Performance of a Parallel Multi-Agent Simulation using Graphics Hardware

Timothy W. C. Johnson, Department of Computer Science and Computer Engineering, La Trobe University, Bundoora, Australia

John R. Rankin, Department of Computer Science and Computer Engineering, La Trobe University, Bundoora, Australia

ABSTRACT

Large-scale Agent-Based Modelling and Simulation (ABMS) is a field of research that is becoming increasingly popular as researchers work to construct simulations at a higher level of complexity and realism than previously done. These systems can not only be difficult and time consuming to implement, but can also be constrained in their scope due to issues arising from a shortage of available processing power. This work simultaneously presents solutions to these two problems by demonstrating a model for ABMS that allows a developer to design their own simulation, which is then automatically converted into code capable of running on a mainstream Graphical Processing Unit (GPU). By harnessing the extra processing power afforded by the GPU this paper creates simulations that are capable of running in real-time with more autonomous agents than allowed by systems using traditional x86 processors.

Keywords: GPU, Multi-Agent Systems, Parallel Programming, Simulation, System Performance

1. INTRODUCTION

Agent-Based Modelling and Simulation (ABMS) is a field of research which has garnered increasing attention and interest over the last few years (Macal and North, 2007). Conventional modelling techniques are no longer as applicable as they once were as researchers seek to model intricate systems with complex interdependencies. Simulations performed with ABMS can result in the emergence of interesting macroscopic behaviours due to the thousands of minor interactions that take place between agents, something which may not occur using more traditional forms of simulation (Chan et al., 2010). Areas such as air-traffic analysis, anthropology, biomedical research, chemistry, crime analysis, disaster response management, ecology, energy analysis, market analysis, social network analysis, military simulations, urban simulations, pollution management, and refugee-camp analysis all stand to benefit

DOI: 10.4018/ijats.2014100104

Copyright © 2014, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
from advances made in ABMS. Unfortunately, implementing these systems from scratch tends to be expensive both in terms of the development time required and the monetary costs associated (Moradi et al., 2008).

Agents are software entities that act in an independent, autonomous manner (Wooldridge and Wooldridge, 2001). They are goal-oriented, striving to achieve their stated objectives in a proactive manner. They are also reactive, capable of detecting and responding to unexpected changes in the environment in which they are situated. Finally, agents have social abilities. They can interact with humans and other agents, negotiating, bargaining, cooperating, competing or coordinating as they best see fit (Johnson and Rankin, 2012). Sometimes agents possess the ability to learn and adapt, however this is not a requirement and can even prove detrimental (Winikoff and Padgham, 2004). At a technical level the inner workings of an agent can be viewed as a four step loop which consists of: observation, viewing the world and updating any stored knowledge, deliberation, deciding what to do next, planning, creating a plan to achieve the chosen goal, and execution, carrying out the constructed plan.

To perform large-scale ABMS in real-time we require high quantities of processing power. The rapid growth of the gaming industry has generated high demand for cheap and powerful processing hardware, resulting in the Graphical Processing Unit (GPU) as we know it today. These GPUs are highly parallel, with hundreds of processors executing together in a SIMD (single instruction, multiple data) configuration. Improving the performance of these units is as simple as adding more cores which has allowed the GPU to drastically outstrip traditional Central Processing Units (CPUs) in terms of raw performance (Luebke, 2008). The early 2000s saw the emergence of High Level Shader Languages (HLSL) (Peercy et al., 2000; Proudfoot et al., 2001) which would eventually evolve into GPGPU (General-Purpose computing on a GPU) languages such as OpenCL and CUDA. GPGPU allows us to use graphics hardware for things other than image rendering. It has already been applied to areas such as cosmological data analysis (Roeh et al., 2009), simulating cloud dynamics (Harris et al., 2003) and improving database performance (Zidan et al., 2011). However, GPGPU has its own set of challenges which must be overcome before we can begin to apply the processing power of the GPU to ABMS.

Over the course of this paper we’ll first review the background of this topic, including a discussion of previous work. Next we’ll cover our specific implementation, including the presentation of a model for ABMS, and solutions we have produced to overcome the various problems faced by GPGPU developers. Following this we’ll test the performance of the system, making comparisons between our GPU implementation of an agent simulation and a CPU version developed in earlier work. Finally, we’ll discuss these results and try to draw some conclusions with a look towards the future.

2. BACKGROUND

In this section we’ll begin by presenting the challenges faced when developing applications for GPGPU and conclude by making note of some of the previous attempts made to unite the fields of GPGPU and agents.

2.1. Challenges of GPGPU

2.1.1. Branch Divergence

Branch divergence is arguably the biggest issue faced by GPGPU developers. When a group of threads running simultaneously in a SIMD configuration encounter a branch in the code, if even just one of those threads evaluates the condition differently from the other threads, all threads are forced to compute both sides of the code branch, throwing away results that are not relevant. While small code branches cause a negligible impact on performance, many large code branches can severely hinder the performance of the system.
Hierarchical Design Method for Multi-Agent Systems
[www.igi-global.com/article/hierarchical-design-method-for-multi-agent-systems/141874?camid=4v1a](www.igi-global.com/article/hierarchical-design-method-for-multi-agent-systems/141874?camid=4v1a)