Theoretical Computer Science 590 (2015) 129–146

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Online parallel scheduling of non-uniform tasks: Trading failures for energy ☆

Antonio Fernández Anta [a], Chryssis Georgiou [b], Dariusz R. Kowalski [c], Elli Zavou [a,d,*,1]

[a] IMDEA Networks Institute, Spain
[b] University of Cyprus, Cyprus
[c] University of Liverpool, United Kingdom
[d] Universidad Carlos III de Madrid, Spain

ARTICLE INFO

Article history:
Received 12 November 2013
Received in revised form 15 July 2014
Accepted 16 January 2015
Available online 21 January 2015

Keywords:
Scheduling
Non-uniform tasks
Failures
Competitiveness
Online algorithms
Energy efficiency

ABSTRACT

Consider a system in which tasks of different execution times arrive continuously and have to be executed by a set of machines that are prone to crashes and restarts. In this paper we model and study the impact of parallelism and failures on the competitiveness of such an online system. In a fault-free environment, a simple Longest-In-System scheduling policy, enhanced by a redundancy-avoidance mechanism, guarantees optimality in a long-term execution. In the presence of failures though, scheduling becomes a much more challenging task. In particular, no parallel deterministic algorithm can be competitive against an off-line optimal solution, even with one single machine and tasks of only two different execution times. We find that when additional energy is provided to the system in the form of processing speedup, the situation changes. Specifically, we identify thresholds on the speedup under which such competitiveness cannot be achieved by any deterministic algorithm, and above which competitive algorithms exist. Finally, we propose algorithms that achieve small bounded competitive ratios when the speedup is over the threshold.

1. Introduction

Motivation In recent years we have witnessed a dramatic increase on the demand of processing computationally-intensive jobs. Uniprocessors are no longer capable of coping with the high computational demands of such jobs. As a result, multicore-based parallel machines such as the K-computer [35] and Internet-based supercomputing platforms such as SETI@home [26] and EGEE Grid [15] have become prominent computing environments. However, computing in such environments raises several challenges. For example, computational jobs (or tasks) are injected dynamically and continuously, each job may have different computational demands (e.g., CPU usage or processing time) and the processing elements are subject to unpredictable failures. Preserving power consumption is another challenge of rising importance. Therefore, there is a corresponding need for developing algorithmic solutions that would efficiently cope with such challenges.

☆ This research was supported in part by the Comunidad de Madrid grant Cloud4BigData-CM (S2013/ICE-2894), Spanish MICINN/MINECO grant TEC2011-29688-C02-01, NSF of China grant 61020106002, and FPU12/00505 Grant from MECD. A preliminary version of this work appears in the proceedings of FCT 2013.

* Corresponding author.
E-mail address: elli.zavou@imdea.org (E. Zavou).

[1] Partially supported by FPU grant from MECD.

http://dx.doi.org/10.1016/j.tcs.2015.01.027
0304-3975/

Table 1
Important notation and definitions.

Term	Description
$m \in \mathbb{N}$	Number of machines in the system
$s \geq 1$	Machine's speedup
$c_{\min}$	Smallest task cost
$c_{\max}$	Largest task cost
c-task	Task of cost $c \in [c_{\min}, c_{\max}]$
$\rho = \frac{c_{\max}}{c_{\min}}$	Cost ratio
$\gamma = \max\{\lceil \frac{\rho - s}{s-1} \rceil, 0\}$	Parameter γ used to define competitiveness thresholds
β	Parameter used for redundancy avoidance
Condition C1	$s < \rho$
Condition C2	$s < 1 + \gamma/\rho$

Table 2
Summary of results. We define $\gamma = \max\{\lceil \frac{\rho - s}{s-1} \rceil, 0\}$ to be a parameter representing the number of $c_{\min}$-tasks that, in addition to a $c_{\max}$-task, an algorithm with speedup s can complete in a time interval of length $(\gamma + 1)c_{\min}$. Parameter β is a parameter of Algorithms LIS and LAF, used for avoiding redundancy of task executions. Also note that $\min\{\rho, 1 + \gamma/\rho\} < 2$; this follows from the definitions of γ and ρ, and from $s \geq 1$.

Condition	Number of task costs	Task competitiveness	Cost competitiveness	Algorithm
C1 $\wedge$ C2	≥ 2	∞	∞	Any
$\neg$C1	Any	1	ρ	(m, β)-LIS
C1 $\wedge \neg$C2	2	1	1	γm-Burst
$s \geq 7/2$	Finite	ρ	1	(m, β)-LAF

Much research has been dedicated to task scheduling problems over the last decades, each work addressing different challenges (e.g., [8,11–14,16,18,19,21,24,29,34]). For example, many works address the issue of dynamic task injections, but do not consider failures (e.g., [10,22]). Other works consider scheduling on one machine (e.g., [3,30,33]), with the drawback that the power of parallelism is not exploited (provided that tasks are independent). Some works consider failures, but assume that tasks are known a priori and their number is bounded (e.g., [5,7,11,18,19,23,24]), where others assume that tasks are uniform, that is, they have the same processing times (e.g., [16,17]). Several works consider power-preserving issues, but do not consider, for example, failures (e.g., [9,10,34]).

Contributions In this work we consider a computing system in which tasks of *different* execution times arrive *dynamically and continuously* and must be executed by a set of $m \in \mathbb{N}$ machines that are prone to *crashes and restarts*. Due to the dynamicity involved, we view this task-executing problem as an online problem and pursue competitive analysis [2,31]. We explore the impact of parallelism, different task execution times and faulty environment, on the competitiveness of the online system considered. Efficiency is measured as the maximum *number of pending tasks* as well as the maximum *pending cost* over any point in the execution, where pending tasks are the ones that have been injected in the system but are not completed yet, and pending cost is the sum of their execution times. An algorithm is considered to be x-pending-task competitive, if under any adversarial pattern (for both task arrivals and machine crashes and restarts) its pending task complexity is at most x times larger than the pending task complexity of the offline optimal algorithm OPT, under the same adversarial pattern. This holds similarly for x-pending-cost competitiveness, taking into account the pending cost complexity of the algorithms.

We show that no parallel algorithm for the problem under study is competitive against the best off-line solution in the classical sense, however it becomes competitive if static processoring *speed scaling* [6,4,10] is applied in the form of a *speedup* above a certain threshold. A speedup $s \in \mathbb{R}^+$ means that a machine can complete a task s times faster than the task's system specified execution time (and therefore has a meaning only when $s \geq 1$). The use of a speedup is a form of resource augmentation [28] and impacts the *energy consumption* of the machine. As a matter of fact, the power consumed (i.e., the energy consumed per unit of time) to run a machine at a speed x grows superlinearly with x, and it is typically assumed to have a form of $P = x^\alpha$, for $\alpha > 1$ [1,34]. Hence, a speedup s implies an additional factor of $s^{\alpha-1}$ in the power consumed (and hence energy consumed).

Our investigation aims at developing competitive online algorithms that require the smallest possible speedup. As a result, one of the main challenges is to identify the speedup thresholds, under which competitiveness cannot be achieved and over which it is possible. In some sense, our work can be seen as investigating the trade-offs between *knowledge* and *energy* in the presence of failures: How much energy (in the form of speedup) does a deterministic online scheduling algorithm need in order to match the efficiency (i.e., to be competitive with) of the optimal off-line algorithm that possesses complete knowledge of failures and task injections? (It is understood that there is nothing to investigate if the off-line solution makes use of speed-scaling as well).

We now summarize our contributions. Table 1 provides useful notation and definitions, and Table 2 provides an overview of our main results.