CMST 21(3) 117-121 (2015) DOI:10.12921/cmst.2015.21.03.002

Distributed Processing of the Lattice in Monte Carlo Simulations of the Ising Type Spin Model

Szymon Murawski*, Grzegorz Musiał , Grzegorz Pawłowski

Faculty of Physics, Adam Mickiewicz University
ul. Umultowska 85, 61-614 Poznań, Poland
**E-mail: szymon.murawski@gmail.com*

Received: 06 November 2014; revised: 22 May 2015; accepted: 22 May 2015; published online: 30 June 2015

Abstract: Parallelization of processing in Monte Carlo simulations of the Ising spin system with the lattice distributed in a stripe way is proposed. Message passing is applied and one-sided MPI communication with the MPI memory window is exploited. The 2D Ising spin lattice model is taken for testing purposes. The scalability of processing in our simulations is tested in real-life computing on high performance multicomputers and discussed on the basis of speedup and efficiency. The larger the lattice the better scalability is obtained.

Key words: parallel processing, distributed processing, one-sided MPI communication, Monte Carlo simulations, classical spin lattice model, high performance computing

I. INTRODUCTION

Many works concern parallelization of MC simulations with the aim to reduce processing time, to tackle the problem of larger and larger computational complexity. MC simulations can be conveniently parallelized because the simulated process is Markovian. Considering more complex systems one usually uses more and more parallel processes to obtain results in realistic time. The problem arises when the lattice representing the regarded system does not fit into the memory of a single computer.

We take the spin Ising model on a square lattice as an example of our Monte Carlo (MC) simulation object, as it is simple, reflects the main features of typical systems simulated by a Monte Carlo method and still finds many interesting applications.

In this work we address the following problem: how to organize our MC simulations with the Ising spins having a lattice distributed. We also consider the problem of complexity in computer memory space. It is obvious that from the point of view of computing time we could slow down processing when we keep the lattice distributed. Thus, we focus on analysis of the speedup and efficiency in distributed processing of the lattice in Monte Carlo simulations of the Ising type spin model.

There are numerous problems in which one has to regard greater and greater size of the lattice in MC simulations. Very often one has to extrapolate the MC simulations results to those of macroscopic ones, which usually means extrapolation to infinity (e.g. [1–4]). The larger the lattices considered in the simulations, the better the extrapolated result.

II. THE SIMULATED SYSTEM

The simulation of the Monte Carlo type is a kind of a computer experiment performed to predict the behavior of a macroscopic system (i.e. with a large number of degrees of freedom) when given the laws governing its microscopic behavior. The convenient starting point to the latter is the energy operator H (Hamiltonian), as it allows a description of the behavior of a system on a microscopic level using the quantum and statistical mechanics laws.

To test our idea of keeping the lattice of the simulated system distributed among computer nodes, we take an Ising model on a square lattice, as it is simple but not trivial, and

we verify the correctness of the code by comparing results to the exact ones [5]. The main features of the system come from the Hamiltonian with the following form:

$$-\frac{H}{k_B T} = \sum_{\langle ij \rangle} K s_i s_j, \tag{II.1}$$

where T is the thermodynamic temperature, k_B is the Boltzmann constant and s_i is the Ising degree of freedom (spin) residing on each lattice site; each s_i can freely take one of two values $+1$ or -1 depending on its interactions with neighboring ones; $\langle ij \rangle$ denotes the summation over pairs of nearest neighboring lattice sites i and j; $K = J/(k_B T)$; J is the coupling of the nearest neighbor interaction between the Ising degrees of freedom s_i (i.e. further interactions are neglected in this model).

III. SIMULATION DETAILS

Performing simulations we generate equilibrium microstates (i.e. possible configurations of all degrees of freedom in the system) of the finite-size square samples of the size $L \times L$ for fixed values of the model parameters using the Metropolis algorithm [6], i.e. trying to reverse single spins, thus forming a Marcov chain of possible states of the system. During the simulation the system travels through those states with probability

$$P \sim e^{-\frac{\Delta E}{kT}}, \tag{III.1}$$

where ΔE is the energy difference between the trial state x' and the old state x calculated using Eq. (II.1). Extensive details about this approach to the problem can be found in [7]. We have chosen this algorithm because of its simplicity and typical structure. There are other algorithms of similar structure which flip clusters of spins leading to a significant reduction (or even elimination) of the critical slowing down [8]. An interesting algorithm published recently [9] uses a random walk in the energy space to estimate the density of states of the studied system. Other approaches involve using the so-called Worm algorithm [10] or cluster simulations on an infinite lattice [11].

