data parallel algorithm design as well as optimisation skills.

GPUs have been widely used in many scientific research domains to accelerate applications and showed significant computational performance improvements [4]. There are several domain specific studies that use GPUs to implement various complex multi-agent systems [5], [6], [7]. In the majority of these cases GPUs have been used for simulating continuous or discrete space abstractions as apposed to hybrid approaches which are desirable for large scale immune systems simulations.

This paper describes and demonstrates the implementation of hybrid space biological models with parallel monte-carlo pair-wise interactions executing on a GPU architecture. The paper demonstrates performance characteristics for a simplified large scale biological cellular system simulation through a case study of interacting cells which form the basis of many immune system models. Immune system models are a form of a complex biological system which consists of a large number of agents (cells) communicate indirectly through diffusion of chemical substances or directly through connection of chemical receptors [6]. Types of interactions between agents and governed rules makes the model complex enough to be used as a case study to show the viability of using GPUs for other cellular level biological system with the same type of mechanism and complex behaviours.

Within the context of this paper, the model is implemented using the FLAMEGPU framework [8], a flexible large scale Agent Based Modelling environment that enables modeller from diverse scientific domains such as economics, biology and social sciences to easily write agent based models targeting GPUs [9].The model described within this paper consists of a specific approach for describing pair-wise interactions which can be applied more broadly to general cell-cell or cell-environment interactions within an immune system model.

Our case study model is based on an existing work that was implemented by the Universal Immune System Simulator (UISS) framework [10]. The UISS framework models and simulates immune system related pathologies on CPU and in order to demonstrate how conveniently biological cellular level models can be simulated on GPUs, we decided to implement a very common and necessary biological cell behaviour in FLAME GPU. Result from this study shows the applicability of the technique to a broader class of multi-cell biological system.

The rest of this paper is organised as follows: Section II surveys previous studies on the application of GPU in Agent Based Modelling simulation, specifically in the field of biological cell modelling. Section III presents design considerations required to implementing the model with high degrees of model parallelism. Section IV, reports the result of our experimental evaluation. Finally, we draw our conclusions in Section V.

II. Related work

An immune system is an example of a complex system comprising different types of interactions between a variety of cell types. There are various ways to model immune systems. The most common approach is the use of differential equations [11], [12], [13]. Equation based models track the concentration of immune system entities over the time. The equations are sometimes mathematically sophisticated and cannot capture particular aspects of immune system modelling such as locality of responses.

Agent based methods provide ways of representing the heterogeneity of the entities as well nonlinear interactions among agents [14], [15], [16], [17]. In ABM, mobile agents interact with environment or other individuals in continuous or discrete space. There are various existing works on agent based immune system models implemented using different levels of abstractions (continuous space, continuum or hybrid). Agent based Artificial Immune System (AbAIS) [18] framework uses a hybrid architecture where heterogeneous agents evolve over a cellular automata environment. In this framework, agents are modelled using a genetic approach. CAFISS [19] models cell-cell interactions in a grid where each cell has bit string . The scalability of the model using this approach is questionable due to the large overhead caused by the use of separate thread for each cell. Each immune system cell in this approach runs its own thread. Cell-cell communication is done through events.

ImmSim [20] is a framework based on cellular automata where entities interact with other and diffuse through lattice site. In this mode, individuals consider possible interactions based on the given probability rule. The framework has been developed using an interpreted language which limits the system size resulting in small size simulations. Later, parallel version of ImmSim, C-ImmSim [21] was developed with the focus on scalability and performance. C-ImmSim is an advanced immune system simulation based on ImmSimm with added features that allows simulations at the cells and molecules levels.

ImmunoGrid [22] uses C-ImmSim as an underlying framework. It uses grid technologies which allows very large and complex simulation size matching a real size immune system. Simmune [23] is a framework to model cell-cell and cell-molecule interactions where similar to ImmSim, cells do not have states. Simulating complex and detailed interaction using Simmune framework is very computationally expensive. Sentinel [24] is another framework based on the principles of ImmSim with environment is divided to grids and individuals can move between locations.

