Asymptotic Competitive Analysis of Task Scheduling Algorithms on a Fault-Prone Machine

Antonio Fernández Anta[1], Chryssis Georgiou[2], Dariusz R. Kowalski[3] and Elli Zavou[1,4]

[1]IMDEA Networks Institute [2]University of Cyprus [3]University of Liverpool [4]Universidad Carlos III de Madrid

Abstract

Reliable task execution in systems with machines that are prone to unpredictable crashes and restarts is challenging and of high importance. However, not much work exists on the worst case analysis of such systems. In this work, we analyze the fault-tolerant properties of four popular scheduling algorithms, Longest In System, Shortest In System, Largest Processing Time and Shortest Processing Time, under worst case scenarios on a fault-prone machine. We define three metrics for the evaluation and comparison of their competitive performance in the long run, namely, completed cost, pending cost and latency. We also investigate the effect of resource augmentation by increasing the speed of the machine. Finally, we compare their behavior for different speed intervals concluding that each of them behaves better than the rest under different circumstances and none is clearly the best.

Motivation. The demand for processing computationally-intensive jobs has been increasing dramatically during the last decades, and so has the research to face the many challenges it presents. What is more, in the presence of machine failures (and restarts) things get even worse. One of the techniques used to provide a guaranteed level of service in such cases, is to use multiple machines as a form of resource augmentation. In this work however, we consider a different form of resource augmentation to overcome the failures, that is applying *speed augmentation* [8] in order to increase the computational power of the system's machine. Hence, we consider a speedup $s \geq 1$, under which the machine performs a job s times faster than the baseline execution time.

Model. We consider a system with a single *machine* prone to crashes and restarts, and a *Scheduler* that assigns injected *tasks* to be executed by the machine, following some algorithm. These tasks arrive continuously and have different computational demands referred to as *cost* (i.e., CPU usage or processing time). The task injection is an operation controlled by an arrival pattern A. Specifically, we assume that each task w has an arrival time $a(w)$ and a cost $c(w) \in [c_{min}, c_{max}]$, where c_{min} and c_{max} are the smallest and largest costs respectively. The task cost represents the time it needs to be completed by a machine running with $s = 1$. We also assume tasks to be *atomic* with respect to their completion (preemption is not allowed). What is more, we consider crashes and restarts of the machine to be controlled by an error pattern E that coordinates with the arrival pattern to give the worst-case scenarios. The task being executed at the time of the machine failure is not completed, and has to be eventually re-scheduled.

We explore the behavior of some of the most widely used algorithms (or policies) in scheduling, analyzing their fault-tolerant properties under worst-case combination of task injection and crash/restart patterns, as described above. The four algorithms we consider are: *Longest In System* (LIS), under which the task that has been waiting the longest is scheduled, *Shortest In System* (SIS), under which the task that has been injected the latest is scheduled, *Largest Processing Time* (LPT), under which the task with the biggest cost is scheduled, and *Shortest Processing Time* (SPT), under which the task with the smallest cost is scheduled. We focus on three *evaluation metrics*, which we consider to embody the most important quality of service parameters: machine utilization (performance), buffering (queue size) and user fairness (lateness). The *completed cost,* $C_t^s(\mathrm{ALG}, A, E) = \sum_{w \in N_t^s(\mathrm{ALG},A,E)} c(w)$, the *pending cost,* $P_t^s(\mathrm{ALG}, A, E) = \sum_{w \in Q_t^s(\mathrm{ALG},A,E)} c(w)$, and the *latency,* $L_t^s(\mathrm{ALG}, A, E) = \max\{f(w) - a(w) : \forall w \in N_t^s(\mathrm{ALG}, A, E), t - a(w) : \forall w \in Q_t^s(\mathrm{ALG}, A, E)\}$, where $f(w)$ is the time of completion of task w. Note that, computing the schedule (and hence finding the algorithm) that minimizes or maximizes correspondingly the measures $C_t^s(X, A, E), P_t^s(X, A, E)$, and $L_t^s(X, A, E)$ off-line (having the knowledge of the patterns A and E), is an NP-hard problem [2].

