Trading Off Resource Utilization and Task Migrations in Dynamic Load-balancing

[Extended Abstract]

Juan Luis J. Laredo*
LITIS lab, Université du Havre, France
jimenezj@univ-lehavre.fr

Frédéric Guinand
LITIS lab, Université du Havre, France
frederic.guinand@univ-lehavre.fr

Damien Olivier
LITIS lab, Université du Havre, France
damien.olivier@univ-lehavre.fr

Pascal Bouvry
FSTC-CSC/SnT, University of Luxembourg, Luxembourg
pascal.bouvry@uni.lu

ABSTRACT

This paper aims at investigating platform-independent measures to determine the efficiency of a load-balancer. Counterpoised curves are proposed as a mean to counterweight the utilization of resources with a penalty due to the migration of tasks. As a proof-of-concept, experiments were conducted on the sandpile load-balancer showing that its maximum counterpoised efficiency is reached at 80% of the peak performance of the system.

CCS Concepts

•**Computer systems organization** → **Self-organizing autonomic computing;** •**Theory of computation** → *Scheduling algorithms;* •**General and reference** → Metrics;

Keywords

cellular automata; online algorithms; self-adaptation

1. INTRODUCTION

In Computer Architecture, scheduling is the problem of assigning tasks to resources: a problem that has been proven NP-hard [1]. Among different scheduling methods, dynamic load-balancing consists in the on-line reallocation of tasks in distributed systems. The goal is to minimize the overall completion time of an incoming workload by maximizing the utilization of the computing resources via migration of tasks. Naturally, an increase on the migration of tasks carries an associated penalty that is heavily dependent on the speed and capacity of the interconnection network. Such a dependency introduces a bias when comparing the efficiency of load-balancers in different platforms.

We propose counterpoised curves as a mean to avoid bias and obtain an estimate of the real efficiency of a load-balancer independently of the platform. To that end, the communication costs are computed as a simple count of the number of migrating tasks: the counterpoised efficiency keeps an inversely proportional relation to such a value. As a proof-of-concept, experiments are conducted on a simplified version of the sandpile load-balancer presented in [2, 3].

*Corresponding author.

GECCO '15 July 11-15, 2015, Madrid, Spain

ACM ISBN 978-1-4503-3488-4/15/07.
DOI: http://dx.doi.org/10.1145/2739482.2764677

2. COUNTERPOISED CURVES

Counterpoised curves aim to be a platform-independent benchmark for establishing fair comparisons between load-balancers. In short, the benchmark is composed of three elements:

- A set of $P = \{p_1, \ldots, p_{100}\}$ processors arranged in a 10×10 toroidal grid (see Fig. 1). Each processor has coordinates (x, y) and a processing speed of *1 instruction/cycle*.

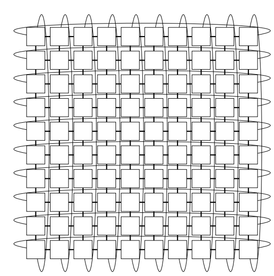

Figure 1: 10×10 **toroidal grid arrangement.**

- A set of workloads $W = \{w_1, \ldots, w_{100}\}$, each w_i being composed of a set tasks (t) such that:

$$w_i = \begin{pmatrix} t_{1,1} & \ldots & t_{1,i} \\ \vdots & \ddots & \vdots \\ t_{1000,1} & \ldots & t_{1000,i} \end{pmatrix}$$

where every task t is composed of a single instruction to process. Rows represent the sequential time of arrival of the tasks. Note that the rate of incoming tasks depends on the index i of the workload w_i, i.e. *1 task/cycle* for w_1, *2 tasks/cycle* in w_2 and so on.

- Every task t is independently assigned at arrival to a randomly chosen processor in P.

Previous elements provide a simple framework defining a common architecture, workload and initial assignation of tasks. Given these initial conditions, every workload w_i will be computed in P according to the dynamics of the load-balancer being tested such that the total amount of migrations is stored in $M(w_i)$. If we let $|w_i|$ be the number of tasks in a workload w_i, the average migrations per task (AMT) can be expressed as $\frac{M(w_i)}{|w_i|}$.

