CMST 13(1) 41-46 (2007)

DOI:10.12921/cmst.2007.13.01.41-46

COMPUTATIONAL METHODS IN SCIENCE AND TECHNOLOGY **13(1)**, 41-46 (2007)

Parallel Simulations of the Monte Carlo Type: 3D Ashkin-Teller Model

G. Musiał, L. Dębski and G. Kamieniarz

Institute of Physics, Adam Mickiewicz University, Umultowska 85, 61-614 Poznań, Poland
e-mail: gmusial@amu.edu.pl, gulf@spin.amu.edu.pl, gjk@amu.edu.pl

(Rec. March 28, 2006)

Abstract: The method of parallelization of processing in simulations of the Monte Carlo type is proposed and tested on the 3D Ashkin--Teller model characterized by a rich phase diagram. The message passing is applied and the MPI library is exploited. It is demonstrated that the method works well in different regions of the phase diagram, where the phase transitions are first-order and continuous. The dependence of speedup and efficiency on the number of parallel processes is studied. The condition for the best speedup and efficiency in these simulations is formulated and discussed using the results obtained on symmetric NUMA multiprocessor and on symmetric multicomputer. The suggestion as to effective use of the method even in highly heterogeneous computer systems is also given.

Key words: parallelization of processing, Monte Carlo simulations, 3D Ashkin-Teller model

I. THE MODEL

The Ashkin-Teller (AT) model [1] has been one of the important reference points in statistical physics for many decades as it is nontrivial generalization of the Ising model. For the latter exact solutions are known in dimensions 1 and 2 and it is a good testing case for new programs or applications. The interest in the AT model has much increased after the appearance of Fan's paper [2] in which he expressed the Hamiltonian H of this model

$$-\frac{H}{k_B T} = \sum_{[i,j]} \left\{ K_2 (s_i s_j + \sigma_i \sigma_j) + K_4 s_i \sigma_i s_j \sigma_j \right\} \qquad (1)$$

by means of two independent Ising degrees of freedom, s_i and σ_i, residing on each lattice site. $[i, j]$ denotes the summation over nearest neighboring lattice sites, $K_i = -J_i/k_B T$, with $i = 2$ or 4, and T is the temperature. Moreover J_2 is the coupling of the nearest neighbor interaction between the degrees of freedom s_i as well as for σ_i, whereas J_4 is the coupling between the product of these degrees of freedom $s_i \sigma_i$.

The three-dimensional (3D) standard AT model has been analyzed by the short series analysis and mainly by the Monte Carlo (MC) method by many authors (see [3] and the papers cited therein). The results are summarized in the phase diagram presented in Fig. 1 where all phases are shown and explained.

In this paper we present our method of parallelization of processing in the simulations of the Monte Carlo type taking a more general AT model as an example in which various kinds of ordering are realized and which has an interesting and complicated phase diagram. We explain the method of parallelization using our program [3] based on a single-flip Metropolis algorithm chosen because of its simplicity and typical structure. There are other algorithms of similar structure which flip clusters of spins leading to a significant reduction of the critical slowing down [4]. Recently an interesting algorithm has been published [5] which uses a random walk in the energy space to estimate the density of states of a system studied.

The test of the method proposed is performed using two points marked with $+'s$ on the phase diagram in Fig. 1, where the phase transitions are of different character – continuous and first-order. Comparing the speedup, we demonstrate that the method works well in both situations.

II. THE PARALLELIZED MONTE CARLO SIMULATIONS

The simplest way to perform such simulations is to run many sequential jobs but it was not satisfactory for our large-scale simulations [3]. It was necessary to execute over 5 000 MC runs to obtain the phase diagram presented in Fig. 1. Moreover, the larger the samples considered, the better the analysis of the results.

The MC simulations presented here are the ones reported in [3] to which we have applied our concept of parallelization. In this way one can perform simulations for much larger samples utilizing more CPUs simultaneously, both with shared and distributed memory, thanks to the message-passing parallelization model applied. Thus the idea is

of a more general character and may be applied to many types of simulations depending on the size of a system.

