The Investigation of Models of Self-Organized Systems by Parallel Programming Methods Based on the Example of an Abelian Sandpile Model

Vahagn, Poghosyan
Institute for Informatics and Automation Problems NAS RA, 0014 Yerevan, Armenia
e-mail: povahagn@gmail.com

Suren, Poghosyan
Institute for Informatics and Automation Problems NAS RA, 0014 Yerevan, Armenia
e-mail: psuren55@yandex.ru

Hayk, Nahapetyan
Institute for Informatics and Automation Problems NAS RA, 0014 Yerevan, Armenia
e-mail: hayknahapetyan@yahoo.com

ABSTRACT

In this work the issues of study and behavior modeling of self-organized systems by means of parallelization algorithms in multiprocessor systems are observed. In particular, the software systems created on the base of a developed algorithm make a transition of Abelian sandpile model in square lattice from unstable state to a stable one by means of parallel topplings of nodes. Software programming is implemented by means of CUDA and OpenMP technologies. A graphical representation of test results carried out on processes with various capacities and video cards is available here. The obtained graphic curves enable researchers to make a comparative analysis, and depending on the model sizes, cores and the number of available processors, to choose an appropriate program structure performed in optimal period of time.

Keywords

Abelian sandpile model, parallel programming, cellular automata.

1. INTRODUCTION

The concept of self-organized criticality was proposed by Bak, Tang and Wiesenfeld in 1987, and has given rise to growing interest in the study of self-organizing systems. Bak et al argued that in many natural phenomena, the dissipative dynamics of the system is such that it drives the system to a critical state, thereby leading to ubiquitous power law behaviors. This mechanism has been invoked to understand the powerlaws observed in turbulent fluids, earthquakes, distribution of visible matter in the universe, solar flares and surface roughening of growing interfaces [1,2].

The Sandpile models being a class of cellular automata are among the simplest theoretical models which show selforganized criticality. An especially nice subclass consists of the so called abelian sandpile models (ASM) [3].

There have been many numerical[4, 5, 6] as well as analytical[7, 8, 9] studies of the ASM.

In numerous works in the absence of analytical relations of various physical characterizers different methods of statistical analysis have been applied for calculation or verification of hypothetical and analytical expressions. By means of numerous independent experiments changing the lattice sizes and the number of the filled sand grains in sandpile models, a lot of statistical data were obtained, on the basis of which (in particular, by the method of Monte Carlo) the graphic curves of the studied characterizers were built. The program, modeling the independent experiment, was concurrently performed on a cluster system with separate memory nodes. The lack of such a system, the demand of accurate calculation of physical characterizers that would dramatically increase the lattice sizes, as well as time constraints for calculations put forward not only a project realizing an independent experiment [10], but also the requirement for parallelization of modeling algorithm. Now we shall present the formal description of the sandpile model.

Consider an undirected graph $G = (V, E)$ with vertex set G and edge set E . A random variable h_i, which takes the integer values, is attached to each site $i \in V$, representing the height of sand at that site. The collection of heights $h_i \in V$ defines a configuration C_T of the system at a given discrete time T .

Each vertex i of the graph G is described by its maximal allowed height h_i^{max} . A configuration is called stable if all heights satisfy $h_i < h_i^{max}$.

We consider two types of vertices - closed and open. If $h_i^{max} = deg(i)$ a vertex is called closed, and a vertex i is called open if $h_i^{max} > deg(i)$. Here $deg(i)$ indicates the degree of i , the number of neighboring vertices of i.

The dynamics of the system is defined by the following rules. Consider a stable configuration C_T at a given time T . We add a grain of sand at a random vertex $i \in V$ by setting $h_i \to h_i + 1$ (we assume that the site is chosen randomly with a uniform distribution on the set V). This new configuration, if stable, defines C_{T+1} . If h_i is bigger than h_i^{max} , the site becomes unstable and topples, losing h_i^{max} grains of sand, while all neighbors of i receive one grain. Note that if the vertex is open the system loses grains. During the toppling of the closed vertices, the number of grains is conserved.

Note that a toppling of a vertex may cause some of its neighbouring vertices to become unstable. In this case they also topple according to the same toppling rule. Once all unstable sites have been toppled, a new stable configuration C_{T+1} is obtained.

