Accelerating Parallel Monte Carlo Simulations for Statistical Physics: Portability on Many-core Processors

Naw Safrin Sattar
Department of Computer Science
University of New Orleans
New Orleans, Louisiana, USA
nsattar@uno.edu

Ying-Wai Li
Computer, Computational, and Statistical Sciences Division
Los Alamos National Laboratory
Los Alamos, New Mexico, USA
yingwaili@lanl.gov

ABSTRACT

Monte Carlo (MC) simulations are important tools for studying thermodynamics and materials properties at finite temperature. Scaling up to larger systems allows for simulations comparable to experimental length scale. In this work, we focus on improving the weak scaling performance to simulate large magnetic spin systems up to millions of atoms. We parallelize the computation of physical observables within a single random walker using checkerboard domain decomposition implemented with OpenMP, and compare the performance of the same code base on different many-core processors. We analyze the effects of thread affinity and hyper-threading on the runtime performance, and achieve 14/12/6× speedups for Arm/Intel/AMD architectures respectively for the largest system we report in this work (about 4.2 millions spins). This study is helpful in informing end-users on the choice of the best multi-core architecture suited to their needs for similar pattern of parallel applications.

KEYWORDS

Monte Carlo simulation, parallel algorithm, domain decomposition, checkerboard algorithm, OpenMP, multi-core architecture, scalable algorithm, many-core processors, portability, thread affinity, hyper-threading

ACM Reference Format:
Naw Safrin Sattar and Ying-Wai Li. 2021. Accelerating Parallel Monte Carlo Simulations for Statistical Physics: Portability on Many-core Processors. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Monte Carlo (MC) simulations are a well-known technique that can be used in different fields of engineering and science to obtain a probability distribution of possible outcomes of a system or model. MC simulations are often deemed "embarrassingly parallel" when strong scaling is concerned, because multiple random walkers can be distributed to scale to a large number of processors and run in parallel independently without communications. Nevertheless, the need for simulations for large systems is important because it allows for the comparison of simulations to experimental length scale, but weak scaling is generally harder to achieve in MC methods because of the serial nature in the acceptance step where the detailed balance condition has to be satisfied. In this work, we focus on achieving weak scaling by making use of more processors to simulate larger system sizes, which offers an additional layer of parallelism to strong scaling. We parallelize the computation for large systems within a single random walker using a special domain decomposition technique, the checkerboard algorithm, and examine the performance portability of the massively parallel MC simulations on different many-core architectures using the same code base. We also study the effect of thread affinity on the parallel performance.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 METHODOLOGY

In this work our MC algorithm of choice is the Metropolis MC sampling [4], though the parallelization is general and is readily transferable to other MC algorithms. We sample the configuration space of a 2D Ising spin model Hamiltonian, and generate the Boltzmann distribution in energy, $P(E)$, at a given temperature, from which the ensemble averages of energy and other physical observables are calculated. In the simplest Ising model only nearest neighbor interactions are considered. The energy function, or the Hamiltonian $\mathcal{H}$, takes the form:

$$\mathcal{H} = -J \sum_{\langle ij \rangle} \sigma_i \sigma_j, \tag{1}$$

where J is the interaction strength, σ_i (σ_j) denotes the spin direction at site i (j) and the sum runs over all the nearest-neighbor spin pairs. We apply the checkerboard algorithm, a specialized technique of domain decomposition for spin models with nearest neighbor interactions [2]. In the checkerboard algorithm, the system is decomposed into two subsystems denoted by the black and white colors in a similar pattern as a checkerboard. Applying a spin flip on a black box only affects interactions with the surrounding white boxes. Therefore, parallelization is achieved by performing spin flips on all black boxes altogether. Spin flips alternate between black boxes and white boxes. We parallelize the "black" and "white" sublattices separately. We convert the 2D Ising spin configurations into a one-dimensional array according to this formula: $i = L * x + y$, where i is the index of the 1D array, L is the length of the system so that the number of spins $N = LxL$, x and y are the Cartesian coordinates. So, the data is stored sequentially in a 1D array where

black and white cells are congregated. We implement the parallel domain decomposition using OpenMP.

3 EXPERIMENTAL SETUP AND MC SIMULATION PARAMETERS/DATASET

We have performed all experiments on Darwin [1], an ASC funded heterogeneous cluster with many kinds of nodes available with different architectures. We run experiments on different configurations of ARM, AMD and Intel multi-core architectures to compare the performance and portability of our code. Detailed description of the architectures are included in poster.