Since the scheduling decisions must be made continuously and without knowledge of neither the future task injections nor the machine crashes and restarts, we see the problem as an *online* scheduling problem [9, 2, 1] and perform

	Condition	Completed Cost, $\mathcal{C}$	Pending Cost, $\mathcal{P}$	Latency, $\mathcal{L}$
LIS	$s < \rho$	0	∞	∞
	$s \in [\rho, 1 + 1/\rho)$	$[\frac{1}{\rho}, \frac{1}{2} + \frac{1}{2\rho}]$	$[\frac{1+\rho}{2}, \rho]$	$(0, 1]$
	$s \in [\max\{\rho, 1 + 1/\rho\}, 2)$	$[\frac{1}{\rho}, \frac{1}{2} + \frac{c_{min}}{2c}]$	$[\frac{1}{2} + \frac{c}{2c_{min}}, \rho]$	$(0, 1]$
	$s \geq \max\{\rho, 2\}$	$[1, s]$	1	$(0, 1]$
SIS	$s < \rho$	0	∞	∞
	$s \in [\rho, 1 + \rho)$	$[\frac{1}{\rho}, \frac{c_{min}}{c'}]$	$[\frac{c' + c_{max}}{c_{min} + c_{max}}, \rho]$	∞
	$s \geq 1 + \rho$	$[1, s]$	1	∞
LPT	$s < \rho$	0	∞	∞
	$s \geq \rho$	$[1, s]$	1	∞
SPT	$s = 1$	$\frac{1}{1+\rho}$	∞	∞
	$s \in (1, \rho)$	$[1, \frac{\hat{\rho}}{\hat{\rho}+\rho}]$	∞	∞
	$s \geq \rho$	$[1, s]$	1	∞

Table 1: Metrics comparison of the four scheduling algorithms. Recall that $\rho = \frac{c_{max}}{c_{min}}$ and $\hat{\rho} = \lceil \rho \rceil - 1$, and task costs $c, c' \in (c_{min}, c_{max})$ are such that $c < \frac{c_{min}}{s-1}$ and $c' < \frac{c_{min}+c_{max}}{s}$.

asymptotic competitive analysis [3, 11] to evaluate the performance of the *online algorithms*, under *worst-case* scenarios, which will guarantee efficient scheduling even in the worst of cases.

We use the so called *asymptotic performance ratio* [11] of the metric, which applies the idea of long term competitiveness ratio over the sets of arrival and error patterns $\mathcal{A}$ and $\mathcal{E}$, and against any algorithm in the set $\mathcal{X}$ designed to solve the scheduling problem. It is important to note that we only consider combinations of arrival and error patterns $A \in \mathcal{A}$ and $E \in \mathcal{E}$ such that $\exists X \in \mathcal{X}$ for which $\lim_{t\to\infty} C_t^s(X, A, E) = \infty$, and which produce worst-case scenarios. The three asymptotic performance measures are therefore defined as:

Completed Cost: $\mathcal{C}^s(\text{ALG}, \mathcal{A}, \mathcal{E}) = \inf_{A\in\mathcal{A}, E\in\mathcal{E}, X\in\mathcal{X}} \lim_{t\to\infty} \frac{C_t^s(\text{ALG}, A, E)}{C_t^1(X, A, E)}$.

Pending Cost: $\mathcal{P}^s(\text{ALG}, \mathcal{A}, \mathcal{E}) = \sup_{A\in\mathcal{A}, E\in\mathcal{E}, X\in\mathcal{X}} \lim_{t\to\infty} \frac{P_t^s(\text{ALG}, A, E)}{P_t^1(X, A, E)}$.

Latency: $\mathcal{L}^s(\text{ALG}, \mathcal{A}, \mathcal{E}) = \sup_{A\in\mathcal{A}, E\in\mathcal{E}, X\in\mathcal{X}} \lim_{t\to\infty} \frac{L_t^s(\text{ALG}, A, E)}{L_t^1(X, A, E)}$.

Contributions. Table 1 summarizes the results we have obtained for the four algorithms. Although not distinguished, the first results we showed apply to *any work conserving scheduling algorithm*, one that does not idle as long as there are pending tasks and does not break the execution of a task (unless the machine crashes): (a) When $s \geq \rho = \frac{c_{max}}{c_{min}}$, the asymptotic completed cost ratio is lower bounded by $1/\rho$ and the asymptotic pending cost ratio is upper bounded by ρ. (b) When $s \geq 1 + \rho$, the asymptotic completed cost ratio is lower bounded by 1 and the asymptotic pending cost ratio is upper bounded by 1 (i.e., they are 1-competitive).

