Simulation Modelling Practice and Theory 57 (2015) 118–132

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Multi-agent simulation on multiple GPUs

N.M. Ho [a], N. Thoai [a], W.F. Wong [b,*]

[a] Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Viet Nam
[b] School of Computing, National University of Singapore, Singapore

ARTICLE INFO

Article history:
Received 17 March 2014
Received in revised form 26 June 2015
Accepted 29 June 2015
Available online 17 July 2015

Keywords:
Multi-agent simulation
MASON
Simulation framework
CUDA
Multi-GPU computing

ABSTRACT

Multi-agent simulation is widely used in many areas including biology, economic, political, and environmental science to study complex systems. Unfortunately, it is computationally expensive. In this paper, we shall explore the implementation of a general multi-agent simulation system on a system with multiple GPUs acting as accelerators. In particular, we have ported the popular Java multi-agent simulation framework MASON to a nVidia CUDA-based multi-GPU setting. We evaluated our implementation over different numbers and types of nVidia GPUs. For our evaluation, we ported three models in the original version of MASON. On the well-known Boids model, we achieved a speedup of 187×. Using a fictional model, we showed that speedup of up to 468× is possible. In the paper, we shall also describe the detailed internals of our system, and the various issues we encountered and how they were solved.

1. Introduction

Many phenomena and systems in nature involve a large number of interacting individual agents. Multi-agent simulation [1] is a methodology for the study of such complex systems. In recent decades, with the availability of more compute power, many simulation frameworks have been developed for various fields of science and engineering. Examples of these include Swarm [2], MadKit [3] NetLogo [4], Repast [5], SPADES [6], and FLAME [7]. More recent examples include TurtleKit [8] that is based on the combination of the Madkit kernel and the Logo simulation model, GAMA [9] that integrated GIS data into the simulation, and Janus [10] that focused on holonic multi-agent applications. MASON [11] was also introduced as a light-weight simulation framework written in Java that supports flexible scheduling as well as facilities that are easy to use. It has been well optimized for speed especially when compared to other frameworks [12]. However, even with the advances in CPU technology, multi-agent simulation is still extremely slow for realistically large scale models that have a large number of agents with complicated interactions between them.

General purpose processing on Graphics Processing Units (GPU) has emerged as an important technology for accelerating applications with massive amount of parallelism, including multi-agent simulation. Simulation for many fields of science and engineering such as traffic [15], soil [16] and blood coagulation system [17] have benefited from its use. In addition to these specific applications, there were also efforts to implement general cellular-based models on GPUs [18]. However, such simulation frameworks were implemented on single GPUs [19,20]. These efforts pre-dated modern GPU programming paradigm such as CUDA [21]. As a result, most of the works focused on the difficult issue of programming using textures. In addressing this issue, FLAME GPU [22], for example, provided a framework for describing models in the XML format with an

* Corresponding author. Tel.: +65 6516 6902; fax: +65 6779 4580.
E-mail address: wongwf@nus.edu.sg (W.F. Wong).

http://dx.doi.org/10.1016/j.simpat.2015.06.008
1569-190X/

abstraction for the user to program the GPU. Writing agent function scripts on FLAME GPU still required programming skills as the abstract description has a C-based syntax with extensions. FLAME's main contribution is in providing a formal method to describe the simulation models. It also has a simple and extensible visualization facility. Besides FLAME, a single GPU solution was also implemented in TurtleKit [23] that was demonstrated on diffusion and perception modules in a grid-based environment. It showed the possibility of not using an all-in-GPU approach when achieving an acceptable speedup. Recently, a new feature in the JADE [24] simulation framework introduced the ability to execute CUDA agents [25]. However, a CUDA agent is still simply a computational service for other Java agents running on the CPU. It is an on-demand service which receives the programming code from other agents. So far, the reported experiments have been about comparing matrix multiplication between CPU and GPU.

The earlier works on single GPU multi-agent simulation systems showed great potentials. There is a good overall match between multi-agent simulation and the hardware platform. However, the previous works also highlighted issues and the limitations of single GPU, often stated as 'future works'. Some of the issues raised are:

- The previous frameworks on GPU support limited kinds of environments and agents, mostly grid-based environment and boids-like agents.
- When the memory or the cores of a single GPU are fully utilized, introducing more agents will result in the significant degradation of performance. The only way out is to scale up by migrating to multi-GPU environment.
- Multiple GPUs have a non-uniform memory access (NUMA) memory model that requires explicit memory transfer. It necessitates additional synchronization processes and changes in the framework, including algorithms.
- To efficiently manage agents and their locations on multiple GPUs, that operate in a non-shared memory environment, there are no trivial methods or automatic tools available. Therefore, single GPU frameworks do not trivially scale to multiple GPUs.

The work in [26] also discussed the scaling issue with a note stating their intention to solve it in the future with OpenCL. Nevertheless, [27] showed that OpenCL performs worse than CUDA. The work in [28] that describes latency hiding schemes for cellular-based models is one of the few existing related work for solving the scaling issue. These together with the need for better visualization and GUI tools for real-time analysis motivated the work in this paper. By using the existing facilities of MASON to describe and simulate models, adding the backend ability to accelerate the core simulation using on multiple GPUs, we have effectively scaled the framework to tackle larger problems. In particular, the contributions of this paper are:

- We proposed a multi-GPU framework that has the capability of handling data over multiple devices in the simulation, and also offers the ease of programming in Java. The ideas we proposed can also be used for other implementations, including OpenCL ones.
- We modified some of the environment facilities of MASON framework to support both single as well as multiple GPUs.
- We introduced some key techniques for handling simulation data, especially how it can be efficiently done on multiple GPUs.
- By leveraging the features of the MASON framework, our solution enables the use of environments beyond restricted grids.
- We performed a performance study on our optimized MASON. It shows the potential for speeding up simulations by as much as two orders of magnitude, depending on the models and hardware configuration.

As far as we know, this paper is among the first to examine the issue of scaling multi-agent simulation using multiple GPUs.

Intel's new Many Integrated Core architecture (MIC) has been considered as one of the solutions. Multi-MIC versus multi-GPU systems have been benchmarked with two different problems in [13,14]. Unfortunately, the result shows that multi-GPU system achieves better efficiency than multi-MIC even with the same programming effort.

The rest of this paper is organized as follow: Section 2 describes the design and related issues of a hybrid version of MASON and CUDA. Section 3 presents and explains the implementation of multi-agent simulation on multiple GPUs. After that, in Section 4, we report the result of our experimental evaluation. Finally, the conclusion will be stated in Section 5.

2. System overview

2.1. System design

Our proposed system combines MASON and JCuda [29] yielding a general multi-agent simulation environment that can exploit acceleration using multiple GPUs. Specifically, our system performs the simulation task on many GPUs, before using the visualization capability of MASON to present the results.

The system, as shown in Fig. 1, consists of three components: the visualization, the simulation model and the JCuda components. The visualization component has a console, a display and other facilities to manipulate the simulation models. It also has the responsibility for holding the portrayals of fields and inspectors. Other tools such as the charting and media