At the start of simulations we drive the system into the equilibrium state. For this purpose part of the initial Monte Carlo steps (MCS) are discarded so as to avoid correlations with the initial state. This process is called thermalization. As usual, a MCS is completed when each of the lattice sites has been visited once.

IV. COMMUNICATION WITHIN PARALLELIZED MC SIMULATIONS

When trying to perform simulations for larger system size L one faces two challenges: time needed to do all calculations rapidly increases (proportionally to L^2) as well as memory needed (also as L^2). Our way to overcome this problem is to divide the whole lattice of size $L \times L$ among p parallel processes using MPI [12], so that each process receives its part of the lattice of size $L \times L/p$ called a stripe. Each process then performs all MCS on its stripe of the lattice and in the end the results from all processes are accumulated. As the consequence of this distribution of the processed lattice the fundamental problem arises when the algorithm tries to construct a trial state x' using a boundary site. Calculation of the energy difference ΔE requires the knowledge about its neighboring sites and for sites located on the boundary of the stripe the process i has to use MPI communication to check the state of the neighboring site belonging to process $i-1$ or $i+1$.

A cooperative way of moving data around the parallel processes, collective or point-to-point communication, involves both the sender and the receiver in the operation. MPI forces point-to-point ordering of the messages between two peers [13] which decrease performance and limits expressiveness of the application as every send has to match a receive. Non-blocking communication does not change the essence of the problem.

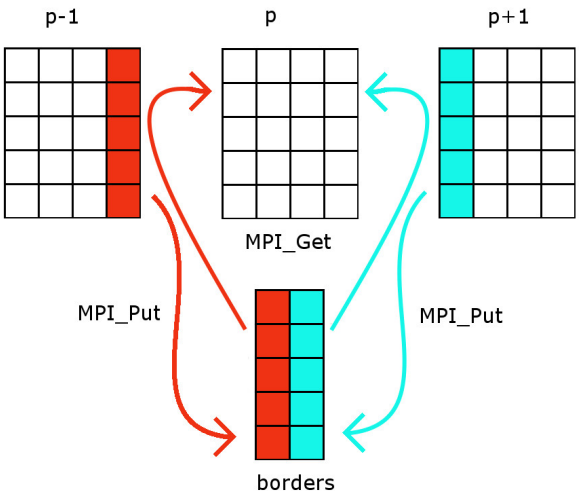

Fig. 1. Example of a piece of a lattice distributed among three processes. Process p keeps information about neighbouring states in separate table `borders` which is put in the MPI window for remote memory access. One-sided communication is used to update this table by neighbouring processes $p-1, p+1$ and to access the data by process p

To overcome these problems we have decided to test remote memory access and keep the boundaries of the local parts of the lattice (on which MC simulations are performed) within the MPI memory window [13]. Here no matching of parallel processes communication is required. Communication with the MPI memory window is one-sided and non-

blocking, and computation can overlap it. The MPI memory window is a contiguous memory region that is available to parallel processes from the same communicator as a target for their put and get operations specifying the base address and the length of the data in bytes. The use of `MPI_Win_lock` and `MPI_Win_unlock` for the `MPI_Put` and `MPI_Get` function calls ensures proper synchronization and coherence of the shared data. Thus, we avoid the well known readers-writers problem in synchronization. Fig. 1 shows the general idea of this approach.

V. RESULTS

Our MC simulations were performed on a multicomputer *reef* in the Supercomputing and Networking Center in Poznań built up of 22 nodes with two dual-core Intel Xeon 3GHz CPUs and of 122 nodes with two quad-core Intel Xeon 2.33GHz CPUs with OpenMPI and InfiniBand technology of interconnect links. Due to avaibility of the cores we had to restrict our calculations to the maximum of 30 nodes. The selected simulation box sizes are: 100, 1000, 10000. The size of 10000 is large enough to distribute the simulation among many (more than 10) parallel processes and the size of 100 is sufficient to check the limits of performance for a relatively small simulation box size. 10^5 MCS were executed during each run of our MC simulations but starting with 10^4 MCS for thermalization as explained above. Each of the p processes keeps its part of the spin lattice in the form of the stripe of size $L \times L/p$ and they have to facilitate values of border spins to the neighbouring stripes, as explained above. The dimensionless temperature dependent parameter k_BT/J was varied in the range 1.8 to 2.7, and magnetization (i.e. the mean value of all spins s) presented in Fig. 2 was calculated using a different number of processes, as presented in Fig. 3.

