Load Balancing at the Edge of Chaos: How Self-Organized Criticality Can Lead to Energy-Efficient Computing

Juan Luis Jiménez Laredo, Frédéric Guinand, Damien Olivier, and Pascal Bouvry

Abstract—This paper investigates a self-organized critical approach for dynamically load-balancing computational workloads. The proposed model is based on the Bak-Tang-Wiesenfeld sandpile: a cellular automaton that works in a critical regime at the edge of chaos. In analogy to grains of sand, tasks arrive and pile up on the different processing elements or *sites* of the system. When a pile exceeds a certain threshold, it collapses and initiates an avalanche of migrating tasks, i.e. producing load-balancing. We show that the frequency of such avalanches is in power-law relation with their sizes, a scale-invariant fingerprint of self-organized criticality that emerges without any tuning of parameters. Such an emergent pattern has organic properties such as the self-organization of tasks into resources or the self-optimization of the computing performance. The conducted experimentation also reveals that the system has a critical attractor in the point in which the arrival rate of tasks equals the processing power of the system. Taking advantage of this fact, we hypothesize that the processing elements can be turned on and off depending on the state of the workload as to maximize the utilization of resources. An interesting side effect is that the overall energy consumption of the system is minimized without compromising the quality of service.

Index Terms—Energy efficiency, Nonlinear dynamical systems, Distributed computing, Scheduling algorithms.

1 Introduction

Energy efficiency is nowadays one of the main areas of research in Computer Science. With the advent of flourishing technologies such as cloud computing, power demands in large-scale computing infrastructures have exploded and the energy consumption has turned into a limiting factor to a further development of the industry: in 2010, volume servers and data centers worldwide [1] consumed approx. 270 billions of KWh, enough to cover the electricity demands of entire countries such as Australia or Spain [2]. Part of the problem is in current technology which has not yet devised the ways to maintaining the energy consumption proportional to the volume of services provided. However, researchers are embarked in this task: energy awareness has become a major technology driver in the design of large-scale distributed systems.

The case of proportional computing [3] has already opened frontiers towards more energy-efficient computing. The approach relies on the idea that the energy consumed by a server should be kept proportional to its utilization. In this line, current devices are implementing power management techniques, such as dynamic scaling [4], to modulate the voltage and frequency of CPUs according to the state of the workloads. Nonetheless, power management is not the final solution to energy efficiency but a mere mechanism that requires *a higher-level control* to save energy; the same way that light switches need to be operated by a human (smart entity) to turn lights on and off.

Brown and Reams [5] state that a control mechanism of this kind is best approached as an optimization problem where *"a system consumes the minimum amount of energy required to perform any task"*. The importance of such a statement is that it allows to formally describe energy efficiency in terms of a scheduling problem, including specific –and potentially conflicting– objective functions such as energy consumption, resource utilization or quality of service (QoS), i.e. energy-efficient computing can be formulated as a multi-objective scheduling problem [6].

Alongside the difficulty of tackling several objectives, this type of problem is particularly challenging because of the high-dimensionality that it may involve, e.g. exascale systems can reach millions of nodes and billions of tasks [7], [8]. That implies analytical intractability of real solutions as the curse of dimensionality [9] prevents to solve high-dimensional instances to optimality. Similarly, meta-heuristics find extreme difficulties to track down high-dimensional solutions in a reasonable time; especially because granting convergence requires to iterate over globally computed solutions (see e.g. [6], [10], [11]). Achieving energy efficiency is therefore calling for further investigations beyond the state of the art. In this scenario, *complex systems* may hold the key to describing large-scale dynamics of energy-efficient distributed systems.

This paper aims at exploring these potentials in the context of dynamic load-balancing: a scheduling approach that tackles the problem of reassigning tasks between resources at runtime [12]. We envision a large-scale computational system as a living organism able to self-organize locally

- *J.L.J. Laredo, F. Guinand and D. Olivier are with LITIS, University of Le Havre, Le Havre, France.*
  E-mails: {jimenezj,frederic.guinand,damien.olivier}@univ-lehavre.fr
- *P. Bouvry is with CSC/SnT, University of Luxembourg, Luxembourg.*
  E-mail: pascal.bouvry@uni.lu

Manuscript received April 19, 2005; revised September 17, 2014.

and acquire emergent properties such as maximizing the energy efficiency. Our proposal relies on a simple idea: turning off resources when they do not have any more tasks to process. This simple power management technique is commonly employed in exploratory investigations [13] like the one presented here. However, more interesting is that it naturally drives to a subsequent (and more challenging) question: when should a computing resource, which has been turned off, be turned on again?. Intuitively, if there are too many active resources in a system, the energy consumption will explode. However, if the system turns off too many resources, the QoS will dramatically drop down.

In order to explore such a trade-off, we investigate the properties of self-organized criticality systems [15] when applied to energy-efficient dynamic balancing: an investigation that to the best of our knowledge is the first of its class. In particular, we propose a model based on the Bak-Tang-Wiesenfeld (BTW) sandpile [14]: a cellular automaton that displays self-organized criticality by mimicking the behavior of avalanches of sand. Typically grains of sand are deposited at randomly chosen sites in a 2-dimensional grid lattice, one grain at a time. When the system reaches a certain critical state, small disturbances can lead to avalanches that can propagate spatially and temporally and reconfigure the system. Here, the analogy with dynamic load-balancing is straightforward: the same way that avalanches scatter sand throughout the lattice, they can produce load-balancing when applied to tasks queuing in a distributed system.

With the issue of energy efficiency in mind, this paper builds up starting from the original sandpile model as described by Bak, Tang and Wiesenfeld. The aim is finding an answer to the following research question:

- What is the trade-off between the energy consumed and the QoS delivered by a sandpile distributed system?

Trying to respond to the previous question, the rest of the paper is organized as follows: section 2 reviews some related works in both fields, complex systems and energy-aware load-balancing. Section 3 outlines the problem of scheduling workloads arriving in the form of independent tasks and introduces some metrics to evaluate the performance of computational systems. Section 4 presents an algorithmic description of our load-balancing proposal. The basic dynamics of the approach are then described in section 5. Section 6 introduces a theoretical framework where the sandpile is able to display organic load-balancing. Section 7 analyzes the trade-offs between the energy consumption and the QoS delivered by the system. Finally, conclusions are drawn in section 8 and some future lines of research are presented.

2 Related Works

This section starts with a description of the BTW sandpile model that will be employed later on as the baseline method in our proposal. Then, it continues with a survey of works tackling dynamic load-balancing from both perspectives, complex systems and energy awareness.

2.1 The BTW Sandpile

The theory of self-organized criticality (SOC) [15] offers a well-defined explanation on how the interactions of sparse elements can lead a complex system to a critical point in which the elements percolate and give rise to a global pattern, a so-called *avalanche*. This phenomenon is observed in a myriad of dynamical systems including earthquakes [16], mass extinctions in biological evolution [17] or the branching process of neocortical cirtuits [18], but it was not until Per Bak, Chao Tang and Kurt Wiesenfeld published in 1987 an artificial system called the sandpile [14] that the concept of SOC was postulated. Overall, SOC describes a critical non-equilibrium state of complex systems where the frequency and size of the events follow a power-law relation independently of the scale of the system, i.e. small avalanches occur very often while large catastrophic avalanches happen very rarely. This paper aims at investigating such features in the context of dynamic load-balancing. To that end, we propose a decentralized load-balancing approach based on the original BTW sandpile model.

The sandpile is a cellular automaton in which grains of "sand" are randomly dropped over a grid lattice. Grains accumulate in the different sites until they exceed a certain threshold leading to avalanches. That way, depending on the state of the system, the single drop of a grain may change nothing or start an avalanche that reconfigures the system. In fact, the sandpile is a class IV cellular automaton (Wolfram [19]) in the region between order (classes I and II) and chaos (class III). This kind of cellular automaton can exhibit universality in the sense of being Turing complete, something that led the American scientist Christopher Langton [20] to analyze criticality as a universal machine. Langton shows how the criticality phenomenology supports basic operations of information transmission, storage and modification to perform computation at the edge of chaos.