Using previous values, the counterpoised efficiency curve is defined according to the following equation:

$$\breve{E}(w_i) = \underbrace{\overline{U}(w_i)}_{utilization} \times \underbrace{e^{-\frac{M(w_i)}{|w_i|}}}_{counterpoise}$$

where $\overline{U}(w_i)$ is the average utilization of the architecture P until completion of the workload w_i; and $e^{-\frac{M(w_i)}{|w_i|}}$ is an inversely proportional factor to the exponential of AMT.

3. CASE OF STUDY: THE SANDPILE

In [2, 3], we proposed a load-balancer following the self-organized criticality principle described by Bak, Tang and Wiesenfied for the sandpile model [4]. The load-balancer features a sandpile cellular automaton, in which sites stand for processing elements and tasks follow the analogy of grains of sand that pile up and slip through such sites.

In this paper, we conduct experiments for a simplified version of the sandpile load-balancer with these rules: initially, adding a task/grain at a site (x, y) results in a simple increase on the height of the pile $h(x, y)$, such that:

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

Given that tasks are processed with a given speed, when a task is fetched for processing the corresponding pile shrinks:

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

The balance between incoming and processed tasks is broken if the height exceeds a certain threshold value $h(x, y) \geq 4$ such that the site collapses and loses a number of tasks:

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

which are then reassigned to its neighbors:

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

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

This initial redistribution of tasks may start a chain reaction by recursively triggering the collapse of neighbors. Therefore, and despite the simplicity of the rules, avalanches of all sizes may take place for critical states of the system. Intuitively such a behavior has a load-balancing effect on the workload that we aim at analyzing using counterpoised curves.

For the settings in this investigation, the maximum throughput of P is at $10 \times 10 = 100$ *tasks/cycle* so that the utilization increases with the index i of the workload, i.e. the maximum utilization can only be reached for $w_{100} = 100$ *tasks/cycle*. Increasing the workload of the system, however, can intensify the migration of tasks. The aim of a counterpoised curve is to find good trade-offs out of such dynamics.

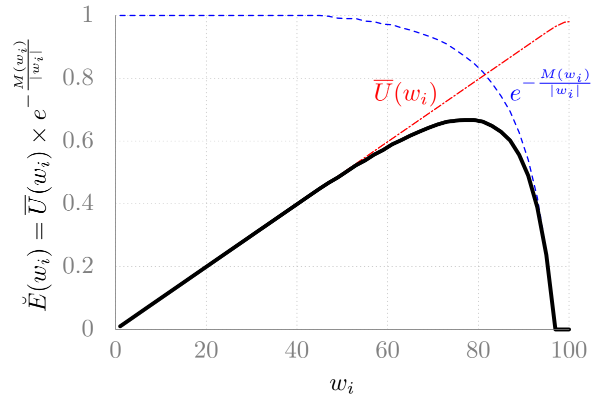

Figure 2: Sandpile counterpoised efficiency. Results are averaged over 20 independent runs.

Fig. 2 shows the counterpoised efficiency curve capturing the sandpile dynamics. Results indicate that the sandpile load-balancer finds best trade-offs for workloads requiring $70 - 80\%$ of the system capacity. In particular, the peak value is at $\breve{E}(w_{78}) = 0.67$ followed by a rapid deterioration of the counterpoised efficiency. Such results indicate a possible weakness of the sandpile approach when targeting heavily loaded scenarios.

4. CONCLUSIONS

In this paper, we have presented counterpoised curves as a tool for the analysis of load-balancers independently of underlying platforms. A simplistic sandpile load-balancer has been used to prove the concept, showing that the curves aid interpretation to the analysis of the dynamics.

Acknowledgments

Work 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.

5. REFERENCES

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

[2] J. Laredo, P. Bouvry, F. Guinand, B. Dorronsoro, and C. Fernandes. The sandpile scheduler. *Cluster Computing*, 17(2):191–204, June 2014.

[3] 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*, pages 315–320, Nov 2012.

[4] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality: An explanation of the $1/f$ noise. *Phys. Rev. Lett.*, 59:381–384, Jul 1987.