The dashed curve shows the Onsager exact result which can be obtained in the thermodynamic limit. The vertical part of the dashed curve in Fig. 2 shows the Onsager phase transition point $K^{-1} = 2/\ln(1+\sqrt{2})$ suitable for a macroscopic system [5]. The presented magnetization curves $M(k_BT/J)$ show that spins s tend to be ordered on the left side of this vertical part of a dashed curve whereas on the right side spins tend to be disordered. The 2D Ising model has been solved exactly [5] and was picked up by us to check whether our parallelized code gives the proper results when the lattice is distributed among p processes. It is worth noting that the finite-size sample results from our simulations for $L = 1000$ give the critical temperature at $k_BT/J = 2.28$, which agrees quite well with the theoretical exact result $k_BT/J = 2.269$.

To determine scaling of our simulations with p parallel processes, we have calculated speedup S defined as $S = t_{seq}/(t_{par})$ and efficiency $E = S/p$ (see e.g. [14]), where t_{seq} and t_{par} denotes the sequential and parallel execution times of our program, respectively, and p denotes the number of parallel processes executing the program. These tests used a much lower number of MCS equal to 10^3, to reduce the time needed to wait for the results. This obviously leads to less accurate physical results, but that was not the point of this particular study.

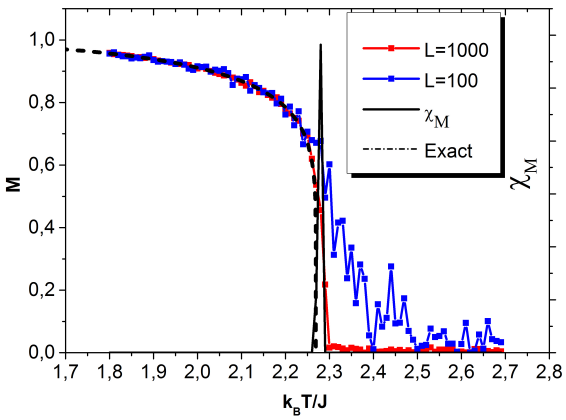

Fig. 2. The dependence of magnetization M and magnetic susceptibility χ_M on the temperature parameter k_BT/J calculated from our MC simulations with the lattice of size $L = 100, 1000$ (magnetic susceptibility only for $L = 1000$). The presented values of M are averaged over the ones obtained from the runs of our parallel simulations with a different number of processes. The dashed line represents the exact solution with the phase transition at $k_BT/J = 2.269$

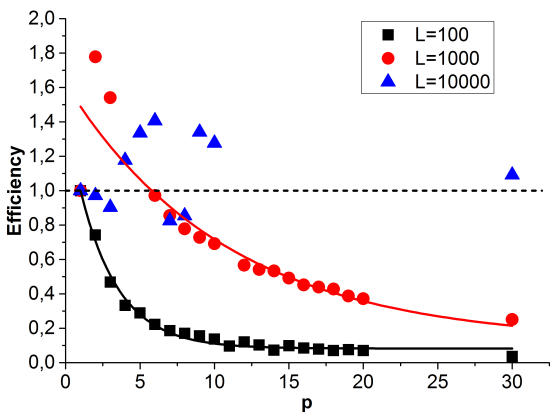

Fig. 3. The dependence of the efficiency E on the number of parallel processes p in our MC simulations run on the *reef* supercomputer. For comparison, the ideal efficiency is presented by a dashed line. Lines are drawn to guide an eye

The p dependence of efficiency E of our MC simulations with distributed processing of the lattice performed on the *reef* multicomputer is presented in Fig. 3. We observe the monotonous but not rapid decrease of efficiency for $L = 100$ and 1000. The obvious reason for this decrease was signalized above. Our MC program continuously flips spins on the

lattice and the number of boundary spins increases when we run more parallel processes. Thus, the overlay on communications between p parallel processes (proportional to the amount of data to be shared i.e. $2pL$) is evidently balanced to a high degree by the speedup of calculations when one increases the number p of parallel processes. Results for $L = 10000$ are scattered in some region, but in the given range of the number p of used processes the efficiency E oscillates around value 1, which is a very good indication for future research.