Fig. 1 exemplifies the basic dynamics of an avalanche in a small sandpile. Initially, adding a grain at a site (x, y) results in a simple modification on the height $h(x, y)$ of the site, such that:

$$h(x, y) \longrightarrow h(x, y) + 1$$

However, the model states that if the height exceeds a certain threshold value (e.g. $h(x, y) \geq 4$) the site collapses and loses a number of grains:

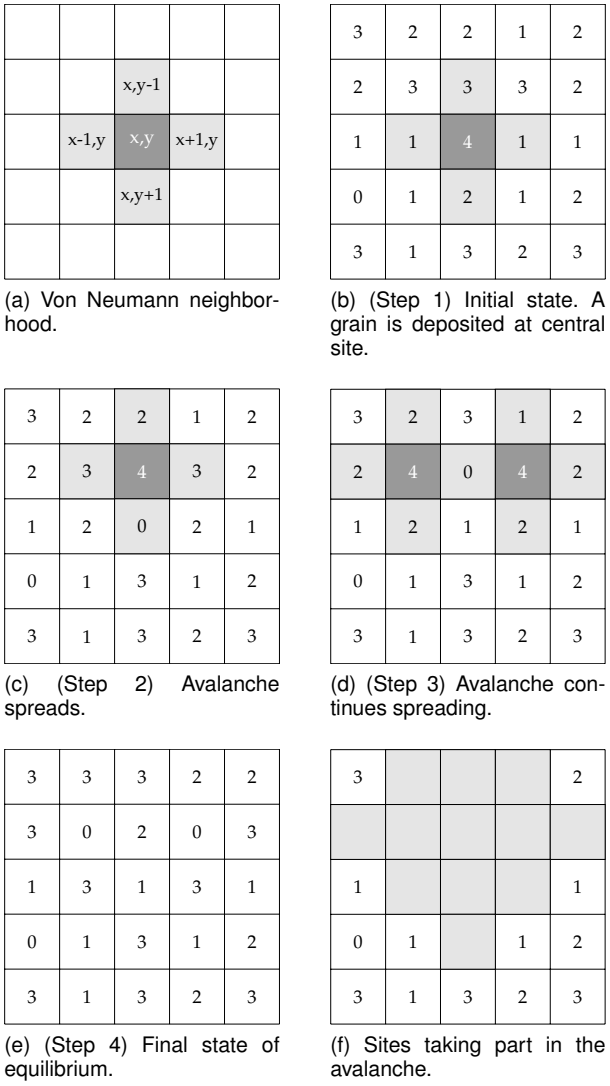
$$h(x, y) \longrightarrow h(x, y) - 4$$

which are then reassigned to its neighbors. If we consider a von Neumann neighborhood, such a reassignment will modify the state of the neighbors in the following way:

$$h(x \pm 1, y) \longrightarrow h(x \pm 1, y) + 1$$
$$h(x, y \pm 1) \longrightarrow h(x, y \pm 1) + 1$$

This starting avalanche can propagate if the respective neighbors also exceed the threshold as a consequence of the new reassignment. Therefore, and despite the simplicity of the rules, avalanches of all sizes may take place. In fact, as a SOC system, the sandpile evolves in the long term into a non-equilibrium critical state where the frequency

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2016.2582160, IEEE Transactions on Parallel and Distributed Systems



(a) Von Neumann neighborhood.

(b) (Step 1) Initial state. A grain is deposited at central site.

(c) (Step 2) Avalanche spreads.

(d) (Step 3) Avalanche continues spreading.

(e) (Step 4) Final state of equilibrium.

(f) Sites taking part in the avalanche.

Fig. 1. Example of sandpile dynamics in a small cellular automaton. Sites are arranged in von Neumann neighborhoods as depicted in (a). From left to right and from top to bottom, (b) to (e) show the propagation of an avalanche. The sites involved in the avalanche are colored in gray in (f).

of avalanches keeps a power-law proportion to their sizes. Intuitively such a behavior has a load-balancing effect on the different heights of the sites that we aim at adapting to the problem of dynamic load-balancing computational workloads.

2.2 Dynamic Load-balancing and Energy Awareness

Complex systems have been devised for a long time as an effective mechanism for dynamic load-balancing. In the early 90s, Willebeck-Lemair and Reeves [21] proposed two dynamic load-balancing methods based on diffusion and gradient models. Both methods follow the principle of decentralization allowing local interactions between neighboring processors: in the diffusion approach the workload spreads from overloaded to underloaded processors and, in the gradient approach, the underloaded resources request for tasks. Despite such differences, the main goal of the work was to provide new scalable means to support large-scale distributed computing since (as the authors pointed out) "*only a few strategies have been designed, or are scalable, to support highly parallel multi-computer systems*", a problem that still persists nowadays.

In the line of diffusion models, Jelasity et al. [22] propose a dynamic load-balancing approach based on peer-to-peer systems. Rather than a fully-operative scheduling system, the authors aim at illustrating the application potentials of gossiping protocols. They describe the problem of dynamic load-balancing as an equivalent to the problem of averaging a set of distributed numbers using decentralized *aggregation*. Despite its simplicity, this work is an inspiring reading on the potentialities of complex networks in distributed load-balancing. In this line, Salman et al. [23] analyze the impact of several interconnection topologies in distinct load-balancing strategies concluding that, independently of the strategy, the structure of the network has a significant impact on the performance of the system.

Inspired by the dynamics of liquids, Hu and Klefstad [24] propose a dynamic load-balancing approach which has a strong connection with the one presented here. Both methods are based on self-organization and both try to mimic the way gravity has to flatten different elements. However, there is a key difference concerning the critical state (i.e. the "C" in SOC). While our approach behaves at the border of chaos, the liquid-based approach only recognizes two states: either the system is in equilibrium, which means that the workload is evenly balanced, or the system is in a non-equilibrium state. Therefore, no matter how small a perturbation is, it will always lead to a non-equilibrium state with the subsequent reallocation of tasks.

Although previous works represent a step forward in the understanding of how a complex system may lead to dynamic load-balancing, none of them tackles the issue of energy efficiency. In fact, and to the best of our knowledge, little is known about whether energy efficiency can emerge bottom up from decentralized processes based on local interacting rules, something that we aim at analyzing in this paper. Up to now, the simultaneous optimization of the energy and performance of the system has been mainly approached from the perspective of problem solving, where global (centralized) optimization methodologies have been coupled with power-management techniques.

Pinheiro et al. [13] propose to balance/unbalance clusters of workstations by dynamically turning nodes on and off[1]. The authors propose a centralized algorithm in which decisions are made by monitoring the workload and performance of the system. Although the study only involves 8 nodes, the work succeeds in describing the potentials of simple power-management mechanisms in the performance vs. energy trade-off: the energy consumption is reduced by a 43% while the performance degradation is kept below 20%.

Khan and Ahmad [25] were the first in applying game theory to the problem of optimizing both objectives: energy consumption and system performance. The problem is modelled as a multi-objective extension of the generalized assignment problem, where the authors are able to prove

1. Like in this paper, the simplest assumption that can be made on power management is that of a system with two possible states in which nodes can be powered on and off.

the Pareto optimality of an optimization approach based on the Nash bargaining solution.

In recent works [6], [10], [11], researchers have found a powerful ally in meta-heuristics such as genetic algorithms or tabu search. This kind of optimization algorithms are widely employed in multi-objective optimization, where they have proven a good compromise between the quality of the solutions and the time of convergence. However, meta-heuristics can only afford instances of relatively low dimensionality, e.g.: the largest instance in all previous works involves 256 machines and 4096 tasks. While this may suffice to optimizing specific types of workloads such as DAGs (directed acyclic graphs), it also prevents the applicability of meta-heuristics to larger systems with exascale dimensions.

In [26], [27], we envisioned the sandpile as a dynamic load-balancing mechanism for the first time. Experiments were conducted for very large problems with up to 2048 heterogeneous computing nodes and more than 2 million tasks, showing that the approach was able to achieve near-optimal solutions in terms of makespan, throughput and flowtime. In order to achieve such results the canonical BTW sandpile model was extended with these three new features:

- The transition rule (the one leading to avalanches), was only triggered if the height of a given pile was larger than the accumulated heights of two neighbors, i.e. establishing a gradient.
- The interconnection topology was designed as a small-world graph. The advantage of using such a topology is that "*the system releases the potential of building up catastrophic avalanches more easily and produces fewer catastrophic avalanches*" [28].
- Avalanches were implemented virtually using a gossiping protocol, i.e. nodes negotiate a state of equilibrium before the real migration of tasks takes place.

Our current proposal is inspired by previous results and aims at investigating the performance of the sandpile when the energetic efficiency of the model is also taken into account. Unlike in our previous works, the BTW sandpile is implemented as it is to maintain the self-organized criticality features of the model unaltered. This paper is therefore a proof-of-concept on the energetic efficiency of the BTW sandpile without further artefacts. In other words, the sandpile is not conceived as the pilot of an efficient load-balancer (e.g. trying to minimize the communications of the model) but rather as a mean to conduct an exploratory investigation on SOC systems in energy-efficient load balancing. In future works, all previous improvements will be incrementally tested from which we may elucidate optimal settings for the model.

3 Scheduling Independent Tasks

In Computer Architecture, scheduling is the problem of assigning tasks to resources. Despite simple in its formulation, scheduling is a multi-faceted problem that may involve many possible objectives and constraints, and has been proven a high-dimensional NP-hard problem [29].

