Mathematical Problems of Computer Science 49, 41–48, 2018.

Dynamic Task Scheduling Based on Abelian Sandpile and Rotor-Router Models

Hayk E. Nahapetyan and Suren S. Poghosyan

Institute for Informatics and Automation Problems of NAS RA
e-mail: hayknahapetyan@yahoo.com, psuren55@yandex.ru

Abstract

This study is dedicated to the possible usage of self-organized criticality models in large-scale computing systems for load balancing and energy-awareness. Methods and software tools aimed at modeling and visualization of dynamic tasks scheduling in virtual distributed systems constructed over sandpile and rotor-router models, are also presented.

Keywords: ASM, Rotor-Router decentralized systems, Dynamic task scheduling.

1. Introduction

The concept of self-organized criticality was first introduced by Bak, Tang and Wiesenfeld in 1987 [3], and gave 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. The Sandpile models, being a class of cellular automata, are among the simplest theoretical models, which exhibit self-organized criticality. A special subclass of interest consists of so called Abelian sandpile models (ASM). The Abelian sandpile and rotor-router models were discovered several times by researchers in different communities operating independently. The Abelian sandpile model was invented by Dhar [1], where the rotor-router model, a deterministic analogue of random walk, was first defined by Priezzhev et al. under the name of Eulerian walkers [2]. The Abelian property means that the final stable state of the CA is independent of the order in which the updates of cells are carried out. This property plays a key role during the numerical, as well as analytical studies of the ASM [4] – [7].

There are a number of solutions for tasks scheduling and load balancing based on the Sandpile model [8]–[10]. Anyway, for large-scale real-time computing systems, critical performance constraints are imposed by the environment, and the correctness depends not only on the logical result of the computation, but also on the time at which the results are produced with keeping energy-awareness. Newer solutions based on rotor-router model may provide a better solution in the area. Besides, the paper presents appropriate software packages for simulating and verifying large-scale cluster systems relied on sandpile and rotor-router-based tasks scheduling.

The scheduler and load balancer possessing the above characteristics are constructed on an agent system in which the agents render the cells of a cellular automaton. Thus, agents

are essential components of the architecture, where the topology for interconnecting and where collaborating the agents is another issue for consideration. As a model for simulation, a 2-dimensional lattice is investigated, where every agent denotes itself as a computer node with a private computing resource. Depended on the assigned workload, the node itself makes a decision whether to migrate tasks to adjacent/neighboring nodes or not. Detailed description of the proposed model is given in the third section.

2. Sandpile Model

Consider an undirected graph $G = (V, E)$ described with the set of vertices $V = \{v_1, v_2, \ldots, v_N\}$ and the set of edges E. Each vertex $v_i \in V$ is assigned a variable h_i which takes integer values and represents the height of the sand at that vertex. h_i^{max} denotes the maximal allowed height for the vertex v_i in the graph G. For a d-dimensional lattice, we take $h_i^{max} = 2d + 1$. C_T denotes the set of heights h_i, which determines the configuration of the system at a given discrete time T. A configuration is called stable, if all heights satisfy $h_i < h_i^{max}$. The vertex v_i is called closed, if $h_i^{max} = deg(v_i)$, where $deg(v_i)$ indicates the degree of v_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 to a random vertex $v_i \in V$ by setting h_i to $h_i + 1$ (we assume that the vertex is chosen randomly with a uniform distribution on the set V). This new configuration, if stable, defines C_{T+1}. If $h_i \geq h_i^{max}$, then the v_i becomes unstable and topples losing h_i^{max} grains of sand, while all neighbors of v_i receive one grain. Note that if the vertex is open, then the system loses grains. During the toppling of the closed vertices, the number of grains is conserved. Note also that toppling of a vertex may cause some of its neighboring vertices to become unstable. In this case, those vertices also topple according to the same toppling rule. Once all unstable vertices are 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 a finite number of topplings. Moreover, the new stable configuration is independent of the toppling order. Let $\hat{a}_i$ be an operator, which acts on sandpile configurations and adds a grain to vertex i. It can be easily 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.

