GPU-accelerated and CPU SIMD Optimized Monte Carlo Simulations of ϕ^4 Model

P. Białas, J. Kowal, A. Strzelecki
Faculty of Physics, Astronomy and Applied Computer Science,
Jagiellonian University in Krakow

Abstract. We present a high-performance implementation of Monte-Carlo simulations of 2 and 3 dimensional φ^4 model both on GPU and on CPU using *OpenMP* and *SSE/AVX* SIMD vectorization for *Intel Core i7*. This simulation is a part of on ongoing educational project but we took this opportunity to compare the two programming models and check the claims of over 100× advantage of GPU over generic CPU made recently in the literature.

The implemented model differs from the ones presented in the literature (*e.g.* [4]) by the extended neighborhood that has to be taken into account when updating a single lattice field. While model is inherently parallelizable, grid points that lie in the same neighborhood cannot be updated together. That proved to be a problem especially on the GPU where much more threads access the grid in parallel. Taking into account a larger neighborhood means that a simple checkerboard decomposition pattern cannot be used and we have devised a new two-level grid decomposition scheme. Greatest challenge for GPU implementation was to provide efficient memory synchronization satisfying those constraint. Shared memory was used extensively to minimize update latencies between single threads. The CPU implementation have run into similar problems although less severe because of smaller number of parallel threads. We have however opted to use the same partitioning scheme as on the GPU.

To ensure valid simulation results we had to provide efficient random number generator with good statistical quality and long period. We have chosen *Tausworthe* random generator[2] as the best one providing uniform distribution for the simulation. Our GPU *CUDA* implementation is based on *GPU Gems 3*[1], while our own CPU implementation uses *SSE2* SIMD integer extensions.

Altogether we managed to achieve 0.13 nanoseconds for single lattice field update on *NVIDIA GTX 470*, reaching around 430 Gflops of 1088 Gflops peak performance of this device.

While SSE gives theoretical 4× benefit for single precision floating point operations, AVX providing 8×, not all scalar *x86* instructions have vector counterparts. In particular direct *XMM* registers gather and scatter and vectorized integer operation for full length AVX 256-bit registers are missing, which makes impossible to port random number generator from 128-bit SSE to 256-bit AVX. Initially planned for AVX standard, these were postponed to AVX2 planned for 2013. As soon AVX2 capable CPU devices appear on the market we plan to revise our evaluation.

Our CPU *OpenMP* and *SSE/AVX* implementation compiled via GCC 4.4 or higher and running on *Intel Core i5 2.5 Ghz* quad core CPU presented 15× performance boost comparing to single threaded scalar code and 3.76 nanoseconds for single lattice field update. Which gives the 15 Gflops. There is no significant increase in performance while switching from SSE to AVX instructions.

This gives around 28× advantage to GPU, which is noticeably less than promised by many publications, however much higher that comes from comparison of 160 Gflops peak performance of tested i5 CPU to *GTX 470*[3]. This can be traced back to 128-bit only *Tausworthe* random number generator implementation and inefficient store and load operations (gather/scatter) which can be partially mitigated by the reorganizing the way in which the grid is stored in the memory. This is the subject of an ongoing work.

References

[1] Lee Howes and David Thomas. Efficient random number generation and application using CUDA. In Hubert Nguyen, editor, *GPU Gems 3*, chapter 37. Addison Wesley, August 2007.

[2] Pierre L'Ecuyer. Maximally equidistributed combined tausworthe generators. *Math. Comput.*, 65(213):203–213, 1996.

[3] Victor W. Lee, Changkyu Kim, Jatin Chhugani, Michael Deisher, Daehyun Kim, Anthony D. Nguyen, Nadathur Satish, Mikhail Smelyanskiy, Srinivas Chennupaty, Per Hammarlund, Ronak Singhal, and Pradeep Dubey. Debunking the 100x gpu vs. cpu myth: an evaluation of throughput computing on cpu and gpu. *SIGARCH Comput. Archit. News*, 38(3):451–460, June 2010.

[4] Tobias Preis, Peter Virnau, Wolfgang Paul, and Johannes J. Schneider. Gpu accelerated monte carlo simulation of the 2d and 3d ising model. *Journal of Computational Physics*, 228(12):4468 – 4477, 2009.