This paper aims to be a proof-of-concept on the potentialities of the sandpile as an effective load-balancing mechanism. To that end, the problem is intentionally simplified and assumes certain conditions such as a negligible cost of communications or the homogeneity on the workload and computing architecture. The interested reader may refer to [26] for a more realistic characterization that includes heterogeneous settings and a model for communications. In general, the problem of scheduling independent tasks has two main components:

- A **workload** consisting of a set $N = \{\langle n_1, a_1\rangle \ldots, \langle n_v, a_v\rangle\}$ of v independent tasks, such that:
  - n_i is the length in terms of computing instructions of the i^{th} task in N.
  - a_i is the respective arrival time.
- A **computing architecture** is a set $P = \{p_1, \ldots, p_q\}$ of q interconnected processors able to process tasks with a given speed. At this early stage of the research, we propose a simple architecture that consists of:
  - *Resources:* Let us denote p_i as the i^{th} processor in the architecture P. We assume that every processor p_i has a homogeneous speed $p_i = 1\frac{instruction}{cycle}$ and can be turned off (p_i^-) and on (p_i^+).
  - *A topology:* The q processors are arranged in a grid lattice such that $p_{i,j}$ refers to the processor at position (i, j) in a square matrix:

$$P = \begin{pmatrix} p_{1,1} & \cdots & p_{1,\sqrt{q}} \\ \vdots & \ddots & \vdots \\ p_{\sqrt{q},1} & \cdots & p_{\sqrt{q},\sqrt{q}} \end{pmatrix}.$$

3.1 Metrics

In order to assess our load-balancing approach, we consider a representative set of metrics taking into account different aspects of the problem such as the efficient utilization of the infrastructure or the QoS delivered. The proposed metrics can be analyzed separately although they represent together a set of conflicting objectives, e.g.: increasing the size of a computing infrastructure may enhance some QoS criteria but will certainly increase the overall energetic requirements of the system. All in all, the aim is finding the schedule that maximizes the utilization of a computing infrastructure while guaranteeing a certain QoS to users.

3.1.1 Flowtime as QoS measure

The flowtime is the average waiting time of an item in a system [30]. Therefore, it represents a good QoS indicator as it relates to the waiting time of users: the shorter the flowtime, the better the user satisfaction.

Let us define W as a $v \times q$ matrix where $w_{i,j} \in W$ is either zero or the global timestamp for the i^{th} task at the moment of being fetched for non-preemptive execution in a processor $p_j \in P$. Hence, the flowtime of a task n_i in a system P can be defined as:

$$f(i, P) = n_i + \max_{\forall j \in [1,\ldots,q]}(w_{i,j}) - a_i \quad (1)$$

where a_i is the arrival time of the i^{th} task in the system. In general, we may also talk about the average flowtime as:

$$\overline{F}(N, P) = \frac{\sum_{i=1}^{v} f(i, P)}{v} \quad (2)$$

3.1.2 Resource utilization

In addition to minimizing the flowtime, we also seek to maximize the utilization of resources: a metric that relates resources with the time they are occupied doing some effective job. Hence, this metric is defined in the subset of active resources $P^+ \subseteq P : \{P^+ \cap P^- = \emptyset \wedge P^+ \cup P^- = P\}$. At a given time t the utilization of a resource $p_j^+ \in P^+$ can be expressed as:

$$u(p_j^+, t) = \begin{cases} 1 & if \quad \neg idle(p_j^+, t) \\ 0 & otherwise \end{cases} \tag{3}$$

where $idle(p_j^+, t)$ is a function indicating whether p_j^+ is idle at time t or busy. For the entire system P^+, the utilization of resources can be defined as:

$$\overline{U}(P^+, T) = \frac{\sum_{t=1}^{T} \sum_{j=1}^{q^+} u(p_j^+, t)}{Tq^+} \tag{4}$$

which provides an averaged value for active resources composing P^+ ($q^+ = |P^+|$) during a period of time T.

3.1.3 Energy consumption

To assess the energetic efficiency of our proposal, we propose a simple model where the energetic consumption of the system is in linear proportion to the number of active resources (see [31] for more realistic energy models). In a period of time T the energy consumption of a system P is:

$$E(P, T) = \sum_{t=1}^{T} \sum_{j=1}^{q} e(p_j, t) \tag{5}$$

where:

$$e(p_j, t) = \begin{cases} 1 & if \quad p_j \in P^+ \\ 0 & if \quad p_j \in P^- \end{cases} \tag{6}$$

After this part that describes the problem of scheduling independent tasks, we now focus on our proposed method for dynamic load-balancing.

4 Model Description

This section presents a decentralized approach for dynamically load-balancing workloads in distributed systems. The proposed model is based on the BTW sandpile (see section 2.1), from which we expect it inherits the features of SOC systems, i.e. scale-invariance of the critical point, and robustness to the tuning of parameters. Overall, there is a main difference between the canonical BTW sandpile and our extended version for dynamic load-balancing. In analogy to grains of sand, tasks also arrive and pile up in the different sites of a grid lattice. However, unlike in the original sandpile, every site is a processing element that can consume tasks with a certain speed. Therefore, the analogy can be pushed forward by thinking about a sieve, with grains piling up but also slipping through the gaps as depicted in Fig. 2.

We find that these new features are comprised in the functionality of a double-ended queue (dequeue) that we propose here as an efficient data structure to implement the decentralized algorithm. Table 1 presents a reduce set of functions to that end:

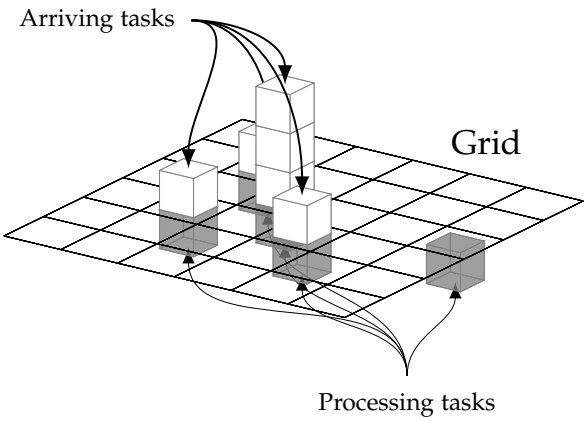


Fig. 2. Sandpile in a sieve.

- The *neighbors()* function returns a list with all the processing elements in the neighborhood[2] of a given resource $p_{x,y}$ which are arranged in a toroidal grid:

  $$p_{x,y}.neighbors() = \{p_{i,j} : |i - x|_{\sqrt{q}} + |j - y|_{\sqrt{q}} \leq r\}$$

  where r is the range of a neighborhood as depicted in Fig. 3. Hence, for a given r, the function returns a list with $2r(r+1)$ neighbors.
- The *push* and *pop* functions act at the back of the queue so that task migrations can be implemented using both. Note that, while *push (m)* inserts m tasks to the back of a queue, *pop (m)* removes $m \times 2r(r+1)$ tasks, i.e. m tasks for each of the $2r(r+1)$ neighbors.
- The h function monitors the state of the queue and returns its height.
- The *idle* function checks the flags of the processing element to determine whether its current status is (*idle*) or (*busy*).
- Finally, the *shift* function acts at the front of the queue and is the only way for the processor to retrieve tasks; this function assumes a non-preemptive operational mode for the sake of simplicity.

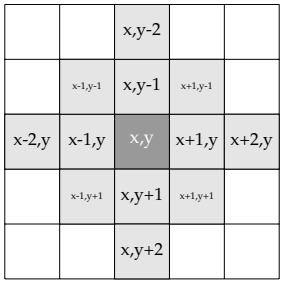


Fig. 3. Von Neumann neighborhood of range $r = 2$. The neighbors of a site (x, y) are those ones contained within a manhattan distance $\leq r$. Hence, the size of a neighborhood is $2r(r+1)$.

2. For the sake of simplicity, this study focuses on von Neumann neighborhoods although other types of neighborhoods, such as Moore, should also be applicable without any loss of generality.

Function	Description	Pre-/Postconditions
neighbors()	Returns the $2r(r+1)$ adjacent queues	r is the von Neumann range
push(i)	Inserts a task with index i at the back	The task i is fetched at time a_i
push(m)	Inserts m tasks at the back	m is a list of tasks
pop(m)	Removes $m \times 2r(r+1)$ tasks at the back	Returns a list with the removed tasks
h()	Counts the no. of tasks in the queue	Returns the size/height of the pile
idle()	Verifies that the current processor is not busy	Returns true if the processor is idle
shift()	Removes/processes one task at the front of the queue	Returns the removed element

TABLE 1
Double-ended queue (dequeue) functions at a given site $p_{x,y}$. Please, note that the function *Push* is a polymorphic function which can accept two different types (i = the index of a single task) or (m = a list of tasks).

Algorithm 1 PSEUDO-CODE OF THE SANDPILE MODEL

1: **for all** $N = \{\langle n_1, a_1\rangle, \ldots, \langle n_v, a_v\rangle\}$ tasks **do**

 # Task arrival:

2: Choose randomly a processor $p_{x,y} \in P$
3: $p_{x,y}.push(i)$ ▷ **drop the *i-th* task in** $p_{x,y}$ **at time** a_i

 # Load-balancing phase:

4: **while** $\exists p_{i,j} : p_{i,j}.h() \geq h_c$ **do**
5: $p_{i,j}.pop(m)$
6: **for all** $p_{i',j'} \in p_{i,j}.neighbors()$ **do**
7: $p_{i',j'}.push(m)$
8: **end for**
9: **end while**

 # Processing phase:

10: **for all** $p_{i,j} \in P$ **do**
11: **if** $p_{i,j}.idle()$ **then**
12: $p_{i,j}.shift()$
13: **end if**
14: **end for**

15: **end for**

4.1 Algorithmic description

Using the previous set of functions, the sandpile algorithm can be easily built according to Algorithm 1. By every site implementing a double-ended queue, the emergent behavior of the system is expected to display SOC properties and act as a decentralized load-balancing system.

