European Journal of Operational Research 000 (2017) 1–11

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Interfaces with Other Disciplines

Accelerating Petri-Net simulations using NVIDIA Graphics Processing Units

Panayioti C. Yianni[a], Luis C. Neves[b], Dovile Rama[b], John D. Andrews[b,*]

[a] Amey Strategic Consulting and Technology, Furnival St., London EC4A 1AB, United Kingdom
[b] Resilience Engineering Research Group, University of Nottingham, Faculty of Engineering, University Park, Nottingham NG7 2RD, United Kingdom

ARTICLE INFO

Article history:
Received 5 July 2016
Accepted 30 June 2017
Available online xxx

Keywords:
CUDA
GPU
Petri-Net
Parallel
Asset management

ABSTRACT

Stochastic Petri-Nets (PNs) are combined with General-Purpose Graphics Processing Unit (GPGPUs) to develop a fast and low cost framework for PN modelling. GPGPUs are composed of many smaller, parallel compute units which has made them ideally suited to highly parallelised computing tasks. Monte Carlo (MC) simulation is used to evaluate the probabilistic performance of the system. The high computational cost of this approach is mitigated through parallelisation. The efficiency of different approaches to parallelisation of the problem is evaluated. The developed framework is then used on a PN model example which supports decision-making in the field of infrastructure asset management. The model incorporates deterioration, inspection and maintenance into a complete decision-support tool. The results obtained show that this method allows the combination of complex PN modelling with rapid computation in a desktop computer.

1. Introduction

Petri-Nets (PNs) have been gaining in popularity in the field of Operations Research, specifically for work-flow management systems (Salimifard & Wright, 2001), supply chains (Viswanadham & Raghavan, 2000) and enterprise resource planning (Aloini, Dulmin, & Mininno, 2012). One of the limitations of PNs in conjunction with probabilistic transitions is the requirement for Monte Carlo (MC) simulations resulting in high computational cost. Studies have been undertaken into accelerating Petri-Nets (PNs) with GPGPUs, however these have mostly been in the fields of Biology and Physics.

The use of MC methods is very popular in science, engineering and economics research. Over time MC-based models have gotten progressively more sophisticated which often means longer computation times, limiting their applicability. The use of GPUs in academia as GPGPUs is gaining in popularity as it has the potential to dramatically decrease computation time allowing for much faster throughput. At the same time, the technological development of GPUs and their progression into general computing has meant that the cost per computing core has dropped significantly.

This study investigates the suitability of GPGPU acceleration with PNs for a decision support tool, in this case using an example from asset management. Asset management models often combine engineering, management and economics to be able to make informed decisions. A simple PN example is used as a proof of concept and then the same approach is applied to an established railway bridge model. This model was selected as it contains a number of complex PN features and therefore would be appropriate to test the suitability of GPGPU acceleration. The bridge model used as the example in this study provides predictions of condition over time, including the effects of maintenance actions and the associated costs.

Although the study uses a railway bridge asset management example, many of the topics discussed are appropriate for GPGPU acceleration of general decision support PN models. As decision support tools become more engrained into business practices and as the adoption of Petri-Nets (PNs) as a modelling approach becomes more common, their efficient computation will become more important.

2. Simulation acceleration using GPGPUs

A number of studies have been undertaken in the field of Physics to accelerate MC-based models with GPGPUs. Although not directly related to PNs, the application of the GPGPU to the modelling approach is relevant.

* Corresponding author.
E-mail address: John.Andrews@Nottingham.ac.uk (J.D. Andrews).

http://dx.doi.org/10.1016/j.ejor.2017.06.068
0377-2217/

Tomov, McGuigan, Bennett, Smith, and Spiletic (2005) performed a study to investigate the suitability of GPGPUs with 2D and 3D Ising models. An Ising model represents the spin of magnetism in a lattice structure. Each of the magnets can be in one of two states. The model becomes more complex when considered in a 3D space. The authors state that the common way of incorporating MC methods into the Ising model is to choose a random path through the magnets, at each magnet generating a random number which decides whether the spin should be reversed or not. At this time the Compute Unified Device Architecture (CUDA) framework (the standard framework for GPGPU development currently) was yet to be established and so an older framework known as Cg was used. The GPU used was an NVIDIA NV30, a high-end GPU from 2003. The authors conclude that the GPGPU implementation of the Ising model is three times faster than the Central Processing Unit (CPU) implementation on average. This holds true for both the 2D and 3D examples tested. The authors mention that the GPGPU performance may have been limited due to branching as the contemporary GPU used for this study did not support this feature.

Preis, Virnau, Paul, and Schneider (2009) also perform a study using GPGPUs on Ising models, testing both 2D and 3D models. They use an NVIDIA GeForce GTX 280, a high-end GPU from 2008. The CUDA framework had been released by this time. They use a checker-board pattern approach to make the computation more efficient. They achieve a 60 times speed-up factor in the 2D Ising model test and a 35 times speed-up factor in the 3D scenario.

