The University of York Department of Computer Science

Submitted in part fulfilment for the degree of MEng.

Parallel Programming Tools for Exploring Immune System Development

Oliver Binns

18th March 2018

Supervisor: Dr. Fiona Polack, Dr. Kieran Alden

Number of words = 3718, as counted by `wc -w`.
This includes the body of the report only.

Abstract

More powerful computers are paving the way for realistic simulations of previously underexplored complex biological systems. As advancements in computing tend towards parallelism and distributed systems, increasingly realistic simulations must take full advantage of this change in order to be computed in a reasonable time period.

Some significant problems, including a significant computer science skills shortage, are slowing down the rate of progress in this area.

In this project, I will propose model-driven engineering (MDE) as a solution to several of these problems, explore some of the current MDE technologies for creating simulations and propose further advancements to these tools for the future.

Acknowledgements

I would like to my supervisors, Fiona Polack and Kieran Alden for their support and guidance throughout this project. I would also like to thank the Sheffield FLAME GPU development team for their help in utilising the latest and pre-released features of FLAME GPU.

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

1.1 Project Overview

TALK ABOUT MODELING AND THE DESIRE FOR
Faster AND More Accessible Simulation

1.2 Motivation

The motivation for this project stems mainly from the new knowledge that could be gained from the ability to create efficient parallel biological simulations more easily.

If hypotheses for *drug testing* can be more easily and extensively tested in silico, this could significantly improve the process by which drugs are developed. New drugs and cures could be developed more quickly, allowing these to be available more quickly. The significant financial burden of drug development, which is estimated to be around $2.9bn[1], could be reduced, freeing up heavily contested funds for additional research. The use of animal testing could be reduced, and potentially eliminated completely in the long term.

1.3 Project Aims

In summary, the aims of this project are to

1. Provide a thorough review of the use of simulations as part of Computational Biology, the advantages they provide and the problems that must be overcome for their mass adoption.

2. Develop a parallel implementation of an existing simulation of Peyer's Patch development and explore any speed increases that can be produced using General Purpose GPU programming.

3. Establish a firm grounding for the future development of new tools to allow fast, parallel simulations of biological systems to be easily created by non-technical users.
    a) Explore the findings of the new implementation and discuss how these can be generalised to new simulations.
    b) Discuss techniques for allowing non-technical users to easily create formal models that can be transformed into new simulation implementations.

1.4 Statement of Ethics

This project was conducted in accordance with the University of York's code of practice on ethics. This project does not involve human participants, so guidelines on informed consent and confidentiality will be met. No confidential medical data or personal information has been used during the course of the project development. This project has involved no direct animal participation.

The simulation of the biological model is for the purpose of developing understanding of applying GPGPU methods to an agent based model of a biological system. It will not be used its current form to publish novel biological findings and does not fully simulate a biological process.

1.5 Report Structure

This report details the work done throughout the project and

Chapter 2.1 gives a general overview of simulations and the benefits and limitations of their use particularly with regard to computational biology.

Chapter 2.2 explores some of the limitations of simulations in additional detail and proposes future solutions for these.

Chapter 3 details the development of an improved, inherently parallel, implementation of PPSim.

2 Literature Review

2.1 Simulation

Simulations are model-based imitations of a system which feature its key characteristics and behaviours. Computer simulations are used in a wide range of disciplines on applications such as video games, medicine, product development and even nuclear weapons.

The models of systems used in simulations may have a varying amounts of abstraction. Simulations used for teaching will likely have models which remove significant amounts of complexity from the system. Simulations used for video games tend to be as realistic as possible as realism has been shown to produce a higher level of immersion[2], a highly desirable attribute of games.

This chapter will explain the benefits and current limitations of using simulations and how these have affected this project.

2.1.1 Benefits

Feasibility

Exploring computer simulations is often far more feasible than exploring a real world environment. Video games are simulations which may allow players to experience scenarios that they may not otherwise get the opportunity to encounter. For example, car racing games are significantly cheaper and safer than real life racing.

Often real-world scientific testing may not be feasible for a number of reasons. Simulating the aerodynamics of new car designs virtually is far quicker and cheaper than creating multiple different prototypes for physical tests. Morality may be a factor, animal testing for cosmetic products or medicine is a good example of this. With nuclear weapons, legality is a key issue, as some weapon testing is banned under a number of global treaties[3], [4]. Finally, some real-world tests may be too dangerous to perform, such as in the case of invasive medical examinations. In all of

these examples, computer simulation is often used to reduce or replace real-world testing.