Jacob, Litorco and Lee [16] presented a swarm agent based 3d model of immune system in continuous space using Breve simulation [25]. Agents move randomly in the continuous space and only interact with those within their specific distance. Note this model is different from all other mentioned above due to the fact that is continuous based. The visualisation and continuous space approach impose constraints on the simulation size [26] which could be improved by employing parallel processing techniques [27]. Generally, simulating large scale complex models is computationally expensive. GPUs have been used to accelerate scientific application and proven to achieve significant performance for computationally problematic cases. There are several studies on the application of GPUs to biological systems [28], [29], [30], [31]. There are several existing works on parallel implementation of the immune system model simulation in continuous space [6], [7], [32]. PI-FLAME [32] uses GPUs to simulate immune system models in continuous space which is less ideal for reaction diffusion modelling and requires detailed modelling on agent space.

In this paper, we are replicating an immune system model previously implemented by UISS [10]; a universal immune system simulator framework. UISS is a hybrid simulator based on the Monte Carlo approach in discrete space where pairwise cell interactions are done based on the probability within the site of interaction and not the actual physical space. The simulator is more computationally efficient than continuous space, however it is inherently serial. Parallelising the model using Monte Carlo approach is challenging due to pairwise interactions.

Note the term hybrid (continuum and agent-based) approach has several meaning in the existing literature [33]. In this context, hybrid approach is where we have particles within continuum.

III. A GPU Implementation of Cell-Cell Interactions within Continuum

We implement a simplified version of the pairwise interaction that exists in human immune systems and almost all of the biological systems. An example of this interaction can be seen between B cells (an immune system cell type that is part of the adaptive immune system and is responsible in production of antibodies) and antigens. Our model uses the FLAME GPU library to map our model description to GPU executable code.

Developed since 2008, FLAME GPU framework is a generalised large scale ABM framework that employs the parallel architecture of Graphic Processing Unit (GPU) to enable real time model interaction and visualisation. FLAME GPU abstracts away the complexity of the GPU architecture from the users (modellers) by providing a high-level modelling syntax-based on a formal state-machine representation. In other words, it allows modellers from any domain to write a model to target GPUs capable of simulating millions of interacting agents/individuals without the need to obtain specialist knowledge typically required to program GPU architectures.

FLAME GPU is a template-based simulation environment that maps formal description of agents into simulation code. Agent representation is based on the concept of a communicating X-Machine which the communication is done via messages. An overview of the features and capabilities of the FLAME GPU simulation platform has been demonstrated through an example in [9].

Fig 1 shows FLAME GPU code generation process which automatically translates a high level model description to optimised GPU code described in a series of code generation templates.

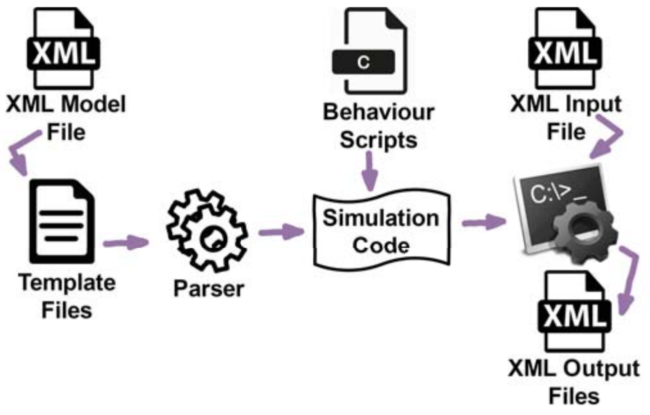

Fig. 1. The FLAME GPU Modelling Process. An XSLT template processor translates a user defined XMML model into simulation code to be linked with the behavioural function scripts to produce a custom simulation executable.

