Computer Physics Communications 195 (2015) 84–94

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Multi-GPU-based Swendsen–Wang multi-cluster algorithm with reduced data traffic

Yukihiro Komura *

RIKEN, Advanced Institute for Computational Science, 7-1-26, Minatojima-minami-machi, Chuo-ku, Kobe, Hyogo 650-0047, Japan
Department of Physics, Tokyo Metropolitan University, Hachioji, Tokyo 192-0397, Japan

ARTICLE INFO

Article history:
Received 2 December 2014
Received in revised form
28 April 2015
Accepted 28 April 2015
Available online 11 May 2015

Keywords:
Monte Carlo simulation
Cluster algorithm
Ising model
Parallel computing
Multi-GPU

ABSTRACT

The computational performance of multi-GPU applications can be degraded by the data communication between each GPU. To realize high-speed computation with multiple GPUs, we should minimize the cost of this data communication. In this paper, I propose a multiple GPU computing method for the Swendsen–Wang (SW) multi-cluster algorithm that reduces the data traffic between each GPU. I realize this reduction in data traffic by adjusting the connection information between each GPU in advance. The code is implemented on the large-scale open science TSUBAME 2.5 supercomputer, and its performance is evaluated using a simulation of the three-dimensional Ising model at the critical temperature. The results show that the data communication between each GPU is reduced by 90%, and the number of communications between each GPU decreases by about half. Using 512 GPUs, the computation time is 0.005 ns per spin update at the critical temperature for a total system size of $N = 4096^3$.

1. Introduction

Computational applications that use accelerator/co-processor technology are increasingly popular, not only amongst personal users but also in large-scale computing. A list of the world's most powerful supercomputers [1] showed that a total of 75 systems were using accelerator/co-processor technology in November 2014, up from 62 in June that year. Of these 75 systems, 50 employ NVIDIA GPUs. To perform calculations on accelerators such as GPUs, NVIDIA released the Compute Unified Device Architecture (CUDA) [2]. CUDA is a parallel computing platform and programming model that is essentially C/C++ with a few extensions to allow one to execute functions on NVIDIA GPUs.

The Ising model is a simple and standard model in statistical physics, and the three-dimensional (3D) Ising model has been used to test numerical methods for statistical mechanics systems. Nevertheless, accurate analyses of the 3D Ising model are still being studied [3,4], as it is difficult to guarantee the accuracy of critical exponents in a small system because of the effect of correction exponents. To reduce the effect of correction exponents, the 3D Ising model computations should consider a large system size.

The Metropolis algorithm [5] is widely used in Monte Carlo simulations. However, single spin-flip methods such as the Metropolis algorithm are sometimes plagued by long autocorrelation times (known as critical slowing down) at the critical temperature of the phase transition. For example, the autocorrelation time of the 3D Ising model with the Metropolis algorithm grows like the square of the linear system size at the critical temperature [6]. To overcome the problem of critical slowing down, cluster spin-flip algorithms have been proposed. Typical cluster spin-flip algorithms include the multi-cluster algorithm [7] and the single-cluster spin-flip algorithm [8]. The autocorrelation time can be drastically decreased using a cluster spin-flip algorithm.

Studies of the Metropolis algorithm on a GPU for large systems have been reported by several researchers [9–11]. However, as the Metropolis algorithm suffers from the problem of critical slowing down, it is highly desirable to apply parallel computations to cluster algorithms. The application of the Swendsen–Wang (SW) multi-cluster algorithm on a GPU has been studied by several researchers. The present author and Okabe [12] proposed an SW multi-cluster algorithm with a single GPU that uses the idea of Hawick et al. [13], which

* Correspondence to: RIKEN, Advanced Institute for Computational Science, 7-1-26, Minatojima-minami-machi, Chuo-ku, Kobe, Hyogo 650-0047, Japan. Tel.: +81 90 3597 0297.
E-mail address: yukihiro.komura@riken.jp.

http://dx.doi.org/10.1016/j.cpc.2015.04.025
0010-4655/

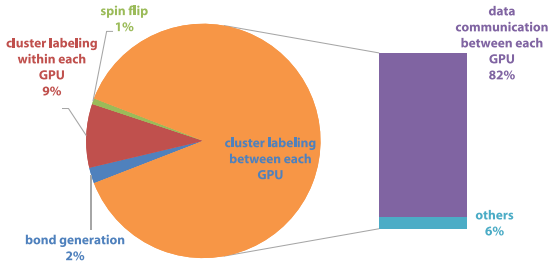

Fig. 1. Percentage of total computation time for the 3D Ising model at the critical temperature using the methods described in [20] and [15] without overlapping communication and computation. The SW multi-cluster algorithm with multiple GPUs developed in [15] consists of four steps: bond generation, cluster labeling within each GPU, cluster labeling between each GPU, and spin flip. For cluster labeling within each GPU, I use the method reported in [20]. The left-hand chart shows the percentage of the total computation time required for each step. Data communication occurs in the cluster labeling between each GPU. Thus, this step can be separated into the data communication between GPUs, other parts of the GPU computation, and checking the termination condition. The right-hand bar chart shows a breakdown of time for the cluster labeling between each GPU.