Reducing Complexity

Reducing complexity through abstraction allows better understanding of the system to be gained as the complexity may initially be overwhelming.

Environmental Control

Computer simulations allow for the environment to be more easily controlled. The ability to adjust external factors and independent variables that may affect the system on demand can be particularly useful. This ability can be used for illustrating why different variables in system processes are important.

Time can be manipulated to show system processes at more reasonable time scales. A chemical process that takes a fraction of a second can be slowed down to ensure that it can be seen.

Simulations allow for additional tests to be easily added at a later date. If the researcher wants to discover how an additional variable is related to system behaviour, it can be added and the simulation can be easily re-run.

Visualisation

One of the biggest benefits of simulation is the ability to graphically visualise a system or its constituent parts. Using simulation for visualisation has number of benefits over attempting to demonstrate real world systems. Several of these benefits translate from the previous two sections- they are often more feasible to explore than a real world environment and featuring a reduced complexity often allows the key concepts to be understood without overwhelming the user.

Simulation can be particularly useful for visualising concepts for education.

Particularly immersive visualisations can also be produced using virtual and augmented reality. The Virtuali-Tee is an educational tool that uses simulation and augmented reality to provide a view at the body's internal organs[5]. Little Journey aims to reduce kids' anxiety about surgery by providing a realistic tour of their hospital ward given by animated cartoon characters[6].

2.1.2 Limitations and Constraints

While simulations seem very useful across a wide number of fields, there are some significant limitations as to where and how they can be used.

Insufficient Domain Knowledge

A simulation is based on a model of a system. A model is an abstraction from reality representing only the necessary key characteristics and behaviours of the system. A lack of knowledge regarding the domain of the simulation is one of the most significant constraints regarding its implementation. If this is the case, the model produced may be incorrect or abstractions may remove necessary detail required for the system to function as expected.

For complex systems, having too many abstractions from the original domain may invalidate the model and produce incorect results[CITATION NEEDED].

Compute Power

Complex models with too few abstractions from their domain may require significant computing power to simulate. Additional abstractions may not be possible as they may invalidate the model. In these cases, cutting edge hardware may be needed for the simulation to be run in an acceptable time.

Powerful hardware is expensive to access, so this may be a significant constraint on the ability to simulate.

Skills Shortage

The previous section discussed reliance that complex simulations have on advanced hardware. However, advanced hardware alone will not necessarily allow a simulation to compute in a reasonable amount of time. Often, and particularly with the increasingly parallel modern architectures, the simulation code must be tailored to take advantage of the computing power available. Efficient parallel programs that do this rely on the availability of experienced programmers. May recent studies have highlighted the existance of significant computer science skills shortages, across the world[7], [8]. These skills shortages may be significantly limiting the possibility for cross-disciplinary work to utilise fast, advanced simulations.

Bugs

As with any form of computer program, mistakes can be made causing bugs to be present in the simulation code. Bugs may cause the simulation to be incorrect meaning any hypotheses and results are based on incorrect data.

This is linked to, but not the same as, having insufficent domain knowledge. Both of these limitations will cause the simulations fail silently, produce incorrect results with no immediately obvious failure[CITE]. However, while these problems are specific to simulation, they are not dissimilar from the issues that can occur from poorly designed real-world testing.

If the simulation needs to be safety-critical, developing it using formal methods and refinement may be a good way to ensure that no bugs are present in the code.

2.2 Improving Simulations

Significant constraints on when simulations can be used were discussed in the previous chapter. In order for simulations to become more widely adopted, some of these constraints must be overcome. In particular, this project will focus on two particular sections of which I believe will have the biggest immediate impact.

Firstly, attempting to overcome the problems provided by the current computing skills shortage. Initial work to help overcome the skills shortage will likely come by providing tools that can automate work done by technical users in order to increase their productivity. Eventually, if this technical work can be fully automated, non-technical users will be able to easily create advanced simulations.

Secondly, simulations need to be able to generate results in a realistic amount of time. As many biological simulations include significant elements of randomness, they must be run a large number of times in order to produce reliable results. This can often take

This chapter will discuss the potential solutions to each of these problems.