Previous works using FLAME GPU framework, demonstrated that performance gains are easily achievable in large-scale models of continuous space (particle like) agents when compared to the traditional CPU based simulators [34], [35]. In this work, we implement a simplified version of pairwise cell interactions in Immune system models in FLAME GPU. In the simplified version, we are only considering a the complex problem of behaviour within a single lattice site (representing a continuum of well mixed cells) where cells are represented within the continuum as individuals with tracked states (i.e. a hybrid approach). As such no movement between sites has been taken into account however this can be achieved using features of FLAME GPU which have been described extensively in previous publications. In the proposed model, we have two cell types (`cell_A` and `cell_B`), each having a unique identifier and lattice site identifier. `cell_A` agents keep track of their interactions with `cell_B` agents by storing the unique identifier of a `cell_B` agent which they have interacted. Moreover, `cell_B` agents have a `quantity` variable that holds the total number of that type of `cell_B` at the given lattice site.

The two cell types interact based on a probability (i.e. a monte-carlo approach) which is in this case is determined globally rather than per cell type pair based on the hamming distance (or within a fixed radius). During the simulation, each `cell_A` tries to interact with `cell_B` until it successfully interacts within a given iteration (Algorithm 1).

In the FLAME GPU implementation of the model, `cell_B` have two agent functions, called `output` and `update_from_message` and `cell_A` has a single agent function `a_interact_b`. To further simply the model, both

Algorithm 1 Pseducode for pairwise cell interaction implementation

```
for each $cell\_A$ do
  $cell\_A \rightarrow interactionTarget = MAX$
  for each $cell\_B$ do
    if $cell\_B \rightarrow Quantity > 0$ then
      for each $cell\_B \rightarrow Quantity$ do
        $p = interaction\_probability$
        $r = rand()$
        if $r < p$ then
          $cell\_B \rightarrow Quantity - -$
          $cell\_A \rightarrow interactionTarget = cell\_B \rightarrow id$
          $break$
        end if
      end for
    end if
    if $cell\_A \rightarrow interactionTarget \neq MAX$ then
      $break$
    end if
  end for
end for
```

cell agent types have only a single state (which can be trivially extended with FLAME GPU).

Figure 2 shows the process for a single iteration of the simulation. Horizontal dashed line rectangles demonstrate the FLAME GPU function layers. Vertical arrows lines show the process for a single agent state list, beginning at the top and proceeding downwards. Green arrow lines show the relationship between messages_lists (diamond shape) and agent functions (rectangle shape). The total number of function layers in this model is 3.

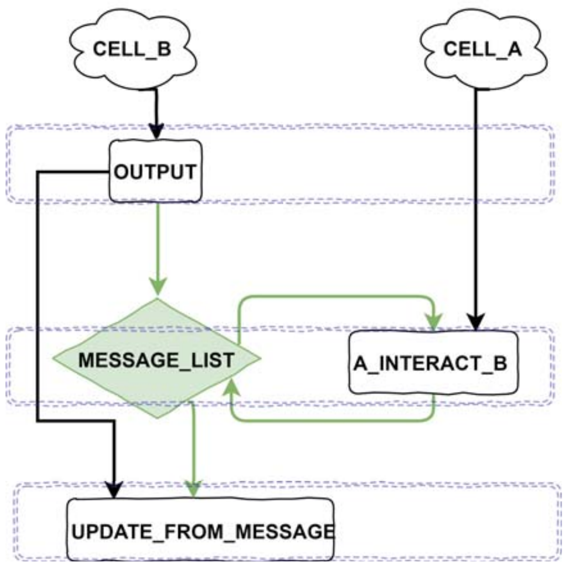

Fig. 2. FLAME GPU state diagram for pairwise cell interaction model. The diagram shows the agents (CELL_A and CELL_B) functions per layer (shown in blue dashed rectangle) and the message type MESSAGE_LIST

In FLAME GPU, the pair-wise interactions between the two agent types are implemented using multiple agent functions (Figure 2) which preserves correct system level behaviour compared to the serial implementation. Below is the list of events happening per simulation step. Within each layer behaviour is parallel, each layer is executed sequentially in turn with a global synchronisation ensuring previous layers have completed:

- Layer 1: Each of the cell_b agents execute the output agent function, which outputs a message of type message_list containing information about the cell_b agent, including the id, latticeSite and quantity. These information are required by cell_a to complete the pairwise interaction.
- Layer 2: Each of the cell_a agents execute the a_interact_b agent function, which reads in the message_list messages. Each agent iterates over the list of messages and for each message (contains information of each cell_b agent), the cell_a attempts to interact with the cell_b, only if it has not yet had a successful interaction. Moreover, the message_list will be updated upon any successful interaction.
- Layer 3: Each of the cell_b agents execute the update_from_message agent function, which reads in the modified message_list variables to update their quantity value.