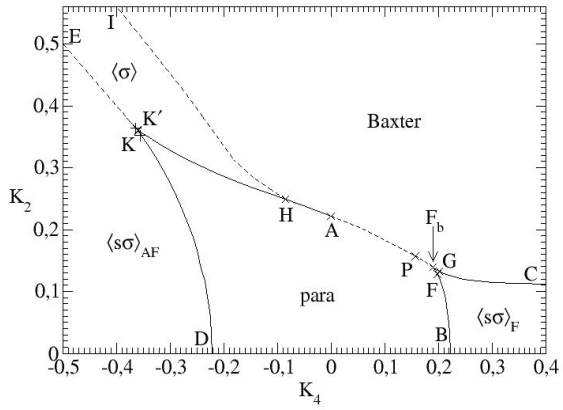

Fig. 1. The phase diagram of the 3D Ashkin-Teller model on a cubic lattice – the present state of knowledge. The broken lines denote the first-order phase transitions, whereas the solid lines – the continuous ones. The phase labeled Baxter is ferromagnetically ordered with $\langle s\rangle$, $\langle\sigma\rangle$ and $\langle s\sigma\rangle$ non-zero, whereas in the phase labeled para they are all zero. In the phases "$\langle s\sigma\rangle_{\mathrm{F}}$" and "$\langle s\sigma\rangle_{\mathrm{AF}}$" $\langle s\rangle = \langle\sigma\rangle = 0$ and only the parameter $\langle s\sigma\rangle$ is ferro and anti-ferromagnetically ordered, respectively. For the phase "$\langle s\rangle$" two equivalent phases exist in which $\langle s\sigma\rangle = 0$ and either $\langle s\rangle$ or $\langle\sigma\rangle$ is ferromagnetically ordered but the other is not. The letters A, F, G, H, K and K' denote the tricritical points, F_b is the bifurcation point and in point P the AT model is reduced to the 4-state Potts model. The *+'s* denote the points at which the test of our parallelization model was performed in this paper

We have generated equilibrium microstates (*i.e.* configurations of spins in the system) of the finite-size cubic spin samples of the size $L \times L \times L$ for fixed values of the model parameters, using the Metropolis algorithm, *i.e.* trying to reverse single spins. However, we had to drive the system into the equilibrium state independently in each parallel processes. For this purpose thermalization of the initial microstates of the length of order 10^6 Monte Carlo steps (MCS) was applied using completely different sets of random numbers in each parallel processes, as presented in Table 1. As usual, a MCS is completed when each of the lattice sites was visited once. The 64-bit random number generator was used. Periodic boundary conditions were imposed.

A MC run was split into k segments consisting of about 10^6 MCS and one partial average of each measured quantity was calculated from microstates in each segment to estimate the statistical scattering of the results. Only every l = 6-th to 10-th MC step contributes to the calculation of the partial averages to avoid correlations between sampled microstates of spins in the system and to sample microstates with the Gibbs distribution of probability.

Table 1. The scheme of organization of parallel processing in the simulations of the Monte Carlo type

Main activities in the parallel program	(comments in parenthesis)
Initialize the parallel MPI environment Get the number p of parallel processes Get the rank r of the current parallel process	(that the user has started for his program)
In 0th process: Input of the model parameters Input of the simulation parameters: i, m and l	(*i.e.* in the master process) (here these are L, K_2, K_4) (i is the number of partial averages per parallel process, m is the number of MCS per one partial average and only every l-th MCS is taken for calculations)
Send the model parameters and the simulation parameters i and m to the other parallel processes	
Produce $r*i*m*l*L3$ random numbers	(to use the different sets of random numbers for each parallel process)
Run $m*l*L^3$ MCS Calculate i partial averages and send them to 0th process	(thermalization) (the main part of the program)
0th process collects all partial averages, calculates and prints the results and their standard deviations	(here these results are the values of cumulants Q_L and V_L for spins s, σ and $s\sigma$)
Finalize the parallel MPI environment	

