A High Performance Agent Based Modelling Framework on Graphics Card Hardware with CUDA (Extended Abstract)

Paul Richmond
University of Sheffield, UK
Department of Computer Science
Regent Court, 211 Portobello
Sheffield, S1 4DP
+44 (0) 114 222 1877
P.Richmond@.sheffield.ac.uk

Dr Simon Coakley
University of Sheffield, UK
Department of Computer Science
Regent Court, 211 Portobello
Sheffield, S1 4DP
+44 (0) 114 222 1800
S.Coakley@dcs.shef.ac.uk

Dr Daniela M. Romano
University of Sheffield, UK
Department of Computer Science
Regent Court, 211 Portobello
Sheffield, S1 4DP
+44 (0) 114 222 1900
D.Romano@sheffield.ac.uk

ABSTRACT
We present an efficient implementation of a high performance parallel framework for Agent Based Modelling (ABM), exploiting the parallel architecture of the Graphics Processing Unit (GPU). It provides a mapping between formal agent specifications, with C based scripting, and optimised NVIDIA Compute Unified Device Architecture (CUDA) code. The mapping of agent data structures and agent communication is described, and our work is evaluated through a number of simple interacting agent examples. In contrast with an alternative, single machine CPU implementation, a speedup of up to 250 times is reported.

Categories and Subject Descriptors
I.2.11 [Computing Methodologies]: Distributed Artificial Intelligence - *Languages and structures, Multiagent systems*. I.3.1 [Computer Graphics]: Hardware Architecture - *Graphics processor, Parallel processing.*

Keywords
Agent Based Modelling, Performance, Parallel Algorithms, Graphics Processing Unit, CUDA

1. INTRODUCTION
Agent Based Modelling (ABM) is the simulation of group behaviour from a number of individually autonomous agents. The ability to simulate complex systems from simple rules makes ABM attractive in numerous fields of research including, but not limited to, systems biology, computer graphics and the social sciences. Generally ABM frameworks and APIs are primarily aimed at a single CPU architecture, and whilst they are well developed and offer good agent specification techniques, their inherent lack of parallelism seriously affects performance and the scalability of models. The work in this paper addresses this problem by utilising the GPUs parallel architecture to achieve a massive performance improvement.

Cite as: A High Performance Agent Based Modelling Framework on Graphics Card Hardware with CUDA, (Extended Abstract), Paul Richmond, Simon Coakley, Daniela M. Romano, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 1125–1126

GPU hardware is primarily designed for streaming graphics primitives. Previous General Purpose computation on the GPU (GPGPU) was therefore forced into using graphics libraries as an un-intuitive way to exploit GPU hardware performance. The recent introduction of the CUDA programming language has however improved this situation by allowing direct access to GPU hardware through a familiar 'C'-like language. Despite the relative simplicity of CUDA, optimal performance can only be achieved only with an expert understanding of the underlying hardware architecture. Aside from performance, the advantage of the GPU framework for ABM presented in this paper is that AB modellers do not need an explicit understanding of the GPU architecture. Additionally common functionality such as agent data mapping, communication and birth and death allocation are readily accessible allowing modellers to concentrate on coding the agents' functionalities.

Previous work has addressed AMB on the GPU, specifically work by D'Souza [2], which demonstrates an implementation of a number of specific Agent Based (AB) systems using discrete space partitioning. The performance of this system is impressive however the reliability of the priority scheme (for agent collisions in discrete space) and randomised birth allocation algorithms are questionable and only converge towards succession. Reusability is addressed by Richmond [7], who demonstrates a simple framework for swarm based modelling on the GPU. The work does not however address multiple agent types, environment interactions and birth and death allocations, all of which are essential for more generalized ABM beyond that of simple swarms. The work presented here provides the first flexible ABM framework for the GPU that incorporates each of these essential functionalities.

2. AN AGENT BASED FRAMEWORK ON THE GPU
Formal AB specification is important within ABM as it allows a simple, intuitive and portable way of defining agents and their associated behaviour. Our work builds upon the FLexible Agent Modelling Environment (FLAME) [1] utilising the X-Machine [4] as a formal modelling concept. More specifically our code uses FLAMEs, X-Machine Mark-up Language (XMML) to describe agents and communicating messages using XML and

'C' code to describe agent behaviour. Within our work the template based system within FLAME has been modified as well as the format of agent function code. As each GPU thread represents a single agent, the key change to function specification is the parameterisation of agent functions. This therefore avoids the use of any global variables between parallel agent threads. The following paragraphs describe key aspects to our GPU FLAME implementation.

GPU memory access is hidden by using a (GPU) device wrapper function for each agent function occurrence. This wrapper function reads agent data stored within a Structure of Arrays (SoAs) [5] to vastly improve coalesced memory access over the more intuitive Array of Structure (AoS) format. The agent function is then called by the wrapper, which passes an individual agent (represented by a C structure) as a parameter to the agent function. Finally the wrapper function writes any changes to the agent's internal variables back to global memory.