The serial implementation of the pairwise interaction between cell_a and cell_b is normally performed using a simulated dice roll per cell_b quantity. In our implementation, we performed a single dice roll per cell_b type and compared it to a probability value which at the system level is observed to produce the same (statistically evaluated) behaviour as the serial implementation. This is calculated based on the per_interaction_probability and the quantity of cell_b: $p = 1 - ((1 - per\_interaction\_probability)^{quantity})$. For example, if the per_interaction_probability is equal to 0.1 and quantity = 5, then the probability of at least one interaction occurring is $1 - (0.9^5) = 0.4095$.

The probability test is performed in layer 2 within the a_interact_b agent function. If the test passes, then cell_a agent will attempt to claim a unit of cell_b quantity. However, as many cell_a agents may be attempting to interact with the same cell_b agent at the same time (due to the parallel nature of the FLAME GPU simulator), conflicts in terms of deciding priority of which cells should interact must be resolved by using an *atomic function* which attempts to decrement the quantity value of the cell_b in a single transaction. The additional of this atomic operation is an extension to FLAME GPU for hybrid modelling and ensures race conditions are prevented by directly modifying the message data.

Note the atomic function returns the previous quantity. The non-zero value indicates the successful interaction between cell_a and cell_b. Upon successful interaction, cell_b will no longer attempt to interact. However, the `cell_a` agent which were unsuccessful in interaction, will continue to attempt to interact until there are no more `cell_b` agent to consider.

Atomic instruction to modify the message data is a novel addition to FLAME GPU to allow pair-wise interactions in FLAME GPU. The use of the atomic function to resolve conflicts between competing parallel agents is *non-deterministic*, and depends on order of execution on the hardware. Although, there are other approaches to resolve this sort of conflicts in a deterministic way. e.g. Using a series of recursive message transactions where agents bid to interact [36], the atomic approach is statistically equivalent, simpler in methodology and computationally more efficient.

IV. Results and Discussion

In order to show the feasibility of using GPUs for simulating complex biological cellular level models, we designed a model comprising a subset of common, yet necessary cell interactions in the immune system model that is applicable to a broader class of complex biology model with the same mechanism and type of cell behaviour. More specifically, we chose to model pair-wise interactions between cells in human immune system model and these interactions are difficult to parallelise due to it nature that is a probability based interactions. We have implemented a simplified version using FLAME GPU framework. Additionally, a reference serial CPU version of the same model has been produced to ensure comparable results. The serial implementation has not been optimised (e.g. using vector instructions) but relevant compiler flags were used. The sequential implementation of the model was used as a base line and the overall performance was measured against its serial implementation at varying population sizes. The results are indicative of the performance differences between serial UISS simulator and a FLAME GPU implementation.

Experiments were conducted using NVIDIA Pascal-based GPUs to evaluate our technique. More specifically, all the experiments were performed on a single PC with an Intel i7-4770k quad core hyper-threaded processor (3.50 GHz), 16GB RAM and an NVIDIA TITAN X (Pascal) GPU with 3840 CUDA cores and 12GB of memory. The generated CUDA programs by FLAME GPU[1] are compiled using NVIDIAs CUDA 9.1 compiler, `nvcc`.

Throughout the simulation, the number of cell interactions is recorded, to be aggregated and summarised after the simulation iterations have completed. This is used to verify that the model is behaving as intended and to compare the GPU and CPU implementations. For runs with interaction probabilities less than `1`, we do not expect to achieve the exact same number of interactions per run, due to differences in the random number generators used. Instead the average value over many runs are compared to check for statistically equivalent behaviour. Moreover, the same input parameters were used for both implementations (serial and parallel versions of the model).

[1]The latest FLAME GPU 1.5.0 release has been used as a base for the FLAME GPU implementation.