Yang, Wang, and Chen (2007) used the power of GPGPUs to accelerate molecular dynamics. The study involved simulations to calculate thermal properties. The process of simulating molecules involves a 3D grid of interacting nodes. Simulations are carried out where the amount of energy passed from atom to atom is recorded. The stochastic nature of the modelling is due to the movement of molecules based on a decision which uses random numbers. They used an NVIDIA GeForce 7800 GTX, a high-end card from 2005. They used the Cg programming framework with 600,000 iterations per simulation. The number of atoms in the simulation was tested against the execution time. The results show that the execution time almost follows an exponential curve with the number of atoms in the simulation. In every test, the GPGPU execution is quicker than that of the CPU. The authors conclude that their average speed-up factor is between 10 and 11 times using the GPGPU.

van Meel, Arnold, Frenkel, Zwart, and Belleman (2008) also use GPGPUs with molecular dynamics. They implement the N-squared molecular dynamics algorithm for their simulations. In contrast to Yang et al. (2007), the CUDA framework had been released by this point. They perform an interesting comparison between the simulations performed on (1) a CPU, (2) GPGPU with Cg and (3) GPGPU with CUDA. As well as the update in programming framework, they use a newer GPU from Yang et al. (2007): an NVIDIA GeForce 8800 GTX. Their results show that the GPGPU with Cg performs around 40 times faster than the Central Processing Unit (CPU) implementation. Using the newer CUDA framework, they manage an 80 times speed-up factor over the CPU execution.

Geist, Hicks, Smotherman, and Westall (2005) were one of the first studies to try to accelerate PNs with GPGPUs. They use an NVIDIA GeForce 6800 Ultra, a high-end GPU from 2004. The authors create a PN simulator using Cg called “Cgpetri”. They compare this to serial PN simulators “xpetri” (Geist, Crane, Daniel, & Suggs, 1994) and “SPNP” (Hirel, Tuffin, & Trivedi, 2000). The example of the dining philosophers was used in the study, created by E. W. Dijkstra, but formalised by Hoare (1978). This is a common example of a PN with conflict conditions. These arise because each philosopher requires two forks to eat with, however there are too few forks to satisfy each diner. Therefore, the resource must be shared which means that the diners alternate between two condition states. They test the example ranging from 25 diners to 400 diners, recording the execution time. When there are few diners, execution on the Central Processing Unit (CPU) is faster as the data does not need to be packaged, passed to the GPGPU and then transferred back after computation. However, the critical point in this example is at 50 philosophers; beyond which the GPGPU based simulator outperforms the serial simulators. For example, when the number of philosophers reaches 75, the execution time of the serial simulator SPNP is approximately 38 seconds. This is almost twice as long as that of the GPGPU based simulator, which is approximately equal to 18 seconds. The serial simulators follow an almost exponential execution time with the number of diners, however when run with Cgpetri the execution time is approximately linear as a function of the number of diners.

Geist et al. (2005) also apply this approach to a Lattice–Boltzmann example. This is commonly used for fluid dynamics where a series of partial differential equations are to be solved. The more points that are created in the simulation, the more detailed the resulting outputs. There has been studies performed to parallelise Lattice–Boltzmann models and a common approach is to divide the problem into “subcubes” for concurrent computation. The authors propose a PN to schedule the computation of these subcubes to increase efficiency. Again they compare Cgpetri to xpetri and SPNP. They increase the number of places in the flow net, recording the execution time. Again, it can be seen that when the number of places in the flow net is small, the overhead of using the GPGPU means that the execution is slower. However, the advantage of the GPGPU quickly shows its strength as the execution time is almost the same across the whole range of places tested whilst the serial simulators experience significant slow down. The authors close with a comment that Moores Law, a characteristic applied to Central Processing Unit (CPU) development, means that their performance seems to double every 18 months (Schaller, 1997). However, the rate of development of GPUs is closer to doubling every 6 months. Therefore, not only was the GPGPU already accelerating the simulations but the difference was set to increase even more.

Chalkidis, Nagasaki, and Miyano (2011) also attempt to accelerate PNs with GPGPUs. They use a real world example from the field of biology, specifically microbiology, and medicine. They claim that using a state-of-the art modelling approach, PNs in this instance, with GPGPU acceleration is the next step in producing fast, accurate, modelling results. They use the CUDA programming framework with an NVIDIA GeForce 8800, a mid-range GPU from 2006. They explain that they use Hybrid Functional PNs and detail its decomposition for GPGPU programming. They mention that the possible acceleration could be significant if they could parallelise their PN. They do so by splitting their PN into three processes hierarchically. Processes one and two are able to be simulated concurrently as they do not interact directly. Then process three can use the results of processes one and two to finish the simulation. Theoretically, by using this approach the PN simulation could experience a 1.5 times speed-up factor in serial operation. However, when considering the massively parallel characteristic of GPUs, the speed-up could be significantly higher. They conclude by saying that their average PN acceleration when using the GPGPU was 18 times faster than the serial execution.

In summary, a number of studies have been undertaken accelerating simulations in the fields of Physics and Mathematics. Table 1 shows the results of the average speed-up factor in each of these studies. It can be seen that earlier implementations using Cg did not provide as much speed-up as later versions using CUDA. This occurs because: (1) the CUDA framework provided updated algorithms, processes and techniques which would help efficiency and (2) although the studies were carried out with contemporary CPUs and GPUs, the advancement of GPU compute capability outstrips

Please cite this article as: P.C. Yianni et al., Accelerating Petri-Net simulations using NVIDIA Graphics Processing Units, European Journal of Operational Research (2017), http://dx.doi.org/10.1016/j.ejor.2017.06.068