2.1 Rotor-Router Model

To define the rotor-router model on a directed graph G, for each vertex of G, fix a cyclic ordering of the outgoing edges. To each vertex V we associate a rotor (V) chosen from among the outgoing edges from V. A chip performs a walk on G according to the rotor-router rule: if the chip is at V, we first increment the rotor (V) to its successor $e = (v, w)$ in the cyclic ordering of outgoing edges from v, and then route the chip along e to w. If the chip ever reaches a sink, i.e., a vertex of G with no outgoing edges, the chip will stop there; otherwise, the chip continues walking forever.

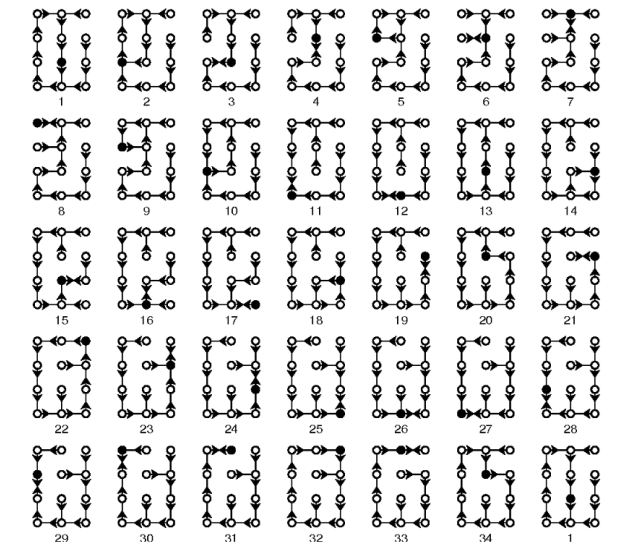

Fig 1. Rotor-Router example.

3. Sand-Scheduler

In this section, we are going to describe a software tool that has a purpose of modeling and visualizing dynamic tasks scheduling in distributed systems. As already mentioned, the scheduling algorithm used by this tool is based on two well-known models: Abelian SandPile model (ASM) and Rotor-Router model.

In original formulation of ASM, each site on a finite grid has an associated value that corresponds to the slope of the pile. This slope builds up as "grains of sand" (or "chips") are randomly placed onto the pile, until the slope exceeds a specific threshold value at which time that site collapses and transfers its sand grains to its adjacent sites, increasing their slope. Bak, Tang, and Wiesenfeld considered the process of successive random placement of sand grains on the grid; each such placement of sand at a particular site may have no effect,or it may cause avalanches, which may have a cascading effect on many sites. The original interest behind the model stemmed from the fact that in simulations on lattices, it is attracted to its critical state, at which point the correlation length of the system and the correlation time of the system go to infinity, without any fine tuning of a system parameter. In the sandpile model dropping another grain of sand onto the pile may cause nothing to happen, or it may cause the entire pile to collapse in a massive slide. We use the above mentioned property of sandpile in order to dynamically schedule tasks, based on the background process of avalanches that is visible in Debug enabled state.

In our workload, tasks may arrive in a group of up to 7 tasks and be assigned to some node in the system. These tasks in a group are typically a set of multiple instances of the same sequential program. That is why the tasks in a group are independent of each other and can be executed in parallel. All the tasks in a group have only one important property, assigned by t_i, which shows the number of rounds needed for the task to be executed in a system. All the tasks of a given group have the same t_i. Preliminaries for the sandpile-based dynamic scheduling problem are the following:

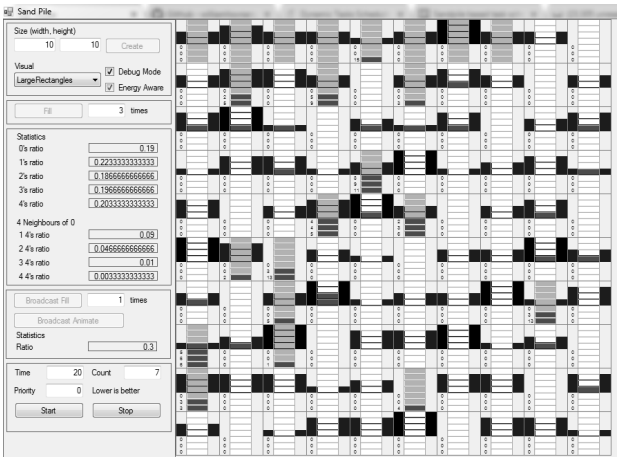

Fig 2. Sand-Scheduler "Debug" enabled.