Then, for specific cases, we obtain better results for different algorithms. However, it is clear that none of the algorithms is better than the rest. With the exception of SPT, no algorithm is competitive in any of the three metrics considered, in the case when $s < \rho$. Algorithm SPT is competitive only in terms of asymptotic completed cost. For $s \geq \rho$ only algorithm LIS is competitive in terms of asymptotic latency. This however, might not be surprising as algorithm LIS gives priority to the tasks that have been the longest in the system. Another interesting observation is that algorithms LPT and SPT become 1-competitive in terms of completed and pending cost, for $s \geq \rho$, whereas LIS and SIS require greater speedup to achieve this. In some sense, these results collectively demonstrate clearly the differences between these two classes of policies: the ones that give priority based on the task *arrival time* (LIS and SIS) and the ones that give priority based on the task *cost* (LPT and SPT). Observe also, that different algorithms scale differently with respect to the speedup, in the sense that, with the increase of the machine speed the asymptotic competitive performance of each algorithm changes in a different way for each policy.

Related Work. We relate our work with the online version of the *bin packing* problem, where the objects to be packed are the tasks and the bins are the time periods between two machine failures. Some of the research that has taken place over the years around this problem, we consider related to ours [7, 5, 11]. However, the essential difference of the online bin packing problem with the one that we are considering, is that in our system the bins and their sizes (corresponding to the intervals between consecutive failures) are unknown. On a different tone, Boyar and Ellen [4] have looked into a problem similar to both online bin packing problem and ours, considering job scheduling in the grid. The main differences with our setting is that they consider several machines, but mainly that the arriving items are processors, which have limited memory capacities, and there are some fixed jobs in the system that must be completed. They use fixed job sizes and achieve lower and upper bounds that only depend on the fraction of such jobs in the system. On a previous work [1], we considered online packet scheduling, where packets of two different sizes were scheduled through an unreliable link. In that work, the the goodness metrics is the long-term competitive ratio, which is called *relative throughput*. Online algorithms as well as bounds for the proposed metric for any online scheduling protocol are then given there. We can also directly relate our work with research done on machine scheduling with availability constraints (e.g., [10, 6]), one of its most important results being the necessity of online algorithms in case of unexpected machine breakdowns. However, in most works on this topic, preemptive scheduling is considered and optimality is shown only for nearly on-line algorithms (need to know the time of the next job or machine availability). The study in this paper was triggered by our previous work [2], in which we assumed multiple processors and distributed scheduling algorithms. That paper unveiled the need of a systematic study of more basic and fundamental models and protocols.

Discussion. Although our study focused on the simple setting of one machine, we believe that interesting conclusions have been derived with respect to the efficiency of these algorithms under the three different metrics (completed cost, pending cost and latency) and under different speedup. An interesting open question is whether one can obtain efficiency bounds as functions of speedup s, that would give upper bounds for the completed cost and lower bounds for the pending cost and latency competitive ratios. A natural next step would also be to extend our investigation to the setting with multiple machines.

References

[1] Antonio Fernández Anta, Chryssis Georgiou, Dariusz R. Kowalski, Joerg Widmer, and Elli Zavou. Measuring the impact of adversarial errors on packet scheduling strategies. In *SIROCCO*, pages 261–273, 2013.

[2] Antonio Fernández Anta, Chryssis Georgiou, Dariusz R. Kowalski, and Elli Zavou. Online parallel scheduling of non-uniform tasks: Trading failures for energy. In *FCT*, pages 145–158, 2013.

[3] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, New York, NY, USA, 1998.

[4] Joan Boyar and Faith Ellen. Bounds for scheduling jobs on grid processors. In Andrej Brodnik, Alejandro Lpez-Ortiz, Venkatesh Raman, and Alfredo Viola, editors, *Space-Efficient Data Structures, Streams, and Algorithms*, volume 8066 of *Lecture Notes in Computer Science*, pages 12–26. Springer Berlin Heidelberg, 2013.

[5] Leah Epstein and Rob van Stee. Online bin packing with resource augmentation. *Discrete Optimization*, 4(34):322 – 333, 2007.

[6] Anis Gharbi and Mohamed Haouari. Optimal parallel machines scheduling with availability constraints. *Discrete Applied Mathematics*, 148(1):63 – 87, 2005.

[7] D. Johnson, A. Demers, J. Ullman, M. Garey, and R. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3(4):299–325, 1974.

[8] Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM (JACM)*, 47(4):617–643, 2000.

[9] Kirk Pruhs, Jiri Sgall, and Eric Torng. Online scheduling. *Handbook of scheduling: algorithms, models, and performance analysis*, pages 15–1, 2004.

[10] Eric Sanlaville and Gnter Schmidt. Machine scheduling with availability constraints. *Acta Informatica*, 35(9):795–811, 1998.

[11] Rob Van Stee. *On-line Scheduling and Bin Packing*. 2002.