In order to decide whether to accept a single spin-flip or not, we compared the energies of the new (primed) and old microstates. If the energy change $E_{\alpha'} - E_\alpha$ was negative, then the new microstate was automatically accepted. If, however, it was positive, the new microstate was accepted with a probability $P = e^{-\beta(E\alpha' - E\alpha)}$ by comparing P with a random number. Physically it means that both microstates are in equilibrium and none of them arises at the expense of the other. Using this method, we generated equilibrium microstates which allowed us to calculate physical quantities in a direct way, according to the Metropolis algorithm. It is worth noting that only the use of different sets of random numbers for each parallel process ensures that the partial averages can be reliably regarded as independent. Such an importance sampling of microstates ensures that we spend the most part of computing time with microstates bringing a significant contribution to the values calculated.

The phase transition points were determined from the common intersection point of the curves [3, 6, 7]

$$Q_L = \frac{\langle M^2\rangle_L^2}{\langle M^4\rangle_L} \tag{2}$$

versus K_2 at fixed K_4, where $\langle M_\alpha^n\rangle_L$ denotes the n-th power of the α spins order parameter, with $\alpha = s$, σ or $s\sigma$, averaged over an assembly of independent samples of the size

$L \times L \times L$. This analysis is called the intersection method and was used in localization of points on the phase diagram in Fig. 1 (see [3] and the papers cited therein).

For the analysis of first-order phase transitions (together with the above mentioned cumulant Q_L) we calculate another fourth order cumulant V_L defined by [8]

$$V_L = 1 - \frac{\langle E^4 \rangle_L}{3\langle E^2 \rangle_L^2}, \tag{3}$$

where $\langle E^n \rangle_L$ denotes the n-th moment of the internal energy E averaged over an assembly of independent samples of the size $L \times L \times L$. Although V_L does not have any obvious physical interpretation, it is extremely useful in distinguishing the first-order phase transitions from continuous ones. For the latter $V_L = 2/3$ in the limit $L \to \infty$ and it remains fixed even for $T \neq T_c$ [8], whereas varying T for the first-order phase transitions, V_L has a minimum $V_\alpha^{\min}$ at $T = T_c(L)$, the finite-size critical temperature. Moreover [8]

$$V_L^{\min} = 1 - \frac{2(E_+^4 + E_-^4)}{3(E_+^2 + E_-^2)^2} \tag{4}$$

in the limit $L \to \infty$. Here $E_\pm = E(T \to T_c|_\pm)$. Thus when the latent heat $E_+ - E_-$ tends to zero, $V_L^{\min}$ approaches the value 2/3, as described above for the continuous phase transitions. It is also important that the functions $V_L^{\min}$ and $K_i(T_c(L))$ versus L^{-d} are linear [8], which allows one to obtain the limit value $V_\infty^{\min}$ and the true critical value of the couplings K_i. These considerations were carried out for a system with a single order parameter. In the paper [9] we have generalized the use of the cumulant V_L for a system with many order parameters, like the AT model. To determine the phase transition connected with the order parameter $\langle s \rangle$, the moments $\langle E^n \rangle_L$ in the cumulant (3) should be calculated only for the interactions between spins s (the first term in the Hamiltonian (1)). Analogously, we investigate the phase transitions connected with the order parameter $\langle \sigma \rangle$, whereas calculating the moments $\langle E^n \rangle_L$ in (3) for the transitions connected with the order parameter $\langle s\sigma \rangle$, we have to take into account only interactions between the spins $s\sigma$ (the third term in the Hamiltonian (1)).

There are several factors that today stimulate evolution of computing systems towards parallel ones. The finite speed of light and effectiveness of heat dissipation impose physical limits on the speed of a single computer. Moreover, the cost of an advanced single-processor computer increases more rapidly than its power of computing. A significant reduction of the price per performance ratios is achieved by utilizing networks of PCs or workstations as parallel computers. The list of the sites operating the 500 most powerful computer systems [10] suggestively manifests this tendency. One should also note the step by step growth of wide-area networks that can span the globe in a more or less distant future.