Each simulation was performed for `50` iterations, and was repeated `3` times to capture average execution times. The time per iteration and time for all iterations are captured and output as a number of milliseconds. In order to demonstrate how the scaling affect the ratio of cell populations, we varied the number of agents for each simulation. The performance of computing the updates (i.e. excluding input and output) was averaged over the 9 runs for the same agent populations.

Figures 3 and 4 show the performance and population results for runs. The X-Axis shows the simulation number representing the unique model parameters over which performance results are averaged. Simulation populations generally increase as x increases (2d data). In other words, simulation number is the different configuration for initial population count of `cell_A` and `cell_B` agents. Figure 3 shows population number for both `cell_A` and `cell_B` agents (left and right Y-Axis) for various simulations. Figure 4 shows the total population (right Y-Axis) and the speedup relative to serial implementation (left Y-Axis). Figure 5 shows the average simulation runtime for both parallel and serial implementations. For lower number of agent population, the CPU (serial implementation) outperforms GPU (parallel implementation) due to device under utilisation.

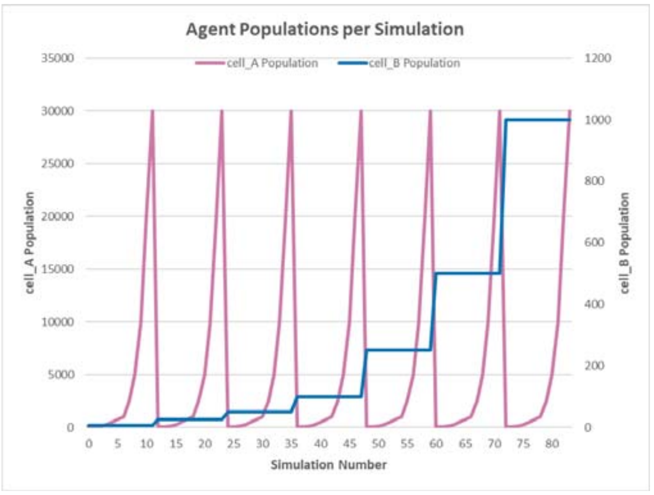

Fig. 3. Agent population per simulation

Across all of the simulation runs, the GPU simulations are on average `28.8x` faster than the CPU equivalent, ranging between `0.06x` the speed of the CPU simulation and up to `210x`. The broad range of performance is attributed to the scale of the simulation. Larger population counts show greater performance improvements, which are naturally more suited to the GPU. This trend continues until the GPU is fully saturated (which can be hundreds of thousands of individuals on modern GPUs).

The saw-tooth nature/pattern seen in the figures is from the varying population sizes. The trend per-saw-tooth is due to overall population, meaning the more `cell_A` agent, the