is called the label equivalence method, and the improved idea proposed by Kalentev et al. [14]. They also extended the SW multi-cluster algorithm to multiple GPUs [15]. Unlike other parallel Monte Carlo update algorithms, domain decomposition of the SW multi-cluster algorithm on multiple GPUs obeys a detailed balance, because the generation of each cluster bond is independent from all others, and the flipping of each cluster spin can be performed independently of every other cluster spin [15–17]. The SW multi-cluster algorithm with multiple GPUs has been applied to large-scale Monte Carlo simulations of the 2D classical XY model up to a system size of $65\,536 \times 65\,536$ [18]. More recently, sample programs were published [19] for the CUDA computation of the SW multi-cluster algorithm, and the present author [20] also proposed an SW multi-cluster algorithm with a single GPU that does not use conventional iteration. Weigel [16] proposed an SW multi-cluster algorithm that combined a self-labeling algorithm with a label relaxation or hierarchical sewing algorithm. Similar to the breadth-first search and tree-based union-and-find approach, the self-labeling algorithm is used to partition a set of elements into disjoint subsets. More recently, Wende et al. [21] proposed an SW multi-cluster algorithm as a combination of local cluster search and global label reduction by means of atomic hardware primitives. They calculated each sub-lattice using the self-labeling method, and then applied a novel label reduction across sub-clusters that used atomic hardware primitives to resolve label equivalences. In that paper, they showed that their code could achieve a factor-of-two performance gain over two previous methods [12,16].

With multiple GPUs, computational performance can be degraded by the data communication between each GPU. Along with Okabe, I proposed a four-step SW multi-cluster algorithm for multiple GPUs [15]. This method consists of bond generation, cluster labeling within each GPU, cluster labeling between each GPU, and the spin flip. The computation time for each step is shown in Fig. 1. From Fig. 1, we can see that data communication between each GPU makes up a significant proportion of the total computation time. Thus, to realize high-speed computation with multiple GPUs, we need to minimize the cost of data communication between each GPU. In this paper, I propose an SW multi-cluster algorithm for multiple GPUs that reduces the data traffic between each GPU. I realize this reduction of data traffic by adjusting the connection information between each GPU in advance. The performance of this multiple GPU calculation for the 3D Ising model is then tested. The remaining part of this paper is organized as follows. In Section 2, I briefly describe the calculation of SW multi-cluster algorithms on a single CPU and on multiple CPUs under distributed memory computing. In Section 3, I review the calculation of SW multi-cluster algorithms with a single GPU and multiple GPUs [15]. Section 4 introduces the proposed method for the SW multi-cluster algorithm on multiple GPUs with a reduction in data traffic between each GPU. In Section 5, I test the performance of this multiple GPU calculation on TSUBAME 2.5, a supercomputer located at the Tokyo Institute of Technology. A summary and discussion are given in Section 6.

2. CPU computation of the Swendsen–Wang multi-cluster algorithm

The Hamiltonian of the Ising model is given by

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} s_i s_j, \quad s_i = \pm 1, \tag{1}$$

where J is the coupling and s_i is the spin at lattice site i. The summation is taken over the nearest neighbor pairs $\langle i, j \rangle$, and periodic boundary conditions are employed.

In the SW multi-cluster algorithm [7], we separate the spins into clusters, and then flip all of the spins that belong to the same cluster. The SW multi-cluster algorithm for the Ising model consists of three main steps:

(1) *Active bond generation*: Construct a bond lattice of active or non-active bonds with probability $p = 1 - e^{2J/T}$, where T is the temperature.
(2) *Cluster labeling*: The active bonds partition the spins into clusters. These are identified and labeled using a cluster-labeling algorithm.
(3) *Spin flip*: All spins in each cluster are set randomly to $+1$ or -1.

In the cluster labeling step, the Hoshen–Kopelman algorithm [22], which assigns the proper cluster label to each site in a sequential manner, is widely used in CPU implementations.

The Hoshen–Kopelman algorithm is realized by sequential computations on a single CPU. To extend from a single CPU implementation to a multiple CPU implementation with distributed memory, we need to only consider the cluster labeling method with multiple CPUs by including information about the new spin state in the label information. Several cluster labeling algorithms with multiple CPUs have been reported. These approaches all execute the cluster labeling on each CPU using the Hoshen–Kopelman algorithm, and then perform the cluster labeling between each CPU. The approach to cluster labeling between each CPU differs in each method: there is a master–slave method [17], a method that prepares a global label array [23], and a relaxation method that exchanges boundary information between neighbors until all labels are unchanged [24,25]. As a refinement of this relaxation method, Bauernfeind et al. [26] proposed a technique