We have used the MPI library to parallelize the computational process in our simulations. The main reason for this choice is that message passing is the most effective and universal among the parallel computational models, as it effectively works in every computer system with the distributed, shared or mixed type of a memory. The most powerful computer systems now have the combined memory structure: the system is built of some distributed groups of processors (usually 2, 4 or 8) which share their local memory, and these groups are connected with intercommunication networks that keep up with speeds of advanced single processors. Although for many applications, including our simulations, the use of Ethernet for their communication environment is sufficient.

The message-passing model has some advantages. The fist of them is its universality as this model fits well separate processors connected by fast or slow communication network. This model is expressive in parallel algorithms which in addition are debuggable. Moreover, using more (distributed) memory and cache this model gives the better performance of a system.

There have been many attempts to work out a library for message-passing model as programming with sockets was of minimal functionality. Before formulating a message-passing standard, the most popular has become parallel virtual machine (PVM) [11] which is still in use. The MPI library is such a standard formulated for the first time in 1994 by MPI Forum [12] which still takes care for its standardization when formulating its new versions. MPI is only a specification, not a particular implementation like PVM. This standard ensures full portability for Fortran, C and C++ programs which can be compiled with ordinary compilers and linked with the MPI library. Moreover, MPI ensures both efficiency and functionality. For these reasons we prefer to take MPI for message-passing in our simulations.

Besides after driving the system to the thermodynamic equilibrium independently on each of the p parallel processes, as mentioned above, different processes of the parallelized job calculate their parts of k partial averages of the moments of an order parameter M and of the internal energy E. Speedup u of such a job is defined as $u = t_{\text{ser}}/t_{\text{par}}$, where t_{ser} and t_{par} denote the sequential and parallel execution time, respectively. In this paper, to test the method, we have used such numbers p of parallel processes which are integer divisors of $k = 90$, the number of partial averages calculated. In this way we have the same number of partial averages calculated by each parallel process and each MC run had approximately the same amount of computational work.

Thus, assuming $k = ip$, the speedup u in the ideal case, *i.e.* when there are no latencies between parallel processes, should be equal to

$$u = \frac{t_0 + ipt_1}{t_0 + it_1} \tag{5}$$

where t_0 is the computing time for leading the system to the thermodynamic equilibrium, and t_1 is the computing time for one partial average of the moments of an order parameter M and of the internal energy E. Usually $t_0 = t_1$, then we obtain

$$u = \frac{1 + ip}{1 + i}, \tag{6}$$

When the number i of partial averages per process is $i \gg 1$, we obtain the ideal speedup $u = p$.

III. TESTS OF THE METHOD

To test our method of parallelization of processing in the simulations, we have calculated the cumulants Q_L and V_L from partial averages of the moments of an order parameter M and of the internal energy E for spins s, σ and $s\sigma$. We have used two points marked with $+'s$ on the phase diagram in Fig. 1. The first point at (–0.356, 0.352) lies in the critical region of the continuous phase transition [3]. For this test, we have fixed the parameters of the simulation: the size of the cube $L = 16$, the number of all partial averages $k = 90$, the number of MCS performed on driving the system to the thermodynamic equilibrium and the number of MCS for calculation of one partial average were equal to $m = 8 \times 10^5$. The results of our test carried out on SGI Origin 3800 multiprocessor of NUMAflex™ architecture with R12000 processors are presented in Table 2.

Table 2. Runtimes t (in 10^3s), speedups u, total CPU times t_{CPU} (in 10^3s) and efficiencies $E = u/p$ of parallel Monte Carlo jobs run on SGI Origin 3800 supercomputer versus the number p of parallel processes, i is the number of partial averages per a parallel process

p	i	t	u	t_{CPU}	E
1	90	178.8	1.0	178.8	1.00
2	45	86.9	2.0	173.7	1.00
3	30	58.6	3.0	175.6	1.00
5	18	36.1	5.0	180.2	1.00
6	15	30.2	5.9	181.4	0.98
9	10	20.8	8.6	187.1	0.96
10	9	18.9	9.4	189.0	0.94
15	6	13.3	13.4	199.4	0.89
18	5	11.4	15.6	205.0	0.87
30	3	7.7	23.3	228.3	0.78
45	2	5.8	30.9	257.6	0.69