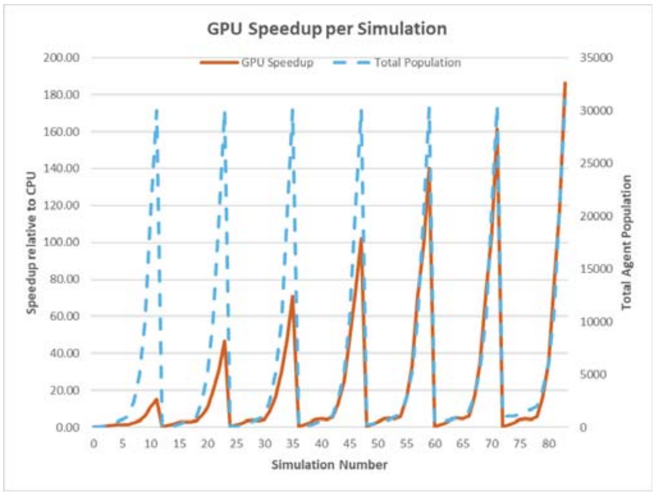

Fig. 4. Speedup relative to serial implementation

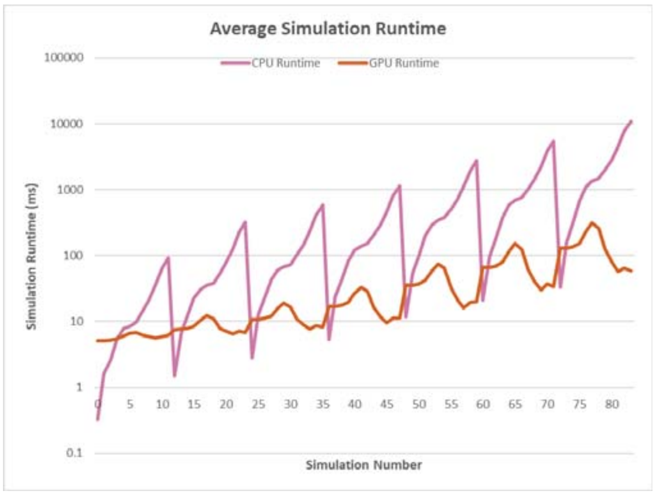

Fig. 5. Average simulation runtime (Log scale)

better device utilisation for the most-computationally expensive task (`cell_A` kernel function (`A_INTERACT_B`) which includes message iteration), amortising the over-head cost of data transfer and kernel launch overhead.

The increasing trend across the pattern of saw-tooths is due to the increase in `cell_B` population (gradually to right). This both increases device utilisation in `cell_B` kernels, and also increases the amount of work done in the interaction kernel (i.e more `cell_B` means more total quantity and therefore more interactions).

V. Conclusion

This paper aimed to prove the feasibility of applying GPU to implement a hybrid pair-wise interaction model representative of an agent based immune system model. We implemented a specific type of cell interactions known as pairwise interaction that is very common in biological cellular level systems. Using FLAME GPU to simulate the simplified model with only two cell agent types with pairwise interactions, we demonstrated that the technique is computationally more efficient than the serial counterpart and demonstrated the addition of a novel atomic based approach for reproducing equivalent serial behaviour. The model demonstrated is applicable to a broader class of biological systems with the same type of cell interactions and can be used as the basis for developing complete immune system models on parallel hardware.

Acknowledgement

Authors of this paper acknowledge support from the STriTuVaD project. The STriTuVaD project has been funded by the EC contract H2020-SC1-2017-CNECT-2, No. 777123. Authors recognize also support by EPSRC fellowship Accelerating Scientific Discovery with Accelerated Computing (EP/N018869/1).

References

[1] F. Chiacchio, M. Pennisi, G. Russo, S. Motta, and F. Pappalardo, "Agent-based modeling of the immune system: Netlogo, a promising framework," in *BioMed research international*, 2014.

[2] H. Kitano, "Systems biology: A brief overview," *Science*, vol. 295, no. 5560, pp. 1662–1664, 2002. [Online]. Available: http://science.sciencemag.org/content/295/5560/1662

[3] P. Siebers and U. Aickelin, "Introduction to multi-agent simulation," *CoRR*, vol. abs/0803.3905, 2008. [Online]. Available: http://arxiv.org/abs/0803.3905

[4] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable Parallel Programming with CUDA," *Queue*, vol. 6, no. 2, pp. 40–53, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1365490.1365500

[5] M. L. A, R. D. B, and K. R. B, "A Framework for Megascale Agent Based Model Simulations on the GPU," 2008.

[6] A. Oliveira and P. Richmond, "Feasibility study of multi-agent simulation at the cellular level with flame gpu," *Queue*, 2016. [Online]. Available: https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS16/paper/view/12797

[7] P. Richmond, D. Walker, S. Coakley, and D. Romano, "High performance cellular level agent-based simulation with FLAME for the GPU," *Briefings in Bioinformatics*, vol. 11, no. 3, p. 334, 2010. [Online]. Available: + http://dx.doi.org/10.1093/bib/bbp073

[8] P. Richmond and D. Romano, "Template-Driven Agent-Based Modeling and Simulation with CUDA," pp. 313–324, Feb. 2011.

[9] P. Richmond and M. K. Chimeh, "Flame gpu: Complex system simulation framework," in *2017 International Conference on High Performance Computing Simulation (HPCS)*, July 2017, pp. 11–17.