The algorithm can be divided into three different phases: task arrival, load-balancing and processing. These phases include all the necessary steps for the processing of all tasks in $N = \{\langle n_1, a_1\rangle, \ldots, \langle n_v, a_v\rangle\}$.

- *Task arrival (lines 1-3)*: Tasks in N are sorted in order of arrival and fetched iteratively in line 1. Every selected task n_i is then assigned to a resource $p_{x,y}$ by calling its queue function $p_{x,y}.push(i)$. To reproduce the canonical sandpile behavior, the resource $p_{x,y}$ is randomly selected from the pool of resources (P).
- *Load-balancing phase (lines 4-9)*: These lines implement the actual functionality of the sandpile. After a new task has been assigned to the queue $p_{x,y}$, the transition rule in line 4 verifies that the height of every queue $p_{i,j}$ is below a critical value h_c. Otherwise, every $p_{i,j}$ violating the transition rule triggers an avalanche by calling $p_{i,j}.pop(m)$:

$$h(i,j) \rightarrow h(i,j) - m \times 2r(r+1)$$

and pushing the tasks into its $2r(r+1)$ neighbors with $p_{i',j'}.push(m)$:

$$h(i',j') \rightarrow h(i',j') + m : \forall i', j' \in p_{i,j}.neighbors()$$

This avalanche will propagate throughout the system until $\forall p_{i,j} : p_{i,j}.h() < h_c$. However, to avoid endless loops, we add a hop counter to every task and set a *hop limit* in 10000 hops, which is reasonably large. If a dequeue detects a task which has reached the hop limit, the threshold condition is ignored and the avalanche is locally cancelled. That implies that we allow some temporal configurations in which there may exist some processors with a load greater than the threshold. Otherwise, every processor $p_{i,j}$ resets to zero the hop counter of all tasks in the dequeue when $p_{i,j}.h() < h_c$.
- *Processing phase (lines 10-14)*: This last phase takes place after the system has found a state of equilibrium in the load-balancing phase. Every site $p_{i,j} \in P$ retrieves one task at the front of the queue ($p_{i,j}.shift()$) iff the processor is not busy and the queue is not empty.

This algorithm establishes the basic structure of our model. However, the dynamics of the system will depend on a number of decision variables that we aim at analyzing in the following sections.

4.2 Parametrization

Recalling from the algorithmic description, there are three basic parameters that will establish the dynamics of the sandpile:

- h_c, the threshold.
- r, the range of the von Neumann neighborhood.
- m, the number of tasks to be migrated to every neighbor.

These parameters are subject to the following constraint:

$$h_c \geq m \times 2r(r+1)$$

which means that the threshold must be equal to (or greater than) the number of tasks to be reallocated. *Proof*. If we

assume $h_c < m \times 2r(r+1)$, a computing node $p_{i,j}$ may attempt to pop (line 5 in Algorithm 1) more tasks than those piling up in its own queue (i.e. $p_{i,j}.h() < m \times 2r(r+1)$) causing an error at runtime. Therefore, $h_c \geq m \times 2r(r+1)$.

Given that the minimum range of a von Neumann neighborhood is $r = 1$ and the minimum number of tasks migrating to neighbors is $m = 1$, h_c must always be ≥ 4. Such a parametrization ($h_c = 4$, $r = 1$ and $m = 1$) is the one employed in the canonical model [14] and will be the default setting in this paper. Table 2 shows some constraints for different settings of r and m.

	$r=1$	$r=2$	$r=3$
$m=1$	$h_c \geq 4$	$h_c \geq 12$	$h_c \geq 24$
$m=2$	$h_c \geq 8$	$h_c \geq 24$	$h_c \geq 48$
$m=3$	$h_c \geq 12$	$h_c \geq 36$	$h_c \geq 72$

TABLE 2
Constraints on h_c for different values of r and m.

5 Basic Dynamics

The dynamics of a load-balancer are associated with the migration of tasks. In short, the migration mechanism tries to minimize the completion time of an incoming workload by maximizing the utilization of the available resources at runtime. However, migrations also generate a certain overhead –due to the limited capacity of physical resources such as the bandwidth– that may undermine the performance of the results. Dynamic load-balancing consists therefore in a trade-off between maximizing the utilization of resources and minimizing the required migrations to that end.

This section proposes simple simulations to analyze the runtime dynamics of the sandpile in the previous terms. The aim is to gain insights into the emergent properties of the load-balancer in the trade-off of maximizing the utilization of resources and minimizing the migration of tasks. We consider the following experimental setup:

- A **computer architecture** P of $q = 100$ resources arranged in a 10×10 toroidal grid topology.
- A **set of workloads** $\{N_1, \ldots, N_{120}\}$, each workload N_i being composed of 10000 tasks $\langle n, a \rangle$ of length (n) and increasing time of arrival (a) such that:

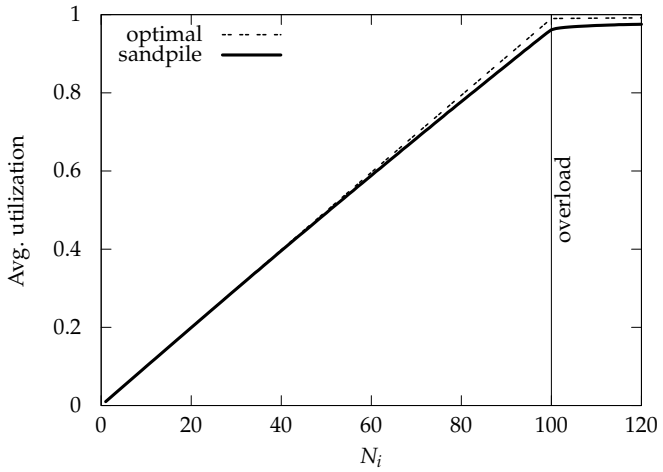
$$N_i = \{\langle n_1, 1\rangle, \ldots, \langle n_{10000}, 10000\rangle\}$$

  where $\forall n_k \in N_i : \{n_k = i \quad instructions\}$, i.e. the length of the tasks in a given workload N_i is homogeneous and depends on the index i.
- The **sandpile** implements default settings: a von Neumann neighborhood $r = 1$, a threshold $h_c = 4$ and a migration rate of one task per neighbor $m = 1$.
- As a counterpart for comparisons, we establish optimal boundaries using a **round-robin (RR)** job assignment policy [32] where tasks are assigned to processors in cyclical order. Note that the optimality condition holds on the homogeneity of the tasks and processing elements.

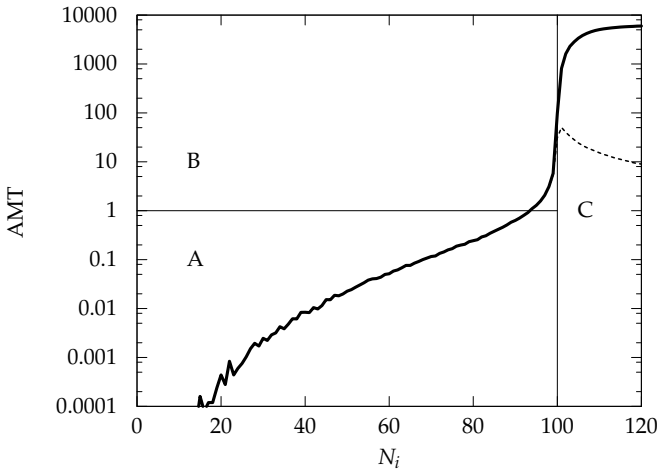
Under these settings, the maximum throughput of P can be reached at $10 \times 10 = 100 \quad instructions/cycle$, which coincides with the incoming ratio of instructions per cycle in N_{100}. Therefore, for workloads $N_{i<100}$ the architecture P will be necessarily under-utilized; while for $N_{i>100}$, P will be necessarily overloaded.

5.1 Phase Transitions and Power Laws

Simulations were conducted in the sandpile simulator[3], where the utilization of P is monitored according to equation 4 in addition to the total amount of migrations M. Given that workloads are composed of 10000 tasks, we refer to the average number of migrations per task (AMT) as $M/10000$.



(a) Average utilization of resources.



(b) Average number of migrations per task (AMT). In dotted lines, the AMT of avalanches involving task migrations of less than 100 hops. Note the logscale in the y-axis.

Fig. 4. Sandpile dynamics for increasing workloads (N_i) in a 10×10 toroidal grid topology. Results are averaged over 20 independent runs.

