GPU-accelerated Steady-State Analysis of Probabilistic Boolean Networks

Andrzej Mizera[1], Jun Pang[1,2], and Qixia Yuan[1⋆]

[1] Faculty of Science, Technology and Communication
University of Luxembourg, Luxembourg
[2] Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg, Luxembourg
firstname.lastname@uni.lu

Problem statement. Steady-state computation is important for analysing biological systems modelled as probabilistic Boolean networks (PBNs). Since the state-space is exponential in the number of nodes, the use of statistical methods and Monte Carlo methods remain the only feasible approach to address the problem for large PBNs (e.g., with more than 50 nodes) [5, 2]. Such methods usually rely on long simulations of a PBN; hence the simulation speed becomes critical. For large PBNs with high precision requirements, a slow simulation speed becomes an obstacle of computing the steady-state probabilities. Intuitively, parallelising the simulation process can be an ideal way to accelerate the computation process.

Our approach. We propose to parallel the simulation of PBNs using multiple graphics processing unit (GPU) cores. A GPU usually contains hundreds or thousands of cores. It uses data parallelism, i.e., the same instruction is run in different cores with different data. The memories provided by GPU can be divided into two types based on the access speed: *fast-memory* and *slow-memory*. Accessing fast-memory is highly efficient, but the size of fast-memory is very limited. A GPU program is executed in parallel by the GPU threads. Usually thousands of threads are launched in parallel to hide latency. Due to the specific architecture of GPUs, parallelising a process in a GPU has to be treated carefully. A discussion of two particular issues follows.

Firstly, synchronisation between cores is very time expensive in a GPU. To avoid it, we let each GPU core handle all the nodes in a PBN. Instead of simulating one trajectory, we simulate multiple trajectories in parallel. Samples from multiple trajectories can be combined to compute steady-state probabilities using a combination of the two-state Markov chain approach with the Gelman & Rubin method [1, 3].

Secondly, the performance of a GPU is highly related to how well the latency is hidden. Latency can be hidden via the use of more threads, more blocks, and/or fast-memory. More threads/blocks require more fast-memory, but the size of fast-memory is very limited. Therefore, a trade-off between the number of threads/blocks and the use of fast-memory is required. We first optimize our

⋆ Supported by the National Research Fund, Luxembourg (grant 7814267).

node #	CPU time (s)	GPU time (s)	speedup
100	635.27	1.84	345×
200	424.18	1.84	231×
500	1567.77	5.80	270×
91	905.10	3.54	256×

Table 1. GPU speedup for computing four steady-state probabilities.

data structure to minimize the use of memory and then follow the rule that the frequently accessed information should be put in fast-memory whenever possible since the latency caused by accessing slow-memory is relatively large. To better understand the optimization, we briefly review what a PBN is. A PBN is composed of a set of binary-valued nodes, each of which has a certain number of Boolean functions. The process of simulating a PBN consists of selecting a Boolean function for each node and updating its value in accordance the selected function (see [4, 6] for details). The state (value of all nodes) and the Boolean functions (stored as a truth table) are repeatedly used in the simulation process and require much memory to store, hence we optimize the data structure to represent the state of a PBN and the truth table. We use the primitive integer type (32 bits) instead of Boolean arrays to store a state of a PBN. The integer type is used due to the following two reasons: 1) it reduces the memory usage by 4 times comparing to Boolean arrays; 2) operations on 32 bits data are faster than or equal to those on non-32 bits data in our GPU architecture. The truth table is optimized similarly as the state, i.e., it is also stored using integers. After optimization, we store the state in *registers* (fast-memory), if possible. In the cases that a PBN is extremely large and registers are not enough, the slow global memory is used. However, accessing this slow global memory is reduced by 32 times using an intermediate register. The truth table as well as other frequently accessed arrays (e.g., the selection probabilities) are stored in *shared memory* (fast-memory). Frequently accessed single variables are stored in registers. After arranging all variables in memory, we compute the optimal number of threads and blocks to be launched based on the usage of fast-memory to hide latency as much as possible.

Preliminary results. We have evaluated the proposed GPU-based simulation of PBNs for computing steady-state probabilities of both randomly generated networks and of a real biological network using the approach in [3]. On randomly generated networks, our proposed GPU-based parallelised approach showed more than two orders of magnitude speedups compared to the sequential CPU version. The evaluation on a real biological network was performed by analysing an apoptosis network with 91 nodes [2]. The speedups for computing steady-state probabilities for 3 randomly generated networks and the real 91-node network are shown in Table 1. All experiments were conducted on a high-performance computing (HPC) node, which contained Intel Xeon E5-2680 v3 @2.5 GHz and a NVIDIA Tesla K80 Graphic Card with 2496 cores @824MHz.

References

1. Gelman, A., Rubin, D.: Inference from iterative simulation using multiple sequences. Statistical Science 7(4), 457–472 (1992)
2. Mizera, A., Pang, J., Yuan, Q.: Reviving the two-state markov chain approach (technical report) (2015), available online at `http://arxiv.org/abs/1501.01779`
3. Mizera, A., Pang, J., Yuan, Q.: Parallel approximate steady-state analysis of large probabilistic Boolean networks. In: Proc. 31st ACM Symposium on Applied Computing. pp. 1–8 (2016)
4. Shmulevich, I., Dougherty, E., Zhang, W.: From Boolean to probabilistic Boolean networks as models of genetic regulatory networks. Proceedings of the IEEE 90(11), 1778–1792 (2002)
5. Trairatphisan, P., Mizera, A., Pang, J., Tantar, A.A., Sauter, T.: optPBN: An optimisation toolbox for probabilistic boolean networks. PLOS ONE 9(7) (2014)
6. Trairatphisan, P., Mizera, A., Pang, J., Tantar, A.A., Schneider, J., Sauter, T.: Recent development and biomedical applications of probabilistic Boolean networks. Cell Communication and Signaling 11, 46 (2013)