Birth and death processing require additional data storage. In the case of agent deaths, an additional integer field per agent is required to act as a flag signalling the agent's status (0 alive, 1 dead). A (linear time step) inclusive parallel prefix sum [3] is then used to calculate a new position for each (non dead) agent in the agent list. Following this a scatter kernel writes the agents into a new buffered agent list which ensures the agent list remains compacted. In the case of agents births a further buffered agent list is required. Each agent thread is then able to write a single agent output to this list in a linear pattern. As with agent death processing, a prefix sum is used to calculate a compacted index in the agent list for the new agents, which are appended using an additional scatter kernel to the original agent list.

The key to efficient $O(n^2)$ agent message communication is optimal use of on chip Shared Memory (SM) [6]. This allows messages to be paged into SM and serially processed by each thread on the processor, with near register access speed. Message access in agent functions is simplified by the use of a get function which returns either the first or next message in the agent list (this returns null at the end of the message loop). The message functions therefore handle iterating blocks of messages in shared memory, as well as paging new blocks of messages into SM after each block has been read by each agent thread.

3. EXPERIMENTAL RESULTS

In order to test the performance of our work we have compared the performance of FLAME GPU and the original FLAME framework have been compared. Agent function code in both cases is identical (excluding parameterisation). The first model is a benchmarking test which consists of a single (circle) agent, a location message and three agent functions. The second model is an implementation of a predator prey simulation consisting of two agent types, three message types and six agent functions.

Figure 1 illustrates the relative speedup of each model where the percentage increase indicates the percentage of speed improvement over the standard FLAME implementation. All results have been obtained on a single PC with an AMD Athlon 2.51 GHz Dual Core Processor with 3GB of RAM and a GeForce 9800 GX2 (utilising only a single PCB device). Whilst there is some initial fluctuation and performance penalty for lower population sizes the overall speedup of the circles model converges towards roughly 90x speedup and the Predator Prey model converges at over 250x speedup. The significant performance difference between the two models is a result of the increased complexity of the Predator Prey simulation. The higher arithmetic intensity in this case hides the memory access latency.

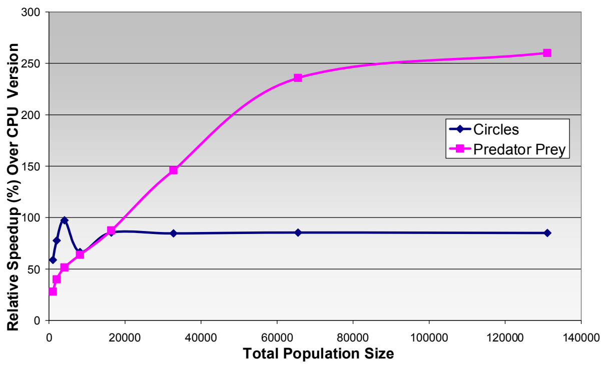

Figure 1 – Relative Speedup of GPU Performance

4. CONCLUSIONS

In this paper we have presented a GPU framework for ABM, which utilises existing agent specification techniques to produce efficient CUDA code. Unlike previous GPU alternatives [2, 7], important agent birth and death functionalities have been implemented and are guaranteed to succeed. Agent communication through messages has been implemented with efficient use of shared memory. A significant speedup over the FLAME frameworks original CPU implementation has been demonstrated. In future this work will be extended by providing more spatially aware communication algorithms and by benchmarking more complex AB models.

5. REFERENCES

[1] Coakley, S., Smallwood, R., and Holcombe, M. 2006. Using {X}-Machines as a Formal Basis for Describing Agents in Agent-Based Modelling, Proceedings of the 2006 Spring Simulation Multiconference, April 2006, pages 33-40.

[2] D'Souza, R. M., Lysenko, M., and Rahmani, K. 2007. SugarScape on steroids: simulating over a million agents at interactive rates. Proceedings of Agent2007 conference. Chicago, IL

[3] Harris, M., Sengupta, S., and Owens, J. 2007. Parallel Prefix Sum (Scan) with CUDA. GPU Gems 3. Chapter 39.

[4] Holcombe, M. (1988). X-Machines as a Basis for Dynamic System Specification. *Software Engineering Journal, 3(2)*, 69-76

[5] Howes, L. 2007. Loading Structured Data Efficiently With CUDA. NVIDIA Technical Report.

[6] Nyland, L., Harris, M., and Prins, Jan. 2007. Fast N-Body Simulation with CUDA, GPU Gems 3, Addison Wesley Professional, Chapter 31

[7] Richmond, P., Romano, D. M. 2008. Agent Based GPU, A Real-time 3D Simulation and Interactive Visualisation Framework for Massive Agent Based Modelling on the GPU. Proceedings International Workshop on Supervisualisation 2008. Kos Island, Greece.. In Press.