We consider Serial [3] MC simulations of 20, 000 MC steps for temperature $T = 3.0$ as a baseline for our multi-threaded parallel MC simulation. We run the experiments with grid size up to 2048, i.e. 4.2 million spins.

4 RESULT

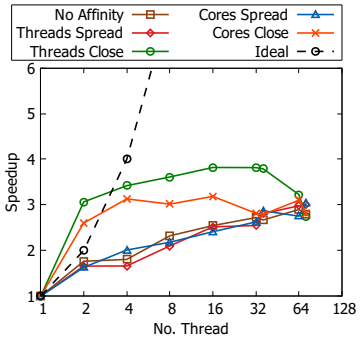


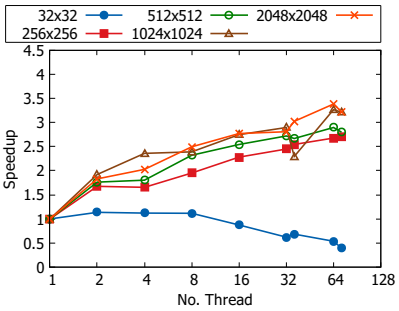
Figure 1: Effect of thread affinity examined on Intel Cascadelake. Experiments performed on a 512 × 512 system size

We measure the performance of our parallel MC simulation on different multi-core architectures based on speedup with the serial MC simulation. It is often much harder to achieve ideal scaling with weak scaling. Ideal scaling is easier to accomplish in strong scaling because the runtime scales inversely proportional to the number of random walkers employed. In weak scaling multiple communications after each parallelization step is necessary. There are all-reduce operations to calculate the total energy after each parallelization step. Again, switching between the black and white sublattices induces overhead for context-switching and makes it difficult to achieve ideal scaling.

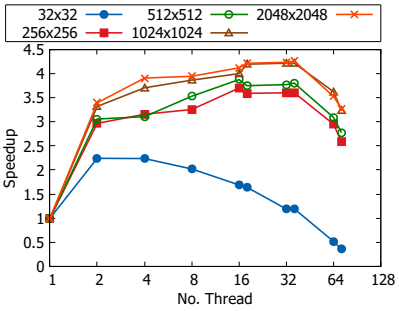
We try different thread affinity options to check which one provides better performance. We find that placing threads close together: OMP_PLACES=threads,OMP_PROC_BIND=close results in largest speedup compared to thread spreading shown in Figure 1. Figures 2a and 2b show the performance gain (1.26×) over speedup for Intel cascadelake when thread affinity is used. The performance for all other architectures is included in the poster due to space limitation.

5 CONCLUSION

In this study we show the performance and portability of parallel MC simulations on different multi-core architectures. We find



(a) without affinity



(b) <thread,close> affinity

Figure 2: Performance on Intel Cascadelake. Skylake shows a similar pattern with slightly different speedup factors.

that thread affinity helps to get optimal performance when the threads are put close together. This yields to gain in speedup of about 1.44/1.23/1.29× for ARM/AMD/Intel architectures respectively. Thread affinity also allows gaining the best performance with minimal number of threads. ARM(Cavium) and Intel Xeon Phi chips are slower for serial code (due to lower clock rates) but achieve larger speedup when multithreaded. Hyperthreading results in higher speedup on ARM(Cavium)-partial[128/256], Intel Xeon Phi-full[272/272], but not in other architectures. Larger systems provide more computational work, yielding better speedup, given our ultimate goal for weak scaling. Similar performance behavior is expected for higher dimensional systems in MC simulations. Overall, this study is helpful to understand the performance for not only MC simulations but also other domain decomposition applications.

REFERENCES

[1] Charles Kristopher Garrett. 2018. The Darwin Cluster. (6 2018). https://doi.org/10.2172/1441285
[2] Dieter W Heermann and Anthony N Burkitt. 1990. Parallelization of the Ising Model and Its Performance Evaluation. *Parallel Comput.* 13, 3 (March 1990), 345–357. https://doi.org/10.1016/0167-8191(90)90137-X
[3] Ying Wai Li, Krishna Chaitanya Pitike, and USDOE. 2019. Oak Ridge Wang-Landau (OWL). https://doi.org/10.11578/dc.20201001.79
[4] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics* 21, 6 (1953), 1087–1092.