Fig. 4 presents speedup S for different lattice sizes and number of processes. For $L = 100$ and 1000 the maximum speedup is 2 and 6, respectively, meaning that our algorithm with $L = 1000$ will run at most 6 times faster compared to the sequential one. That plateau is formed due to parts of the algorithm that cannot be parallelized, also known as Amdahl's Law [15].

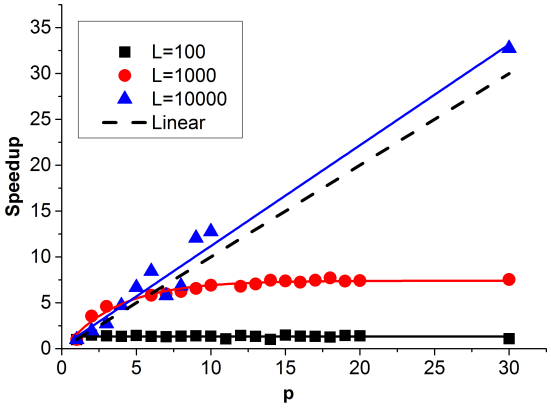

Fig. 4. Dependence of the speedup S on the number of parallel processes p in our MC simulations run on the*reef* supercomputer. For comparison, the ideal speedup is presented by a dashed line. Lines are drawn to guide an eye

We have obtained the best results for the lattice size $L = 10000$ where speedup S dependence is almost linear in Fig. 4. It should be noted that speedup S for $L = 10000$ is a bit above the maximum available ideal speedup marked by the dashed line. Part of this behavior is driven by the quality of the results, as we have two kinds of processors on the *reef* multicomputer, and such simulations take a long time to finish for a small number of processes and for that time stable computer performance cannot be guaranteed.

VI. CONCLUSIONS

Although our results are of preliminary character, they prove that the MC simulations can be successfully performed also with the lattice distributed among many multicomputer nodes. We acknowledge the useful tools for effective operations on the distributed lattice: remote memory access, the MPI memory window and one-sided communication [13]. The test was performed with up to 30 computing nodes, but stabilization of the efficiency in Fig. 3 looks promising for further extends to more nodes.

For the larger system size $L = 10^4$ the gain in performance when using parallelized algorithms is close to ideal, which gives the possibility to model spin systems with a lattice much greater than the one which can be fitted in the memory available on a single computer node.

Thanks to the message-passing parallelization model applied, our simulations can be executed on any kind of computing systems with shared, distributed and mixed type memory. Taking the MPI library [12, 13] to parallelize the computational processes, we simultaneously ensure efficiency, full portability and functionality of our application.

In addition we would like to note that the Ising model, which is a testing system in our paper, still finds new interesting applications. It is also used to calculate properties of quantum spin systems as applying Suzuki-Trotter transformation [16] one obtains a classical spin system but larger by one dimension compared to the original one.

The method proposed herein could also be used in simulations of systems with discontinuous phase transitions (e.g. systems with the same degrees of freedom $s = \pm 1$), where we find the hysteresis as a function of the coupled fields and the physical distribution of the system among computing nodes seems to be a better and more stable option [17]. This method could also be used to lattice models with external magnetic field or chemical potential. such as Hubbard U-J model [18], where cluster algorithms cannot be adapted.

Acknowledgments

This work was partly supported by the Polish Ministry of Science and Higher Education (MNiSzW) within the project No. N519 579138. Most numerical calculations were carried out on the platforms of the Poznań Supercomputing and Networking Center. The remaining part of simulations was performed on multicomputers in the Faculty of Physics at the Adam Mickiewicz University in Poznań.

References

[1] G. Musiał, L. Dębski, *Monte Carlo method with parallel computation of phase transitions in the three-dimensional Ashkin-Teller model*, Lect. Notes in Comp. Scie. **2328**, 535 (2002).

[2] G. Musiał, *Monte Carlo analysis of the tricritical behavior in a three-dimensional system with a multicomponent order parameter: The Ashkin-Teller model*, Phys. Rev. B **69**, 024407 (2004).