2.2.1 Ease of Creation

Ease of creation (and maintenance) is an important feature for future simulation particularly due to the aforementioned computer science skills

shortage (Section 2.1.2).
Biocellion is

Flexible Modelling

Flexible Modelling tools could be a good method for allowing new simulations to be created more easily. Using flexible modelling, non-technical users would be able to create sketches which can be automatically processed by tools into formal models and prototype metamodels[9]. FlexiSketch is a good example of this and provides a good tool for creating models and metamodels for software development[10].

2.2.2 Speed Up

Machine Learning

A solution to the speed problem that has been proposed recently is to use machine learning on a small set of results to produce[11] This has problems in that... likely affected by Standard Machine Learning issues? Bias? Overfitting?
Already an ongoing area of research..

Parallelism

Parallelism fundamentally changes the game and allows computers to keep following Moore's law even has engineers are struggling to make transisters ever smaller and smaller[12]. As modern computers tend further towards parallelism to keep providing the speed-ups that have been inherent in the industry over recent years, new parallel algorithms need developing in order to take full advantage of the computing power available.

Distributed Systems

CPU vs GPU Modern computers provide two main methods for parallelising code. We are building on a previous project[13] which layed the groudwork for this. This previous project outlined the choice between CPU and GPU parallelism and makes the case for exploring GPUs- simplisitically put, this is due to the significantly greater speed ups that can be achieved.

Multiple CPU cores are now commonplace in modern computers, including smartphones. GPU often only found in gaming PCs and consoles This is changing, Apple now includes a custom GPU in many of its mobile devices- metal

OpenGL vs CUDA A previous project compared OpenGL and CUDA

Agent Based Modelling ABM is particularly well suited to parallelism as each agent makes its own decisions. Communication between individual agents can be difficult to implement in parallel but parallel communication is by no means limited to ABM.

Mention Flame (traditional) is an attempt to make simulations more accessible via ABM.

FLAME GPU FLAME (Flexible Large-Scale Agent Modelling Environment) GPU is an extension of the original version of FLAME, where the simulations that are created are compiled down to parallelised CUDA code. This means the simulation can take full advantage of the significant power of modern NVIDIA GPUs.

Relying on a framework such as FLAME Fortunately, as FLAME GPU is an open source project this should be less of an issue.

This framework brings its own advantages. As FLAME maps a model to simulation code, it brings with it many of the advantages of model-driven engineering. The implementations can always be certain of taking advantage of the latest hardware optimisations and portability features that FLAME provides. As FLAME is updated to support new hardware advances, such as *multiple CUDA cores*, existing simulation models may all get faster without requiring major code changes.

FLAME GPU takes input in the form of an XMML model which defines the types of agents that feature in the simulation and their interactions

2.3 Existing Simulations

2.3.1 Agent-Based Modelling

MASON is a multi-agent simulation library for Java.

2.3.2 Biological Simulations

Computational Biology is a relatively new field of study that has been growing signficantly over the last decade[CITATION NEEDED].

Within Biology, simulation is often required as an alternative to invasive medical testing/animal testing

Simulations have even been proposed as a method for exploring a potential set of first principles and mathematics that are specific to biology which could even constitute a new subject- theoretical biology[14].

Genome

Cell Dynamics

2.3.3 PPSim

This project focuses on simulation as a tool for exploring biological systems at cell level. It uses the existing simulation of Peyer's Patch[15] and attempts to use parallel computer architectures in order to speed this simulation up.

Finally, I will propose a new tool, which builds on existing work in order to make this power available to non-technical users.

3 Methods

3.1 The Domain Model

The domain model is taken from the existing sequential simulation of Peyer's Patch development described in Section 2.3.3.

3.2 The Platform Model

OPTIONS:

Custom Code?[13] not well tested less easy to update to support new tools and hardware, new CUDA GPUs Each custom code simulation must be updated separately

model based -> FlameGPU[16] restricted to the framework limitations Host cell creation into the system is a work in progress for next FlameGPU release? helper LTi/LTin agent factory is be required to facilitate this functionality?

A state-based platform model design is already available from the original MASON implementation of this simulation. As FLAME GPU uses state machines for each agent type in its implementation, these have been directly implemented into it.

3.2.1 Testing

Talk about how the model was tested to ensure correctness
Mention missing link in (incorrect) model from Kieran's paper

3.3 Tools

In order to produce a working simulation of Peyer's Patch formation, I have used an unreleased version (1.5) of FLAME GPU. As FLAME GPU is developed openly, this version is freely available online[17].

Git versioning has been used for this project to aid development.