• Each task has its own required execution time t_i.

• There is a set P of homogeneous processors, where $|P| = nxm$ (10x10 in our example).

• Each processor can execute at most k task simultaneously at any given time. ($k = 3$ in our example)

• The nodes are connected between them and only interacting with the small subset of the neighbours (at most 4).

• Each node has a working queue $Q, |Q| \leq 4$ that gets filled up when all the resources are taken.

• The total execution time for any task is said to be $T_i = t_i + s_i$, where s_i is the time required for the task to be scheduled(find empty slot in nodes) and start its execution.

3.1 Sandpile-Based Mode

In this mode, we study the system in a critical state, and the state of the system is being reconfigured periodically. It is a cellular automaton, which models the process of dropping on grains of sand on a surface and the collapsing of grains due to the increase of the height of the slope. This process is going on regardless of the number of assigned tasks to the system. The tasks are pretty similar to the grains of sand but they do not participate in avalanches themselves. When the grains of the current node reach the maximum value, they start to topple. In case there are tasks assigned to the node at that moment and these tasks are not yet ready to be executed (all the computing resources of the node are reserved), they will be toppled with the sands and move to another node along with the corresponding grain of sand. The transition rule in this model is triggered when the height of the current grain is bigger than the configured value for the system (4 in our study). This process of avalanches is going on indefinitely, meanwhile, this system is dynamically balancing the distribution of tasks among all buckets.

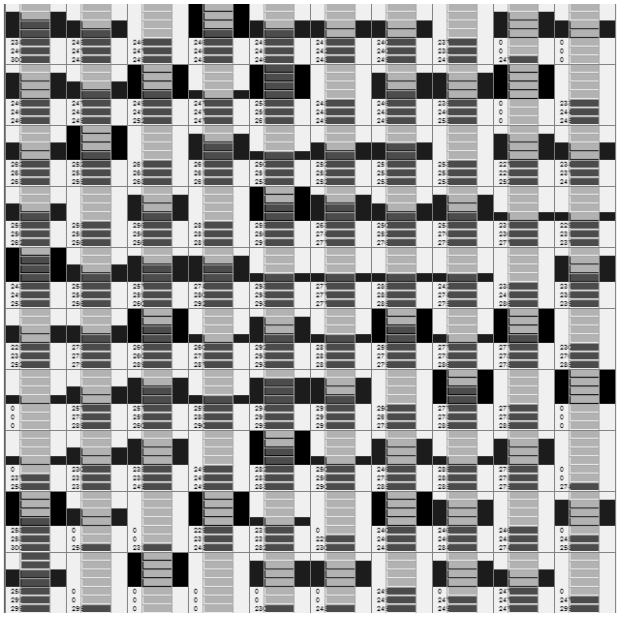

Fig 3. Sand-Scheduler Debug mode. Green - executing tasks, Red - waiting tasks, Blue - fictive sands for simulating avalanches, Black - fictive sands that are in critical state.

Another option that this scheduler supports, is the fault-tolerance. During the execution, we can disable an arbitrary node or nodes, meanwhile, the scheduler will keep working without this kind of failures that are common in large-scale computational systems. The "Energy Aware mode is a modification of this scheduler in order to reduce the number of nodes that are powered on. Depending on the number of non-scheduled tasks in the system at any given moment of time, only the required (min count of nodes needed for executing tasks at the same time) count of nodes are powered on.

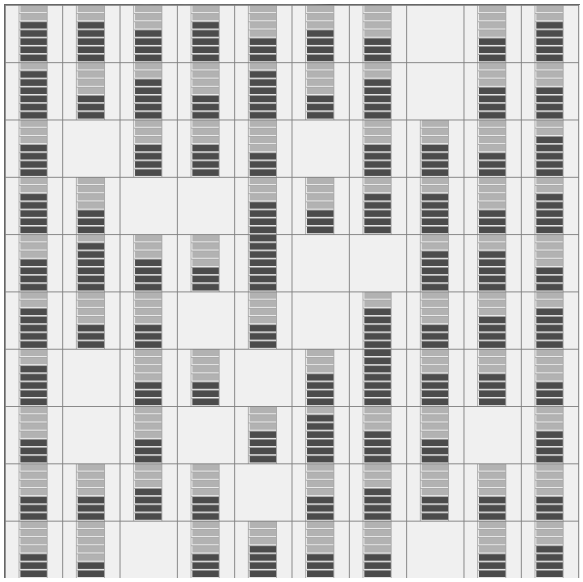

Fig 4. Sand-Scheduler.