Fig. 4 depicts the response of the sandpile to the different workloads $N_{1 \leq i \leq 120}$ in terms of resource utilization and AMT. In Fig. 4a it can be seen how the utilization of resources scales linearly with the increasing size of the tasks when $N_{i \leq 100}$. This trend turns into a plateau for $N_{i>100}$ indicating that the processing elements are overloaded so

3. The source-code for the simulations is publicly available at https://sandpile-scheduler.googlecode.com published under GNU public licence.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2016.2582160, IEEE Transactions on Parallel and Distributed Systems

that tasks must queue in the system before completion. Such results represent a near-optimal solution to the given test-case if compared to the optimal allocation of tasks produced by the round-robin assignment policy. Therefore we can conclude that, without any central control, the self-organizing dynamics of the sandpile maximize the utilization of the architecture.

The price to a high utilization is, however, the amount of migrations required. In this sense, Fig. 4b shows the AMT associated with previous results. The figure reveals distinct trends that are framed in boxes for a more detailed analysis. The first trend, in box A, holds for $N_{i \lesssim 90}$ and shows a graceful increase of the AMT which always remains below 1. That means that the system is able to yield utilizations of up to $\overline{U} \simeq 0.9$ by migrating tasks less than once. This trend changes in box B as approaching $N_{i=100}$: a critical point giving rise to a phase transition, i.e. between the graceful role of migrations in box A and the frenetic activity in box C, the system has entered in a state of *self-organized criticality* in a very thin region between order and chaos.

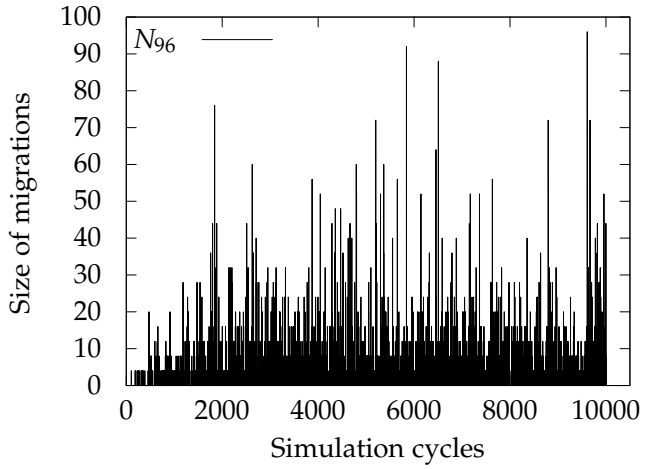


Fig. 5. Runtime dynamics of migrations for N_{96}. The size of avalanches is not homogeneously distributed along the run.

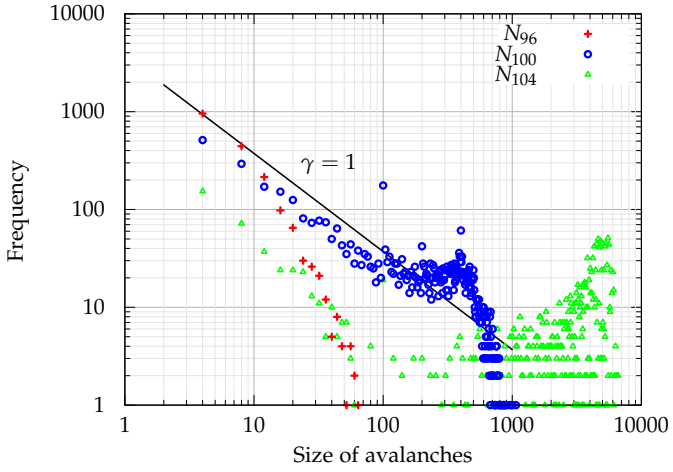


Fig. 6. Distributions of avalanches for $N_{96}(+)$, $N_{100}(\circ)$ and $N_{104}(\triangle)$. The *SOC* phenomenom [14], [33] is typically associated to the $\gamma = 1$ exponent in a power-law $cx^{-\gamma}$.

To gain further insights into this particular state, let us investigate the runtime dynamics of the workloads N_{96}, N_{100} and N_{104} at the phase transition. As exemplified in Fig. 5, a first thing to be noticed is that migrations are not homogeneously distributed along the run but they come in bursts or *avalanches*. A fingerprint of *SOC* systems is that the frequency of such avalanches is in power-law proportion to their sizes: a behavior that emerges without any tuning of parameters but as an inherent property of the system (see e.g. [14]). Fig. 6 compiles these frequency-size curves for the respective workloads under study. The results provide some hints about the dynamics on the phase transition: strictly speaking, neither N_{96} nor N_{104} exhibit the characteristics of a true power-law. On the one hand, N_{96} lacks the representative long tail of such a kind of distribution. On the other hand, N_{104} presents an *U-shaped* distribution with an attractor in large avalanches. In other words, we may say that only the N_{100} scenario is working in a self-organized criticality regime. If we recall that the system evolves to such configuration without any central control, the inference is that the sandpile converges to a self-organized critical state when the incoming rate of instructions and the throughput of the system are in balance (e.g. the incoming rate of instruction in N_{100} is 100 instruction per cycle while the throughput of the architecture 10×10 is also 100 instruction per cycle); a fact from which we can extract a first analytical conclusion of the model:

Conclusion 1. *The sandpile load-balancer is naturally driven to critical states when the incoming workload is equal to the maximum throughput of the system.*

The question arising from the previous conclusion is: how can we use this property to perform energy-efficient computing? The following sections present boundless systems as the framework to answer this question.

6 Introducing Boundless Systems

Let us define a computational boundless system (boundless system for short) as a computing architecture P° that has an unlimited number of resources $P^\circ = \{p_1, \ldots, p_\infty\}$ and where every resource $p_j \in P^\circ$ has two possible states, $p_i^+ \in P^+$ (activated) or $p_j^- \in P^-$ (deactivated), holding that:

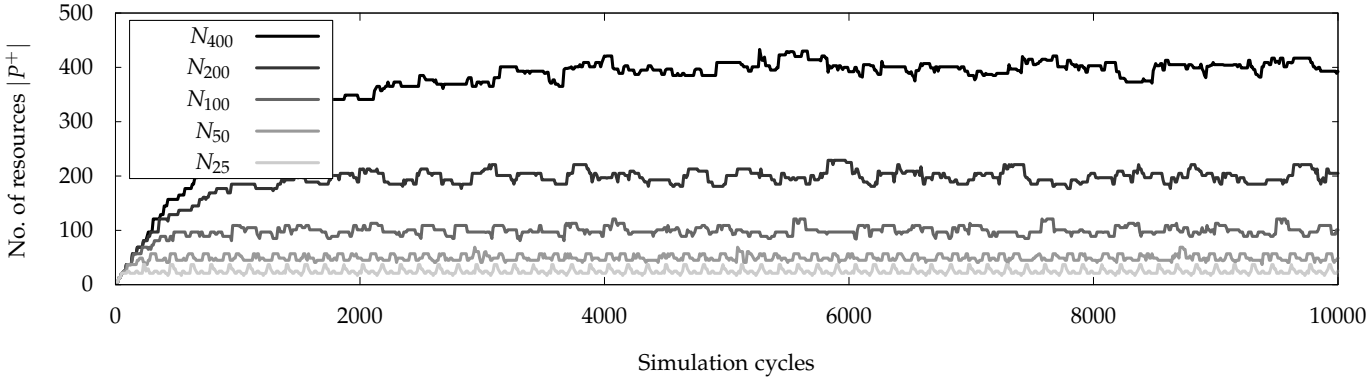
$$\{P^+ \cup P^- = P^\circ\} \wedge \{P^+ \cap P^- = \emptyset\}$$

Although the definition poses a strong theoretical assumption on the availability of resources, many real systems, such as the scale-on-demand services in cloud computing, suffice to support the claim in practice: a boundless system is simply an oversized infrastructure granting access to more resources than those potentially needed.

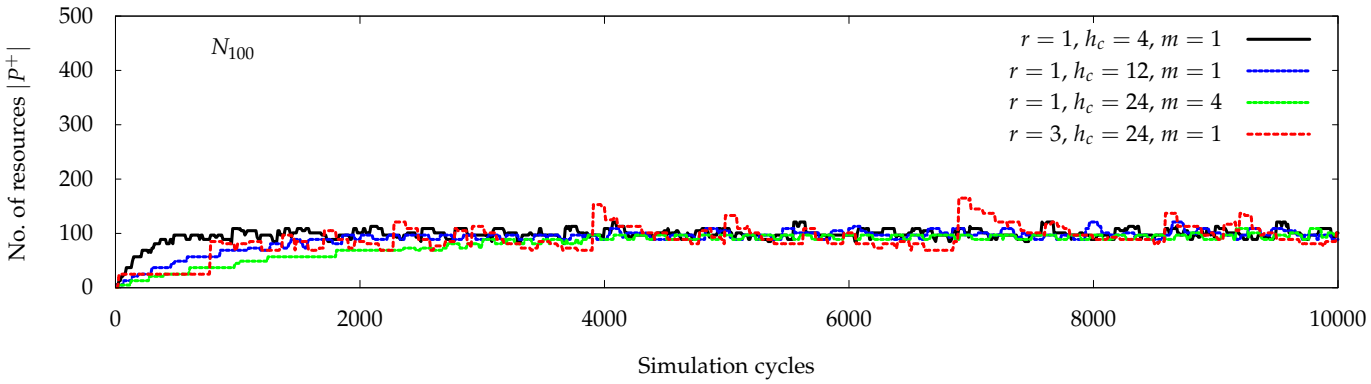
A system of this kind offers a unique theoretical framework to investigate different operational modes of the sandpile. In particular, we propose a boundless system as an ideal platform for dealing with the previously presented problems of under- and overloads. In order to adapt the sandpile model to a boundless system, algorithm 1 will require the following modifications:

1) Instead of being randomly dropped, tasks must be initially assigned to a single resource which acts as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2016.2582160, IEEE Transactions on Parallel and Distributed Systems



(a) Different workloads acting on an equally parameterized sandpile with $r = 1$, $h_c = 4$ and $m = 1$.