[10] F. Pappalardo, M. Pennisi, and S. Motta, "Universal immune system simulator framework (UISS)," in *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology, BCB 2010, Niagara Falls, NY, USA, August 2-4, 2010*, 2010, pp. 649–650. [Online]. Available: http://doi.acm.org/10.1145/1854776.1854900

[11] S. Forrest and C. Beauchemin, "Computer immunology," *Immunological Reviews*, vol. 216, no. 1, pp. 176–197. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1600-065X.2007.00499.x

[12] D. S. Jones, M. J. Plank, and B. D. Sleeman, *Differential equations and mathematical biology*. CRC Press, 2010.

[13] B. ksendal, *Stochastic differential equations an introduction with applications*. Springer, 2013.

[14] L. Zhang, Z. Wang, J. A. Sagotsky, and T. S. Deisboeck, "Multiscale agent-based cancer modeling," *Journal of Mathematical Biology*, vol. 58, no. 4, pp. 545–559, Apr 2009. [Online]. Available: https://doi.org/10.1007/s00285-008-0211-1

[15] C. M. Macal and M. J. North, "Tutorial on agent-based modeling and simulation part 2: How to model with agents," in *Proceedings of the 2006 Winter Simulation Conference*, Dec 2006, pp. 73–83.

[16] C. Jacob, J. Litorco, and L. Lee, "Immunity through swarms: Agent-based simulations of the human immune system," in *Artificial Immune Systems*, G. Nicosia, V. Cutello, P. J. Bentley, and J. Timmis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 400–412.

[17] A. Siddiqa, M. Niazi, F. Mustafa, H. Bokhari, A. Hussain, N. Akram, S. Shaheen, F. Ahmed, and S. Iqbal, "A new hybrid agent-based modeling amp; simulation decision support system for breast cancer data analysis," in *2009 International Conference on Information and Communication Technologies*, Aug 2009, pp. 134–139.

[18] C.-M. Ou, C. R. Ou, and Y.-T. Wang, *Agent-Based Artificial Immune Systems (ABAIS) for Intrusion Detections: Inspiration from Danger Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 67–94. [Online]. Available: https://doi.org/10.1007/978-3-642-35208-9_4

[19] J. C. Tay and A. Jhavar, "Cafiss: A complex adaptive framework for immune system simulation," in *Proceedings of the 2005 ACM Symposium on Applied Computing*, ser. SAC '05. New York, NY, USA: ACM, 2005, pp. 158–164. [Online]. Available: http://doi.acm.org/10.1145/1066677.1066716

[20] R. Puzone, B. Kohler, P. Seiden, and F. Celada, "Immsim, a flexible model for in machina experiments on immune system responses," *Future Generation Computer Systems*, vol. 18, no. 7, pp. 961 – 972, 2002, selected papers from CA2000 (6th Int. Workshop on Cellular Automata of IFIP working group 1.5, Osaka, Japan, Aug. 21-22, 2000) and ACRI2000 (4th Int. Conf. on Cellular Automata in Research and Industry, Karlsruhe, Germany, Oct. 4-6, 2000). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X02000754

[21] M. Bernaschi and F. Castiglione, "Design and implementation of an immune system simulator," *Computers in Biology and Medicine*, vol. 31, no. 5, pp. 303 – 331, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010482501000117

[22] M. Halling-Brown, F. Pappalardo, N. Rapin, P. Zhang, D. Alemani, A. Emerson, F. Castiglione, P. Duroux, M. Pennisi, O. Miotto, D. Churchill, E. Rossi, D. S. Moss, C. E. Sansom, M. Bernaschi, M.-P. Lefranc, S. Brunak, O. Lund, S. Motta, P.-L. Lollini, A. Murgo, A. Palladini, K. E. Basford, V. Brusic, and A. J. Shepherd, "Immunogrid: towards agent-based simulations of the human immune system at a natural scale," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1920, pp. 2799–2815, 2010, exported from https://app.dimensions.ai on 2018/09/24. [Online]. Available: https://app.dimensions.ai/details/publication/pub.1001411341 and http://rsta.royalsocietypublishing.org/content/roypta/368/1920/2799.full.pdf