The dependence of the speedup u versus the number p of parallel processes is illustrated in Fig. 2. For comparison, the dashed line represents the ideal speedup $u = p$. Although Fig. 2 looks similar to the Amdahl's law, the main reason for such a run of $u(p)$ is of different nature. It follows from the structure of the algorithm of MC simulations, namely the part of the computing time in each parallel process is used for independent driving of the system to the thermodynamic equilibrium.

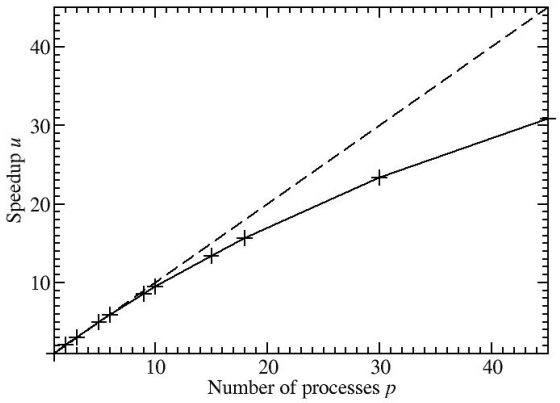

Fig. 2*. The dependence of the speedup u versus the number p of parallel processes for parallel Monte Carlo jobs run on SGI Origin 3 800 supercomputer

One can see from Table 2 and Fig. 2 that the latencies were small during these computations, because the reduction of the speedup compared with the ideal case is mainly the effect of the decrease in the ratio of the number of MCS used for calculation of partial averages (the results of the simulations) in a process and the number of MCS used for leading the system to the thermodynamic equilibrium. Besides the parallel overhead (coming from communication operations and redundant computations), also the part of computations performed sequentially is small compared to those performed in parallel. This is clearly seen from the point with $p = 45$ for which only two partial averages per process are calculated. We have to remember that driving the system to the thermodynamic equilibrium takes the same time as the calculation of one partial average, which reduces the speedup by the factor approximately equal to 1/3 in this point.

We have performed these simulations also on the SUN multicomputer (built up from units with two dual-core AMD64 Opteron 2GHz CPUs) obtaining a similar picture

* Figures 1 and 2 were obtained with the Grace program developed by Grace Development Team, version 5.1.14, which is distributed with the GNU General Public License

as that presented in Fig. 2. Only the times listed in Table 2 should be decreased by a factor 0.27.

We can conclude from these considerations that during the simulations one should keep the number of partial averages per process $i \gg 1$ as followed from the theoretical considerations presented in the previous section. Then the best speedup is obtained and it is not at the expense of a significant increase in the total CPU time t_{CPU} or a similar decrease in the efficiency $E = u/p$, as presented in the last two columns of Table 2.

These conclusions hold for simulations in the region of continuous phase transitions. We have obtained a similar picture as shown in Table 2 and in Fig. 2 when we have performed simulations for the point (–0.364, 0.364) marked with + on the phase diagram in Fig. 1. This point lies in the critical region of the first-order phase transition [3]. Thus, comparing the speedup in the regions where phase transitions are of different character – continuous and first-order, we demonstrate that the method works well in both situations.

IV. CONCLUDING REMARKS

In our paper we have presented our method of parallelization of processing in simulations of the Monte Carlo type and demonstrated on the example of the 3D Ashkin-Teller model but the idea is of more general character and may be applied to many types of simulations, as most of them depend on the size of a system. A simultaneous use of more CPUs permits the simulations for much larger samples and they can be executed on computing systems both with shared and distributed memory, thanks to the message-passing parallelization model applied. Taking the MPI library to parallelize the computational processes, one simultaneously ensures efficiency, full portability and functionality of an application.

Comparing the speedup of our simulations for two points at which the phase transitions are of different character – continuous and first-order, we demonstrate that the method works well in both situations.