4 Results and Evaluation

4.1 Findings

[18] could be important for evaluating the performance of FlameGPU against original PPSim

4.2 Conclusion

4.3 Further Work

4.3.1 Hardware Availability

4.3.2 Software Generalisibility

Bibliography

[1] J. A. DiMasi, H. G. Grabowski and R. W. Hansen, '"Innovation in the pharmaceutical industry: New estimates of R&D costs"', 2016, pp. 20–33. DOI: 10.1016/j.jhealeco.2016.01.012.

[2] K. Cheng and P. A. Cairns, '"Behaviour, Realism and Immersion in Games"', in *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '05, Portland, OR, USA: ACM, 2005, pp. 1272–1275, ISBN: 1-59593-002-7. DOI: 10.1145/1056808.1056894. [Online]. Available: http://doi.acm.org/10.1145/1056808.1056894.

[3] United Nations Office for Disarmament Affairs. (Oct. 1963). Partial Nuclear Test Ban Treaty, [Online]. Available: http://disarmament.un.org/treaties/t/test_ban (visited on 08/02/2018).

[4] U.S. Department of State. (Dec. 1990). Threshold Test Ban Treaty, [Online]. Available: https://www.state.gov/t/isn/5204.htm (visited on 08/02/2018).

[5] Curiscope Ltd. (). Curiscope, [Online]. Available: https://www.curiscope.co.uk (visited on 05/02/2018).

[6] Little Sparks Hospital Ltd. (). Little Sparks Hospital, [Online]. Available: https://littlesparkshospital.com (visited on 05/02/2018).

[7] Ecorys UK, '"DIGITAL SKILLS for the UK ECONOMY"', Jan. 2016. [Online]. Available: https://www.gov.uk/government/publications/digital-skills-for-the-uk-economy (visited on 15/01/2016).

[8] Microsoft. (Dec. 2012). Investing in American Innovation and the Next Generation, [Online]. Available: https://blogs.microsoft.com/on-the-issues/2012/12/12/investing-in-american-innovation-and-the-next-generation/ (visited on 16/01/2018).

[9] R. F. Paige, A. Zolotas and D. Kolovos, '"The Changing Face of Model-Driven Engineering"', in *Present and Ulterior Software Engineering*, M. Mazzara and B. Meyer, Eds. Cham: Springer International Publishing, 2017, pp. 103–118. DOI: 10.1007/978-3-319-67425-4_7. [Online]. Available: https://doi.org/10.1007/978-3-319-67425-4_7.

[10] D. Wüest, N. Seyff and M. Glinz, '"FlexiSketch: a lightweight sketching and metamodeling approach for end-users"', *Software & Systems Modeling*, Sep. 2017, ISSN: 1619-1374. DOI: 10.1007/s10270-017-0623-8. [Online]. Available: https://doi.org/10.1007/s10270-017-0623-8.

[11] K. Alden, J. Cosgrove, M. Coles and J. Timmis, '"Using Emulation to Engineer and Understand Simulations of Biological Systems"', unpublished.

[12] H. Sutter. (Dec. 2004). The Free Lunch is Over: A Fundamental Turn Toward Concurrency in Software, [Online]. Available: http://www.gotw.ca/publications/concurrency-ddj.htm (visited on 08/02/2018).

[13] P. Drew, '"Parallel Programming Tools for Exploring Immune System Development"', Master's thesis, University of York, Heslington, York, 2017.

[14] D. Noble, '"The rise of computational biology"', *Nature Reviews Molecular Cell Biology*, vol. 3, pp. 459–463, 2002. DOI: 10.1038/nrm810.

[15] K. J. Alden, '"Simulation and Statistical Techniques to Explore Lymphoid Tissue Organogenesis"', PhD thesis, University of York, Heslington, York, 2012.

[16] 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, pp. 334–347, 2010. DOI: 10.1093/bib/bbp073.

[17] FLAMEGPU. (). FLAMEGPU, [Online]. Available: https://github.com/FLAMEGPU/FLAMEGPU/branches (visited on 18/03/2018).

[18] A. Arcuri and L. Briand, '"A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering"', in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11, Waikiki, Honolulu, HI, USA: ACM, 2011, pp. 1–10. DOI: 10.1145/1985793.1985795. [Online]. Available: http://doi.acm.org/10.1145/1985793.1985795.

5 Appendix

5.1 Simulation Parameters

5.2 Cell Data Structures