3.2 Rotor-Router Mode

The software package developed implements one more scheduling mode based on the Rotor-Router model. Relying on Priezzev's [12, 13] Dhar's studies [14], we can make sure that grains in rotor-router configuration are equally distributed. So, if we change grains with tasks we can be sure that load balancing will be provided in cluster systems. Moreover, we are pushing forward a hypothesis that even for tasks with execution timing(the task will be removed after execution and free up space) in rotor-router configuration tasks will be equally distributed in the system, which is visible via software package described in this paper.

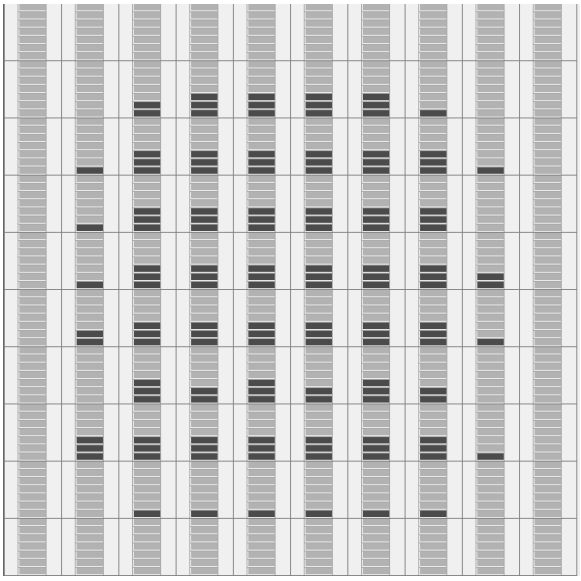

Fig 5.Sand-Scheduler. Non of the tasks is executed yet.

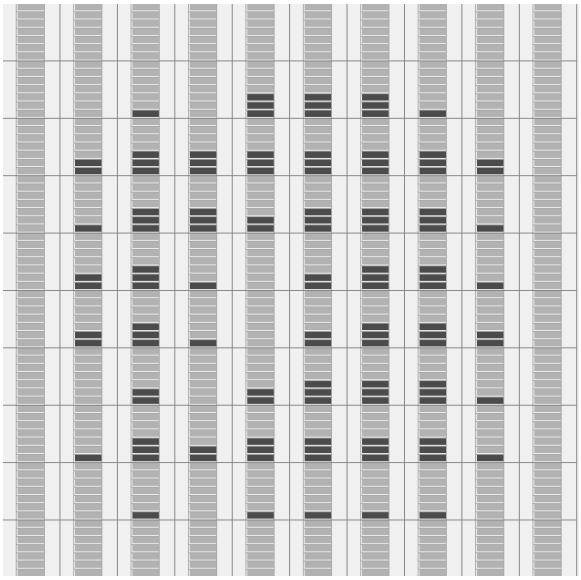

Fig 6. Sand-Scheduler. First tasks have been executed.

4. Conclusion

In this paper, possible usage of ASM and Rotor-Router model in cluster systems has been discussed. Also, appropriate software tools have been developed for cluster simulation and for visualization of tasks dissemination. Perspectives of this work are to deploy ASM and Rotor-Router-based algorithms for task distribution on real systems and obtain a comparative analysis between real-world solutions.

5. Acknowledgement

The authors are grateful to Prof. Yu. H. Shoukourian and Dr. Y. Alaverdyan for important discussions and critical remarks at all stages of the work. This work was supported by the State Committee of Science MES RA, in the frames of the research project No. 16YR-1B008.

References

[1] D. Dhar, "Self-organized critical state of sandpile automaton models", *Phys. Rev. Lett.*, vol. 64, no. 14, pp. 1613–1616, 1990.

[2] B. Priezzhev, D. Dhar, A. Dhar and S. Krishnamurthy, "Eulerian walkers as a model of self-organized criticality", *Phys. Rev. Lett.*, vol. 77, pp. 50795082, 1996.