We also specify the condition at which such simulations can be parallelized effectively, *i.e.* without a significant increase in the total CPU time (or by a similar decrease in the efficiency E): the number i of partial averages per process should be $i \gg 1$. Then the maximum speedup is achieved which is almost linear as the function of p in SMP systems where there are small latencies. Even in highly heterogeneous systems almost ideal speedup and efficiency can be obtained when one uses the environment allowing load balancing, as we have combined openMosix and MPICH environments [13].

The scalability of these simulations, ability of their execution on any distributed systems, even on relatively cheap Ethernet network of PCs, makes them a very attractive tool. Using more and more CPUs, one can consider larger physical systems what leads to increasingly credible results.

Acknowledgements

This work was supported in part by the Polish Ministry of Education and Science under Grant 4 T11F 014 24 and the European Commission under the project MAGMANet NMP3-CT-2005-515767. Numerical calculations were mainly carried out on the platforms of the Poznań Supercomputing and Networking Center.

References

[1] J. Ashkin and E. Teller, Phys. Rev. **64**, 178 (1943).
[2] C. Fan, Phys. Lett. **39A**, 136 (1972).
[3] G. Musiał, Phys. Rev. **B69**, 024407 (2004).
[4] R. H. Swendsen, J.-S. Wang, Phys. Rev. Lett. **58**, 86 (1987); U. Wolff, Phys. Rev. Lett. **62**, 361 (1989).
[5] F. Wang, D. P. Landau, Phys. Rev. **E64**, 056101 (2001).
[6] K. Binder and D. W. Heerman, *Monte Carlo Simulation in Statistical Physics*, Springer Series in Solid State Physics, Vol. 80, Springer-Verlag, Berlin 1988.
[7] K. Binder and D. P. Landau, Phys. Rev. B30, 1877 (1984).
[8] M. S. S. Challa, D. P. Landau, K. Binder, Phys. Rev. **B34**, 1841 (1986).
[9] G. Musiał and L. Dębski, Lect. Notes in Comp. Scie. **2328**, 535 (2002).
[10] http://www.top500.org/
[11] http://www.csm.ornl.gov/pvm/pvm_home.html/ – PVM Home Page
[12] http://www.mpi-forum.org/ – MPI Forum Home Page
[13] L. Dębski, G. Musiał and J. Rogiers, Lect. Notes in Comp. Scie. **3019**, 455 (2004).

GRZEGORZ MUSIAŁ was born in Białożewin, Poland, in 1955. He received his MSc degree in theoretical physics in 1978 from the Faculty of Mathematics, Physics and Chemistry of the Adam Mickiewicz University, at which he has worked till now. In 1987 he earned his PhD degree in solid state physics and he has habilitated in computer physics in 2004, at the same University. In his research he joins physics and computing. Recently, his research work in physics concentrates on many aspects of statistical physics in classical spin-lattice systems using numerical simulations (Ising and Ashkin-Teller models) and in such a systems with itinerant interacting fermions (the Falicov-Kimball model), also quantum systems, mainly magnetic clusters and single molecule magnets, and their utilization to technological developments within the EU network of excellence MAGMANet. 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 43 papers in major professional journals and conference proceedings in physics and computing.

LECH DĘBSKI received his PhD degree in physics at the Adam Mickiewicz University in Poznań in 2001 and he is employed in the Computational Physics Division at the same University. Since 1990s he has been involved in many computing projects, simulation and parallel programming and distributed information systems. His current fields of interest lie in two areas: physics and computing. In physics these are modeling and Monte Carlo simulations in spin-lattice models, obeying also systems containing itinerant interacting fermions, whereas in computing his research work concerns mainly next generation clusters and grids, parallel programming in distributed environments.

GRZEGORZ KAMIENIARZ is a professor in theoretical and computational physics at the Institute of Physics of the Adam Mickiewicz University in Poznań. His field of interest covers statistical mechanics, phase transitions, phenomenological modeling and simulations of the low-dimensional magnetic systems including molecular nanomagnets.