(b) Same workload acting on different parameterized sandpiles.

Fig. 7. Number of active resources $P^+ \subset P^\circ$ for different workloads and settings of the sandpile. In all workloads one new task arrives every cycle during the first 10000 cycles of simulation; the i index in N_i stands for the length of tasks, e.g. in N_{25} the length of every task is $n = 25$ instructions.

an entry point to the system, i.e. line 3 in algorithm 1 must be modified by e.g. $p_{0,0}.push(i)$.

2) Calling the *shift* function must return some element. Otherwise the processor p_j^+ is turned off p_j^-.
3) At the calling of the *push* function, a deactivated processor p_j^- is turned on again p_j^+.
4) As a boundless system is not toroidal, there is no chance for endless avalanches and therefore, we do not require to implement a hop limit control.

As a first approach to investigate the sandpile dynamics in a boundless system, Fig. 7 shows the number of active resources P^+ as the sandpile is turning them off and on at runtime. In particular, Fig. 7a analyzes the case in which different workloads arrive to an equally parametrized sandpile. The aim is showing the self-organizing abilities of the sandpile model for adapting the size of the system to the requirements of the workload. In fact, the system remains in dynamic equilibrium around the sizes where the incoming workload and the throughput of the system are in balance (see conclusion 1).

In a second test case, the response of four different parametrized sandpiles is analyzed over the same workload. The parametrization follows the constraint $h_c \geq m \times 2r(r+1)$ established in Table 2 for different r, m and h_c parameters. The aim is at analyzing the sensibility of the model to parametrization. Fig. 7b displays the robustness of the model to parametrization: for any of the settings, the only remarkable difference is on the granularity of a dynamical equilibrium that stabilizes the number of active resources around those values where the incoming ratio of instructions and throughput cancel each other. Hence, we may conclude that robustness is an organic property of the sandpile:

Conclusion 2. *In boundless systems, the sandpile model robustly converges to self-organized states where the number of active resources are proportional to the workload ($P^+ \propto N$).*

7 Trading-off Energy and QoS in Boundless Systems

The self-organizing capacities of the sandpile are an efficient mechanism for adapting the size of the system to the characteristics of the incoming workload in boundless systems. In this section, we aim at investigating the implications of such a behavior in the energy consumption and the QoS delivered by the system. With this in mind, a new type of workload $N_\sim$ is designed such that the tasks follow a sinusoidal pattern of arrival (see Fig. 8).

The energy consumption of the boundless system is analyzed according to equation 6, where the energy E relates to the number of active resources P^+. Furthermore, as a baseline for comparisons, we establish a set of conventional architectures where the number of resources remains fixed throughout the entire run ($P^+ = P$). The sizes in such a set go from a 5×5 to a 10×10 architecture. In order to produce a fair comparison, the baseline set implements the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2016.2582160, IEEE Transactions on Parallel and Distributed Systems

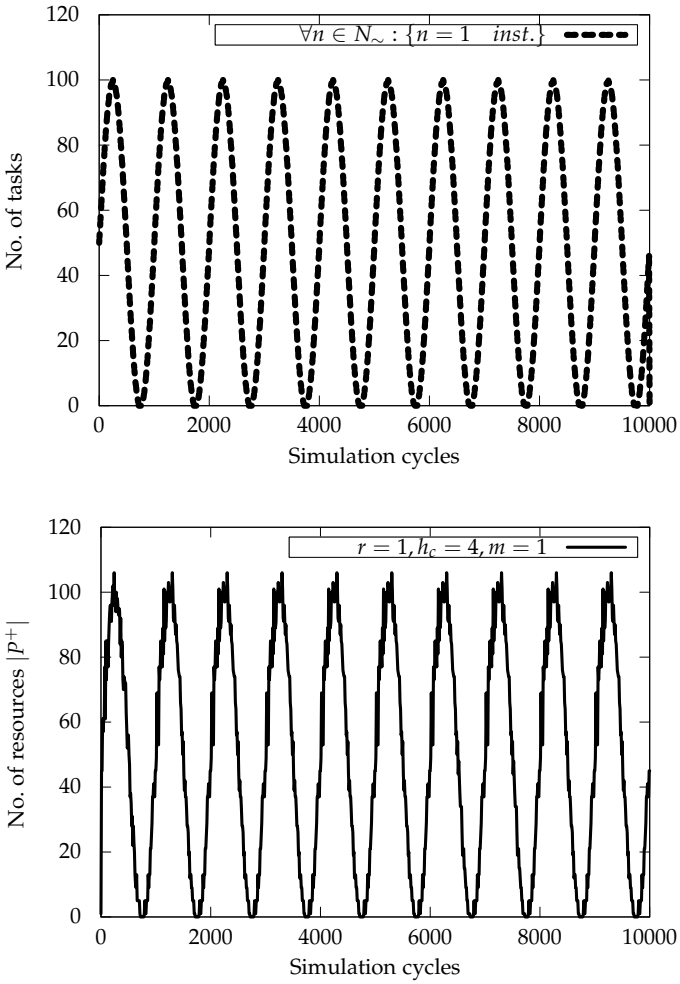


Fig. 8. Sinusoidal pattern of arriving tasks $N_\sim$ (*top*) and respective response of the sandpile model in P°(*bottom*). Every task has a length of one instruction, i.e. $\forall n \in N_\sim : n = 1$ instruction.

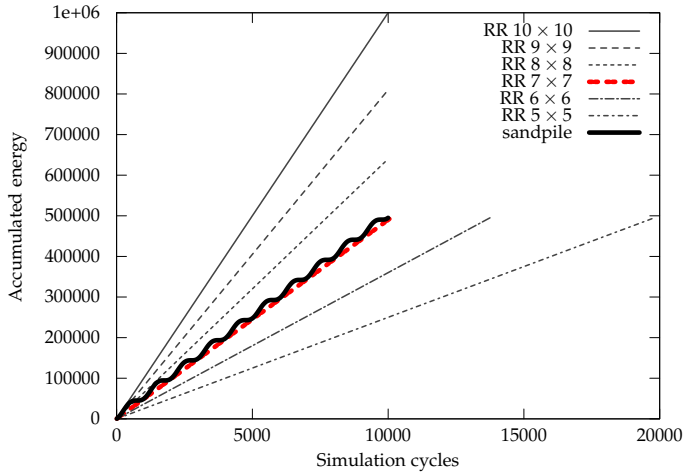


Fig. 9. Energy consumption required for completing the $N_\sim$ workload. The different regular architectures implement a round-robin (RR) scheduling policy, while the sandpile works in a boundless system.

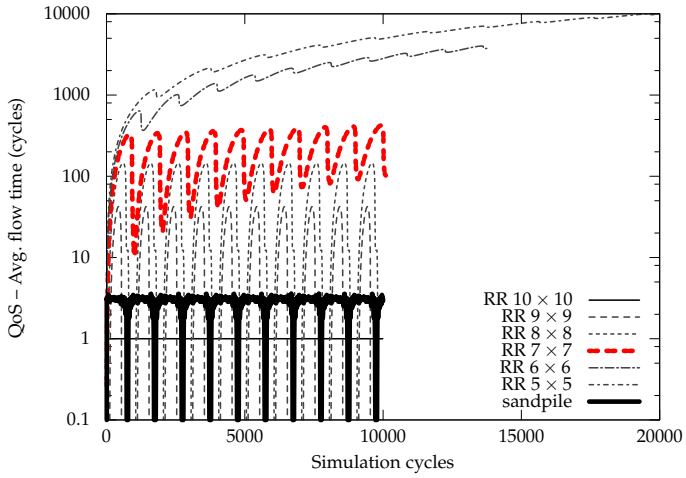


Fig. 10. Average flowtime ($\overline{F}$) of tasks in the different architectures. Please note the log-scale in the y axis.

previously presented *round-robin* (RR) job assignment policy which is optimal when the processing elements and the workloads are homogeneous.

Fig. 9 shows the accumulated energy consumption of the different architectures until completion of all tasks in $N_\sim$. The results point out a conflicting relationship between the two objectives: the time required to process $N_\sim$ and the energy consumed. In that sense, the 7×7 configuration dominates all the rest of *round-robin* counterparts and only the sandpile is able to drive the boundless system to the same results in energy and time. In fact, the average number of active resources in the boundless system is $q^+ \simeq 50$ while in the 7×7 architecture is $q = 49$. A remarkable difference is that the sandpile is able to converge to such results in a self-organizing way.