[3] P. Bak, C. Tang and K. Wiesenfeld,"Self-organized criticality: An explanation of the 1/f noise",*Phys. Rev. Lett.,* vol.59, no. 4, pp. 381384, 1987.

[4] V. S. Poghosyan, S. Y. Grigorev, V. B. Priezzhev and P. Ruelle, "Pair correlations in the sandpile model: A check of logarithmic conformal field theory", *Phys. Lett. B,* vol. 659, pp. 768772, 2008.

[5] Su. S. Poghosyan, V. S. Poghosyan, V. B. Priezzhev and P. Ruelle, "Numerical study of correspondence between the dissipative and fixed-energy Abelian sandpile models", *Phys.Rev. E, 84, 066119*, 2011.

[6] V. S. Poghosyan, S. S. Poghosyan and H. E. Nahapetyan, "The Investigation of models of self-organized systems by parallel programming methods based on the example of an Abelian sandpile model", *Proc. CSIT Conference 2013*, Yerevan Armenia, Sept. 23-27, pp. 260-262, 2013.

[7] H. Nahapetyan, J.-Pierre Jessel, S. Poghosyan and Y. Shoukourian,"A multi user and multi purpose CA simulator",*Phys. Rev. Lett.*, vol.59, no. 4, pp. 381384, 1987. Proc. CSIT Conference 2017, Yerevan Armenia, Sept. 23-27, pp. 260-262.

[8] Y. Rabani, A. Sinclair, and R. Wanka, "Local divergence of markov chains and the analysis of iterative load-balancing schemes", *In IEEE Symp. on Foundations of Computer Science*, pp. 694705, 1998.

[9] J. L. J. Laredo, P. Bouvry, F. Guinand, B. Dorronsoro and C. Fernandes, "The sandpile scheduler", *Cluster Computing* vol.17, pp 191204, 2014.

[10] J. Gsior and F. Seredyski, "A Sandpile cellular automata-based scheduler and load balancer", *Journal of Computational Science*, vol.21, pp. 460-468, 2017.

[11] L. Levine and Y. Peres, "Asymptotics for rotor-router aggregation and the divisible sandpile", *Potential Analysis*, 30: 1. https://doi.org/10.1007/s11118-008-9104-6

[12] A. M. Povolotsky, V. B. Priezzhev and R. R. Shcherbakov, "Dynamics of Eulerian walkers", *Physical review E*, vol.58, DOI:https://doi.org/10.1103/PhysRevE.58.5449

[13] V. B. Priezzhev, "Self-organized criticality in self-directing walks", arXiv:cond-mat/9605094

[14] D. Dhar, "Theoretical studies of self-organized criticality", *Physica A: Statistical Mechanics and its Applications*, vol. 369, no. 1, pp. 29-70, 2006

Submitted 04.09.2017, accepted 15.01.2018.

Դինամիկ առաջադրանքների պլանավորում՝ հիմնված Աբելյան ավազակույտի և Rotor-Router մոդելների վրա

Հ. Նահապետյան և Ս. Պողոսյան

Ամփոփում

Այս ուսումնասիրությունը նվիրված է խոշորածավալ հաշվողական համակարգերում ինքնակազմակերպ կրիտիկական մոդելների հնարավոր օգտագործմանը ծանրաբեռնվածության բաշխման և էներգախնայողության նպատակներով: Ներկայացված են մեթոդներ և ծրագրային գործիքներ, որոնք միտված են ավազակույտի և ռոտոր-ռոուտեր մոդելների հիման վրա կառուցված վիրտուալ բաշխված համակարգերում դինամիկ խնդիրների պլանավորմանը և տեսաբերմանը:

Динамическое планирование задач, основанных на моделях абелевой песчаной кучи и ротор-роутера

Г. Нагапетян и С. Погосяан

Аннотация

Исследование посвящено возможному использованию самоорганизующихся критичных моделей в широкомасштабных системах для балансировки нагрузки и энергосбережения. Также представлены методы и программные средства, предназначенные для моделирования и визуализации планирования динамических задач в виртуальных распределенных системах, построенных на моделях песчаной кучи и ротор-роутера.