If the finite connected graph G has at least one open vertex, then all vertices become stable after finite number of topplings, and the new stable configuration is independent of the order of topplings. Therefore the dynamics is well defined.

Let $a\hat{}_i$ be an operator, which acts on sandpile configurations and adds a grain at site i . It can easily be shown that $\hat{a_i}\hat{a_j} = \hat{a_j}\hat{a_i}$. This is the reason why the sandpile model is called Abelian.

Now consider the Abelian sandpile model on the finite $n \times n$ square lattice L with open boundary conditions, for which we have $h_i^{max} = 4$ for all vertices $i \in V$. Therefore all internal vertices are closed, while all boundary vertices are open.

The question is to find observables of the Abelian sandpile model during long-time evolution and in the limit of large lattices. Namely, the average height is 3.125. Our aim is to write efficient algorithm to perform simulations for high-precision computations.

In different topological spaces in the base of the method proposed for parallel calculations of physical characterizers of Sandpile models lies the concept of parallel implementation of topplings with vertex surfaces of the graph describing the space. We have developed a parallel description of technical implementation for a square lattice and Tor. It was applied to the real-time super accurate calculation of one of the important physical characterizers with node average density. Now let us give the algorithm description of toppling parallelization.

2. THE ALGORITHM DESCRIPTION OF TOPPLING PARALLELIZATION

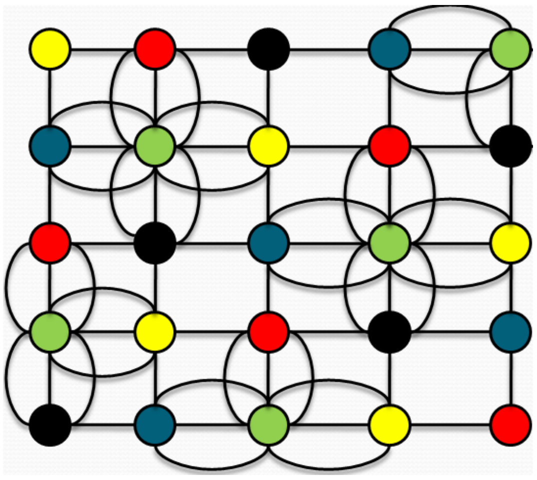

Figure 1: Example of complete coverage with independent vertices

The nodes will be called independent, if they do not have a common neighbor. To perform parallel topplings, concurrently toppled vertices should meet the requirement of being independent. Division of the set of square lattice nodes into a set of independent vertices is carried out in the following way. Let us introduce an information field (ID) for vertices, according to which the coloring of vertices will be defined. The vertex ID, accordingly the color as well will be defined by $(i + 2j) \bmod 5$ where $0 \le i < n$ and $0 \le j < n$ private formula, where i and j are considered to be arguments defining the position of the given vertex in the lattice. Taking the private case of coloring the image of Figure 2 will be obtained.

Algorithm 1 The parallel implementation of the toppling procedure in the Abelian sandpile model.

```
 1: while $\exists i \in V$, such that $h_i > 4$ do
 2:     for all $k \in$ set of sublattices do
 3:         for all $i \in k$-th sublattice parallel do
 4:             while $h_i > 4$ do
 5:                 $h_i \leftarrow h_i - 4$
 6:                 for all $j \in adj[i]$ do
 7:                     $h_j \leftarrow h_j + 1$
 8:                 end for
 9:             end while
10:         end for
11:     end for
12: end while
```

Note that the vertices with the same coloring constitute a complete coverage. In case of another sandpile model some other full coverage will be chosen. To create a software system for toppling parallelization, OpenMP and CUDA technologies have been chosen.

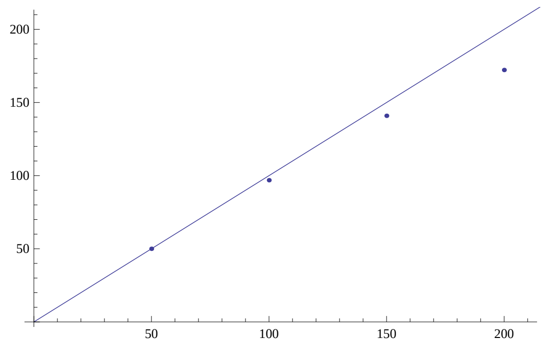

Figure 2. 200.000 grains dropped on 200 × 200 grid

OpenMP is an API that supports multi-platform shared memory multiprocessing programming in C, C++, etc. OpenMP uses a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications for platforms ranging from the standard desktop computer to the supercomputer.