Such results offer a preliminary insight into the trade-off that the sizing of a system represents to the consumption of energy and the QoS delivered. However, the flowtime ($\overline{F}$) defined in equation 1 provides a more accurate metric to measure the actual QoS. According to the flowtime, Fig. 10 shows how the sandpile is able to outperform the *round-robin* counterparts in terms of QoS. Even though the sandpile has been shown energetically equivalent to RR 7×7, the smart activation and deactivation of resources leverages in a better flowtime: the flowtime of the sandpile holds around $\overline{F} \simeq 3$ while the RR 7×7 architecture does around $\overline{F} \simeq 100$. More surprisingly, the sandpile is also superior to higher energy consuming alternatives such as the RR 9×9 architecture. In fact, the only configuration proving a better QoS is the $10x10$ architecture where $\overline{F} = 1$ but where the energy consumption is double. This winning energy/QoS trade-off of the sandpile can be formulated as a third analytical conclusion, which also serves as an answer to the main research question of this paper[4]:

Conclusion 3. *In boundless systems, the sandpile model finds near-optimal trade-offs between the energetic efficiency and the QoS delivered.*

4. Recall that this question is posed at the Introduction: "What is the trade-off between the energy consumed and the QoS delivered by a sandpile distributed system?".

7.1 Test case using a real trace

In order to gain further insights on the runtime dynamics of the sandpile under realistic conditions, we have conducted additional simulations using a real trace from the Grid Workloads Archive[5]. In particular we have considered the 100000 initial tasks from the Auver Grid trace[6]. The parsing of the trace is done straightforwardly by assuming that a second in the trace is equivalent to a simulation cycle.

As a baseline for comparisons, we use the non-blocking implementation of the work-stealing algorithm proposed by Arora, Blumofe and Plaxton in [34] which is a well-established approach in dynamic load-balancing. Furthermore, we also derive optimal boundaries for the trace leveraging on the assumption of a boundless system with infinite homogeneous resources. Specifically, optimality is easy to derive by implementing the following three rules:

1) All resources are deactivated by default $\forall p_j \in P^{\circ} : \{p_j = p_j^-\}$.
2) At arrival (a_i), every task (i) is assigned to a different and unique resource which is turned on for processing the task ($p_j^+.push(i)$).
3) After processing the task, the resource p_j^+ is turned off again p_j^-.

On the one hand, it can be seen that the energy consumption derived by this method is minimal as the resources are only activated when they are doing an effective job by processing the assigned task. On the other hand, these three rules guarantee a minimal flowtime as the queuing time of every task is zero and thus, the flowtime reduces to the processing time. Substituting in equation 1:

$$f(i,P) = n_i + \underbrace{\overbrace{\max(w_{i,j})}^{a_i} - a_i}_{0}$$

Fig. 11 shows the response of the sandpile to the Auver Grid trace and compares it with the optimal number of resources $|P^+|$ established by the method above. Both trends show a strong correlation ($r = 0.915$) which indicates that the sandpile is able to self-organizingly converge to near-optimal states where the number of active resources depends on the incoming workload, i.e. supporting conclusion 2.

As for the QoS delivered, the sandpile is compared against the work-stealing approach in Fig. 12. In the case of work stealing, we assume an architecture with a finite number of 510 resources (which was the maximum number of optimal resources required in Fig. 11). The reason to employ a finite architecture and not a boundless system is that, by design, workers are always awake as they need to pro-actively check the status of their respective queues. Thus, work stealing must run in a finite number of resources or otherwise they would consume an infinite amount of energy.

With respect to the obtained results, Fig. 12 shows that both approaches –the sandpile and work stealing– are able to dynamically yield near-optimal solutions showing

5. http://gwa.ewi.tudelft.nl/ Accessed on April 2016
6. Available at ~/datasets/gwa-t-4-auvergrid

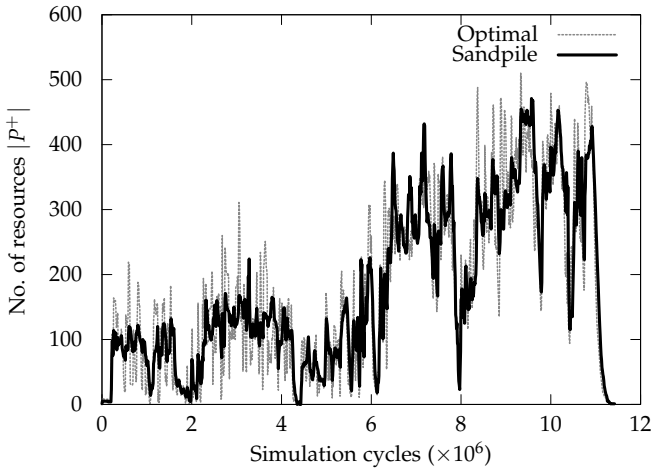


Fig. 11. Adaptive behavior of the sandpile for the Auver Grid trace compared with the optimal number of resources. The peak of active resources is 510, reached around the simulation cycle 9.5×10^6.

a strong correlation with the optimal flowtime, i.e. the respective correlation coefficients are $r = 0.764$ and $r = 0.778$. Although work stealing seems to yield slightly better results than the sandpile, it has to be noted that the optimal size of the architecture has required to be fine-tuned beforehand to that end while the sandpile yields such values in a self-organized manner.

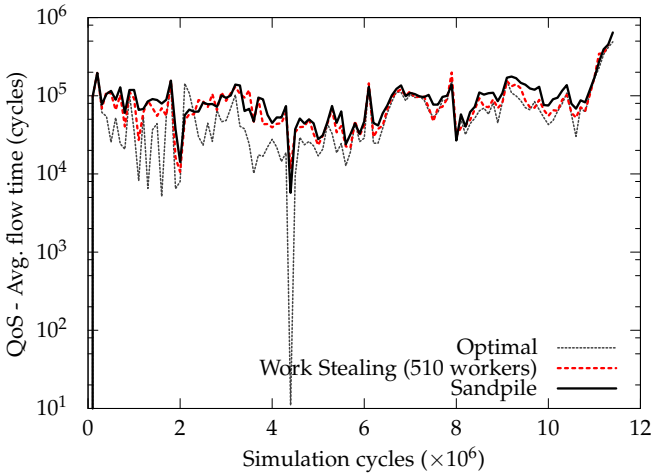


Fig. 12. Average flowtime ($\overline{F}$) of tasks for the sandpile and the work stealing approaches compared against the optimal flowtime. Please note the log-scale in the y axis.

Finally, Fig. 13 shows the energy consumption of the different approaches until the completion of the Auver Grid trace. As the working mechanism of work stealing requires that all resources are always awake and pro-actively checking the status of their piles, the energy consumption of the approach grows linearly with respect to the simulation time. Meanwhile, the self-organizing dynamics of the sandpile are able to yield an optimal energy consumption by turning resources on and off at runtime. For a real trace like the one presented here, that can add up to an energy saving factor of 2.65.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2016.2582160, IEEE Transactions on Parallel and Distributed Systems

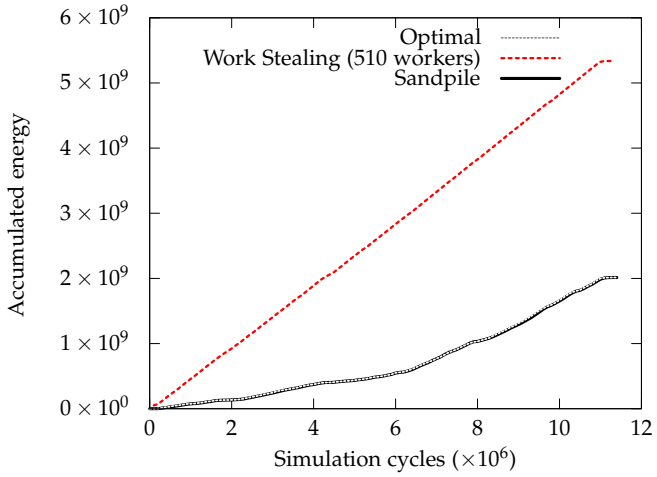


Fig. 13. Energy consumption of the different approaches for completing the Auver Grid workload. The optimal and the sandpile trends overlap to the naked eye.

Overall, the presented test case (which assumes realistic conditions for the workload) supports conclusion 3 on the near-optimal trade-off that the sandpile yields in terms of energetic efficiency and QoS. Nonetheless, more interesting is that these results provide some evidence on the potentialities of SOC systems when applied to energy-efficient dynamic load-balancing, which is the main motivation of this exploratory investigation.

8 Conclusions

In this paper, we present a self-organized criticality approach for dynamically load-balancing computational workloads. The model is inspired by the Bak-Tang-Wiesenfeld sandpile: a cellular automaton reaching critical states at the edge of chaos that are released in the form of avalanches.