[23] M. Meier-Schellersheim and G. Mack, "Simmune, a tool for simulating and analyzing immune system behavior," *CoRR*, vol. cs.MA/9903017, 1999. [Online]. Available: http://arxiv.org/abs/cs.MA/9903017

[24] M. J. Robbins and S. M. Garrett, "Evaluating theories of immunological memory using large-scale simulations," in *Artificial Immune Systems*, C. Jacob, M. L. Pilat, P. J. Bentley, and J. I. Timmis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 193–206.

[25] J. Klein, "Breve: A 3d environment for the simulation of decentralized systems and artificial life," in *Proceedings of the Eighth International Conference on Artificial Life*, ser. ICAL 2003. Cambridge, MA, USA: MIT Press, 2003, pp. 329–334. [Online]. Available: http://dl.acm.org/citation.cfm?id=860295.860347

[26] N. Fachada, V. V. Lopes, and A. C. Rosa, "Agent based modelling and simulation of the immune system : a review," 2007.

[27] J. S. Meredith, S. R. Alam, and J. S. Vetter, "Analysis of a computational biology simulation technique on emerging processing architectures," in *2007 IEEE International Parallel and Distributed Processing Symposium*, March 2007, pp. 1–8.

[28] R. M. DSouza, "Sugarscape on steroids : Simulating over a million agents," in *Proceedings of the agent2007 conference*, 2007.

[29] P. Richmond, S. Coakley, and D. Romano, "Cellular level agent based modelling on the graphics processing unit," in *2009 International Workshop on High Performance Computational Systems Biology*, Oct 2009, pp. 43–50.

[30] R. M. D'Souza, M. Lysenko, S. Marino, and D. Kirschner, "Data-parallel algorithms for agent-based model simulation of tuberculosis on graphics processing units," in *Proceedings of the 2009 Spring Simulation Multiconference*, ser. SpringSim '09. San Diego, CA, USA: Society for Computer Simulation International, 2009, pp. 21:1–21:12. [Online]. Available: http://dl.acm.org/citation.cfm?id=1639809.1639831

[31] K. Pietak and P. Topa, "Towards multi-agent simulations accelerated by gpu," in *Parallel Processing and Applied Mathematics*, R. Wyrzykowski, J. Dongarra, E. Deelman, and K. Karczewski, Eds. Cham: Springer International Publishing, 2018, pp. 456–465.

[32] S. Tamrakar, P. Richmond, and R. M. D'Souza, "Pi-flame," *Simulation*, vol. 93, no. 1, pp. 69–84, Jan. 2017. [Online]. Available: https://doi.org/10.1177/0037549716673724

[33] B. Franz and R. Erban, "Hybrid modelling of individual movement and collective behaviour," pp. 129–157, 2013.

[34] R. Chisholm, P. Richmond, and S. Maddock, *A Standardised Benchmark for Assessing the Performance of Fixed Radius Near Neighbours*. Cham: Springer International Publishing, 2017, pp. 311–321. [Online]. Available: https://doi.org/10.1007/978-3-319-58943-5_25

[35] P. Heywood, S. Maddock, J. Casas, D. Garcia, M. Brackstone, and P. Richmond, "Data-parallel agent-based microscopic road network simulation using graphics processing units," *Simulation Modelling Practice and Theory*, vol. 83, pp. 188 – 200, 2018, agent-based Modelling and Simulation. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1569190X17301545

[36] P. Richmond, "Resolving conflicts between multiple competing agents in parallel simulations," in *Euro-Par 2014: Parallel Processing Workshops*, L. Lopes, J. Žilinskas, A. Costan, R. G. Cascella, G. Kecskemeti, E. Jeannot, M. Cannataro, L. Ricci, S. Benkner, S. Petit, V. Scarano, J. Gracia, S. Hunold, S. L. Scott, S. Lankes, C. Lengauer, J. Carretero, J. Breitbart, and M. Alexander, Eds. Cham: Springer International Publishing, 2014, pp. 383–394.