[3] L. Dębski, G. Musiał, J. Rogiers, *A Monte Carlo study of continuous non-Ising phase transitions in the 3D Ashkin-Teller Model using the OpenMosix cluster of Linux PCs*, Lect. Notes in Comp. Scie. **3019**, 455 (2004).

[4] S. Murawski, K. Kapcia, G. Pawłowski, S. Robaszkiewicz, *On the phase diagram of the zero-bandwidth extended hubbard model with intersite magnetic interactions for strong on-site repulsion limit* Acta Phys. Polon. A **121**, 1035 (2012).

[5] L. Onsager, *Crystal statistics. I. A two-dimensional model with an order-disorder transition* Phys. Rev. **65**, 117 (1944).
[6] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, *Equation of State Calculations by Fast Computing Machines*, J. Chem. Phys. **21**, 1087 (1953).
[7] D. P. Landau, K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*, Cambridge University Press, Cambridge 2000.
[8] R. H. Swendsen, J.-S. Wang, *Nonuniversal critical dynamics in Monte Carlo simulations*, Phys. Rev. Lett. **58**, 86 (1987).
[9] F. Wang, D.P. Landau, *Determining the density of states for classical statistical models: A random walk algorithm to produce a flat histogram*, Phys. Rev. E **64**, 056101 (2001).
[10] W. Prokovef, B. Svistunov, *Worm algorithms for classical statistical models* Phys. Rev. Letters **87**, 60601 (2001).
[11] H. G. Evertz, W. von der Linden, *Simulations on infinite-size lattices* Phys. Rev. Letters **86**, 5164 (2001).
[12] http://www.mpi-forum.org/ – MPI Forum Home Page.
[13] W. Gropp, E. Lusk, A. Skjellum, *Using MPI – 2nd Edition: Portable Parallel Programming with the Message-Passing Interface*, MIT Press, Cambridge 1999.
[14] E. F. Van de Velde, *Concurrent Scientific Computing*, Springer-Verlag, New York 1994.
[15] G. M. Ahmdal, *Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities* AFIPS Conf. Proc. **30**, 483 (1963).
[16] M. Suzuki, *Quantum statistical Monte Carlo methods and applications to spin systems*, J. Stat. Phys. **43**, 883 (1986).
[17] X. Qian, Y. Deng, H. W. J. Blöte, *Percolation in one of q colors near criticality*, Phys. Rev. E **72**, 056132 (2005).
[18] S. Murawski, K. Kapcia, G. Pawłowski, S.Robaszkiewicz, *Monte Carlo study of phase separation in magnetic insulators* Acta Phys. Pol. A **127**, 281 (2014).

Szymon Murawski was born in 1986 in Świnoujście, received the Master of Science degree in Nanotechnology in 2010 at the Adam Mickiewicz University in Poznań. From then on a PhD student at the same University. Research interest concentrate on numerical simulations of strongly correlated systems (mainly Hubbard model) with open system (grand canonical ensemble) and parallel classical Monte Carlo algorithms. Administrator of supercomputers $k1$ and $Bender$ at Faculty of Physics, also interested in IT: networking and security, programming, shell scripting, mobile applications development.

Grzegorz Musiał was born in Białożewin, Poland, in 1955. He holds the Professor position at the Faculty of Physics, Adam Mickiewicz University in Poznań, Poland. In his research he joins physics and computing. Recently, his research work in physics has concentrated mainly on many aspects of statistical physics in classical and quantum spin-lattice systems using numerical simulations. The computing research mainly concerns operating systems of the UNIX type and parallel computing in distributed environment, also with high heterogeneity. He is an author or co-author of 59 scientific papers and 3 books.

Grzegorz Pawłowski. Personal data: born 30 March, 1967, Poznań, Poland; Current Position: Associate Professor, Faculty of Physics, Adam Mickiewicz University, Poland; Professional Activities: Theory of strongly correlated systems, superconductivity, charge orderings, phase separations, disordered systems, computational physics (numerical methods, simulations: ALPS project - Algorithms and Libraries for Physics Simulations). Teaching: operating system LINUX/UNIX, programming in C/C++/C#, Python, Php, ASP.NET, SQL, XML, XHTML, JavaScript, and mobile systems.