CUDA is a parallel computing platform and programming model that enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU).

k-th sublattice is an array containing all the vertices with ID k.

3. TEST RESULTS

Let us present the test results. The technology used CUDA tested on GPU of the model Dell T5500 Workstation with NVIDIA Tesla C1060.

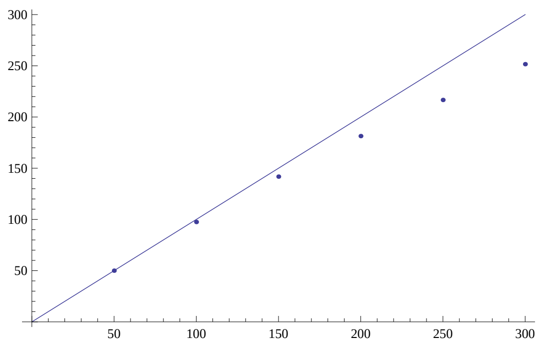

Figure 3. 450.000 grains dropped on 300 × 300 grid

The technology used OpenMP implemented on Szeged supercomputer with 48 cores. The graph is a dependence of acseleration compared with one core.

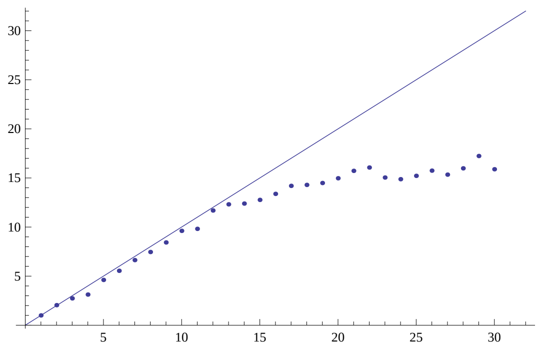

Figure 4. 1.000.000 grains dropped on 500 × 500 grid

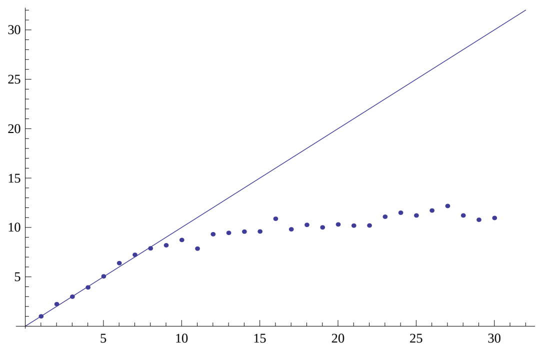

Figure 5. 4.000.000 grains dropped on 1000 × 1000 grid

As it is evident from the graphs depending on lattice sizes the test time is linearly reducing along with the core increase until the moment when the core increase does not play any role. Thus, depending on the lattice size one can choose the best number of cores and get the best time; moreover, knowing the number of cores set free one can run several tests simultaneously and optimize the time.

REFERENCES

[1] D. Dhar. Self-organized critical state of sandpile automaton models. Phys. Rev. Lett., 64:1613,1990

[2] D. Dhar, 2006 Physica A 369, 29

[3] P. Bak, C. Tang, and K. Wiesenfeld, Phys. Rev. Lett. 59, 381 (1987).

[4] P. Grassberger and S.S. Manna, J. Phys. France 51 (1990) 1077-1098

[5] Su.S. Poghosyan, V.S. Poghosyan, V.B. Priezzhev and P. Ruelle, Phys. Rev. E 84, 066119 (2011)

[6] A. Fey, L. Levine, and D.B. Wilson, Phys. Rev. Lett. 104, 145703 (2010); Phys. Rev. E 82, 031121 (2010)

[7] V.S. Poghosyan, S.Y. Grigorev, V.B. Priezzhev and P. Ruelle, 2010 J. Stat. Mech. P07025

[8] V.S. Poghosyan and V.B. Priezzhev, Acta Polytechnica Vol. 51 No. 1/2011

[9] S.N. Majumdar and D. Dhar, 1991 J. Phys. A: Math. Gen. 24 L357

[10] S. Frehmel, ACRI, volume 6350 of Lecture Notes in Computer Science, 35-45. Springer, (2010)