With some minimal modifications, the sandpile model is extended to deal with the problem of scheduling independent tasks. To that end, we consider the case of a boundless system: a topology with an infinite number of sites. All sites are initially deactivated, so that the system does not consume energy, however every site can be turned on at runtime in order to process some tasks.

We show that the decentralized execution of the sandpile is able to self-organizingly adapt the number of active resources to the particularities of incoming workloads with a simple mechanism: when the number of tasks in a site exceed a certain threshold, the site reassigns those tasks to its neighbors. Given that all sites implement the same rule, a simple reassignment of tasks may start an avalanche that will propagate until the system finds a new state of equilibrium. This emergent load-balancing response of the system is analyzed in terms of two conflicting objectives: minimizing the energy consumption (i.e. number of active resources through time) and maximizing the QoS (i.e. the average waiting time of tasks in the system). In this context, the conducted experimentation shows that the approach is able to converge towards near-optimal trade-offs where the energetic efficiency and the QoS are maximized.

In future works, we aim at investigating the applicability of the sandpile in real computing systems. We will consider the most suitable domain from a set of candidate applications such as the consolidation of virtual machines in cloud computing systems, or the scheduling of scientific applications in modern HPC centers. Either way, the challenging part will be minimizing the costs associated to the migration of tasks for which we envision two complementary strategies. On the one hand, the exploratory character of the current investigation leaves much room for further improvements on the model, including the study of optimal interconnection topologies or different migration policies. On the other hand, one of the major potentials of the model lies on its hybridization with other scheduling approaches where, for instance, the sandpile could serve as a mere signalling protocol to control the power states of the system.

Acknowledgments

This work was supported by the Upper Normandy Region GRR-MRT PROTEC project, the scientific network TERA-MRT, and the Luxembourg INTER/CNRS/11/03 Green@Cloud project.

References

[1] J. Koomey, "Growth in data center electricity use 2005 to 2010," Oakland, CA: Analytics Press, 2011, accessed on June 2015. [Online]. Available: http://www.analyticspress.com/datacenters.html

[2] CIA, "The world factbook," accessed on June 2015. [Online]. Available: https://www.cia.gov/library/publications/resources/the-world-factbook/

[3] L. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, Dec 2007.

[4] K. Choi, R. Soma, and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 24, no. 1, pp. 18–28, 2005.

[5] D. J. Brown and C. Reams, "Toward energy-efficient computing," *Commun. ACM*, vol. 53, no. 3, pp. 50–58, Mar. 2010. [Online]. Available: http://doi.acm.org/10.1145/1666420.1666438

[6] M. Guzek, J. E. Pecero, B. Dorronsoro, and P. Bouvry, "Multi-objective evolutionary algorithms for energy-aware scheduling on distributed computing systems," *Applied Soft Computing*, vol. 24, no. 0, pp. 432 – 446, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494614003408

[7] V. Sarkar, S. Amarasinghe, D. Campbell, W. Carlson, A. Chien, W. Dally, E. Elnohazy, M. Hall, R. Harrison, W. Harrod *et al.*, "Exascale software study: Software challenges in extreme scale systems," *ExaScale Computing Study, DARPA IPTO*, 2009.

[8] K. Wang and I. Raicu, "Paving the road to exascale with many-task computing," *Doctoral Showcase, IEEE/ACM Supercomputing/SC*, 2012.

[9] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007, vol. 703.

[10] J. Kołodziej, S. U. Khan, L. Wang, and A. Y. Zomaya, "Energy efficient genetic-based schedulers in computational grids," *Concurrency and Computation: Practice and Experience*, vol. 27, pp. 809–829, 2012.

[11] H. Sheikh, I. Ahmad, and D. Fan, "An evolutionary technique for performance-energy-temperature optimized scheduling of parallel tasks on multi-core processors," *Parallel and Distributed Systems, IEEE Transactions on*, 2015.

[12] K. D. Devine, E. G. Boman, R. T. Heaphy, B. A. Hendrickson, J. D. Teresco, J. Faik, J. E. Flaherty, and L. G. Gervasio, "New challenges in dynamic load balancing," *Appl. Numer. Math.*, vol. 52, no. 2-3, pp. 133–152, Feb. 2005. [Online]. Available: http://dx.doi.org/10.1016/j.apnum.2004.08.028

[13] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load balancing and unbalancing for power and performance in cluster-based systems," in *Workshop on compilers and operating systems for low power*, vol. 180. Barcelona, Spain, 2001, pp. 182–195.

[14] P. Bak, C. Tang, and K. Wiesenfeld, "Self-organized criticality: An explanation of the $1/f$ noise," *Phys. Rev. Lett.*, vol. 59, pp. 381–384, Jul 1987. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.59.381

[15] P. Bak, *How nature works: the science of self-organized criticality*. Springer Science & Business Media, 2013.

[16] Z. Olami, H. J. S. Feder, and K. Christensen, "Self-organized criticality in a continuous, nonconservative cellular automaton modeling earthquakes," *Phys. Rev. Lett.*, vol. 68, pp. 1244–1247, Feb 1992. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.68.1244

[17] R. V. Solé and S. C. Manrubia, "Extinction and self-organized criticality in a model of large-scale evolution," *Phys. Rev. E*, vol. 54, pp. R42–R45, Jul 1996. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.54.R42

[18] J. M. Beggs and D. Plenz, "Neuronal avalanches in neocortical circuits," *The Journal of neuroscience*, vol. 23, no. 35, pp. 11 167–11 177, 2003.

[19] S. Wolfram, "Universality and complexity in cellular automata," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1, pp. 1–35, 1984.

[20] C. G. Langton, "Computation at the edge of chaos: phase transitions and emergent computation," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1, pp. 12–37, 1990.

[21] M. Willebeek-LeMair and A. Reeves, "Strategies for dynamic load balancing on highly parallel computers," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 4, no. 9, pp. 979 –993, sep 1993.

[22] M. Jelasity, A. Montresor, and O. Babaoglu, "A modular paradigm for building self-organizing peer-to-peer applications," in *In Engineering Self-Organising Systems, G. Di Marzo Serugendo*. Springer, 2003, pp. 265–282.

[23] M. Salman, C. Bertelle, and E. Sanlaville, "The behavior of load balancing strategies with regard to the network structure in distributed computing systems," in *Signal-Image Technology and Internet-Based Systems (SITIS), 2014 Tenth International Conference on*, Nov 2014, pp. 432–439.

[24] J. Hu and R. Klefstad, "Decentralized load balancing on unstructured peer-2-peer computing grids," in *Network Computing and Applications, 2006. NCA 2006. Fifth IEEE International Symposium on*, july 2006, pp. 247 –250.

[25] S. Khan and I. Ahmad, "A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 20, no. 3, pp. 346–360, March 2009.

[26] J. Laredo, P. Bouvry, F. Guinand, B. Dorronsoro, and C. Fernandes, "The sandpile scheduler," *Cluster Computing*, vol. 17, no. 2, pp. 191–204, Jun. 2014. [Online]. Available: http://dx.doi.org/10.1007/s10586-013-0328-x

[27] J. Laredo, B. Dorronsoro, J. Pecero, P. Bouvry, J. Durillo, and C. Fernandes, "Designing a self-organized approach for scheduling bag-of-tasks," in *P2P, Parallel, Grid, Cloud and Internet Computing, 2012 Seventh International Conference on*, Nov 2012, pp. 315–320.

[28] C.-C. Chen, L.-Y. Chiao, Y.-T. Lee, H. wen Cheng, and Y.-M. Wu, "Long-range connective sandpile models and its implication to seismicity evolution," *Tectonophysics*, vol. 454, no. 4, pp. 104–107, 2008.

[29] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[30] J. D. C. Little, "A proof for the queuing formula: $l = \lambda w$," *Operations Research*, vol. 9, no. 3, pp. 383–387, 1961.

[31] M. Guzek, S. Varrette, V. Plugaru, J. Pecero, and P. Bouvry, "A holistic model of the performance and the energy-efficiency of hypervisors in an hpc environment," in *Energy Efficiency in Large Scale Distributed Systems*, ser. Lecture Notes in Computer Science, J.-M. Pierson, G. Da Costa, and L. Dittmann, Eds. Springer Berlin Heidelberg, 2013, pp. 133–152. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40517-4_13

[32] M. Harchol-Balter, M. E. Crovella, and C. D. Murta, "On choosing a task assignment policy for a distributed server system," *Journal of Parallel and Distributed Computing*, vol. 59, no. 2, pp. 204 – 228, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731599915770

[33] P. Bak and K. Sneppen, "Punctuated equilibrium and criticality in a simple model of evolution," *Phys. Rev. Lett.*, vol. 71, pp. 4083–4086, Dec 1993. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.71.4083

[34] N. S. Arora, R. D. Blumofe, and C. G. Plaxton, "Thread scheduling for multiprogrammed multiprocessors," in *Proceedings of the Tenth Annual ACM Symposium on Parallel Algorithms and Architectures*, ser. SPAA '98. New York, NY, USA: ACM, 1998, pp. 119–129. [Online]. Available: http://doi.acm.org/10.1145/277651.277678