Design and Comparison of Resilient Scheduling Heuristics for Parallel Jobs

Anne Benoit[1], Valentin Le Fèvre[1], Padma Raghavan[2], Yves Robert[1,3], Hongyang Sun[2]

1. Laboratoire LIP, ENS Lyon, France
2. Vanderbilt University, USA
3. University of Tennessee Knoxville, USA

hongyang.sun@vanderbilt.edu

22nd Workshop on Advances in Parallel and Distributed Computational Models (APDCM'20)

What Is This Paper About?

On large-scale HPC platforms:

- Scheduling parallel jobs is important to improve application performance and system utilization;
- Handling job failures is critical as failure/error rates increase dramatically with size of system.

This paper combines job scheduling and failure handling for parallel jobs running on large HPC platforms prone to failures.

Scheduling Models

Job Model:

- n rigid parallel jobs all released at time 0 (i.e., batched);
- each job has a processor request p_j and an execution time t_j;
- Jobs are to be scheduled on a set of P identical processors.

Error Model:

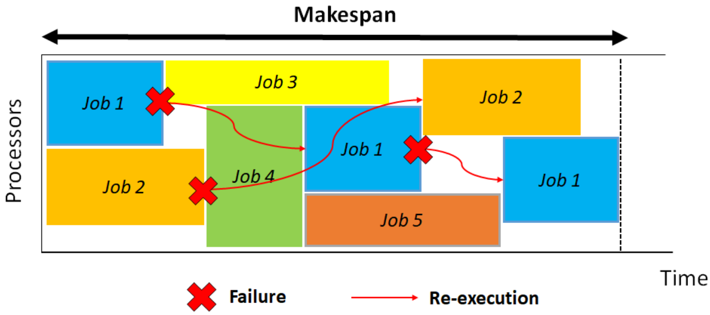
- Jobs are subject to silent errors (or silent data corruptions);
- Silent error detector (of negligible cost) is available to flag errors at the end of each job's execution;
- If a job is hit by silent errors, it must be re-executed (possibly multiple times) till successful completion;

Objective:

- Minimize the makespan (i.e., successful completion time of all jobs);
- Number of failures for each job is unknown a priori;
- No assumption on error rate or distribution.

A failure scenario $\mathbf{f} = (f_1, f_2, \ldots, f_n)$ describes the number of failures each job experiences during a particular execution.

Example: $\mathbf{f} = (2, 1, 0, 0, 0)$ *for an execution of five jobs.*

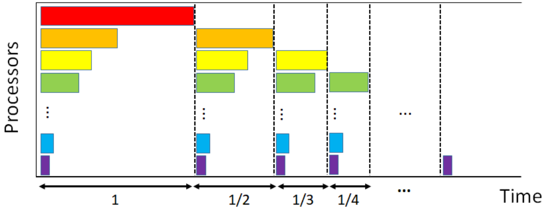


Main Results

1. A resilient list-based scheduling algorithm, and $O(1)$-approximations for any failure scenario:
   - 2-approximation using Greedy heuristic without reservation;
   - 3-approximation using Large Job First priority with reservation.

   The results nicely extend the ones without job failures [TWY'92].

2. A resilient shelf-based scheduling heuristic, but $\Omega(\log P)$-approx. for any shelf-based solution in some failure scenario, e.g.:



   The result defies the $O(1)$-approx. result without failures [TWY'92].

3. Extensive simulation results of all heuristics using both synthetic jobs and job traces from the Mira supercomputer.

Relations with Similar Models

Offline/online scheduling with job release times [NS'02, Johannes'06]:

- In this model, jobs have fixed release times ⇒ 2-approx. for list.
- In our model, "new job releases" (corresponding to failed jobs restarting) depend on scheduling decisions ⇒ 2-approx. for list.

Online one-by-one scheduling of parallel jobs [HP'08, YHZ'09]:

- In this model, independent jobs must be scheduled one-by-one without future knowledge ⇒ $O(1)$-approx. for shelf.
- In our model, no immediate scheduling is required, but (failed) jobs form dependencies ⇒ $\Omega(\log P)$-approx. for shelf.

Offline/online scheduling with general dependencies [FKST'98, Li'99]:

- In this model, jobs form a known DAG ⇒ $\Theta(P)$-approx. for list.
- Our model is a special online case with n linear chains, each having an unknown number of identical jobs ⇒ 2-approx. for list.

Open Questions

Q1. Results for expected makespan when assuming a probability distribution (e.g., exponential) for job failures.

- *In particular, does shelf-based scheduling admit an $O(1)$-approx. for expected makespan under exponential failure distribution?*

Q2. Results for more flexible job models, such as moldable jobs, whose processor allocations need to be decided before execution.

- *Our latest work[1] proves approximation ratios for several speedup profiles, but bounding expected makespan remains an open question.*

[1] "Scheduling moldable jobs on failure-prone platforms", coming soon...

References

Our complete paper is also available as an INRIA research report (RR9296) at:
`https://hal.inria.fr/hal-02317464v2`

Other references:

- J. Turek, J. L. Wolf, and P. S. Yu. Approximate algorithms scheduling parallelizable tasks. In ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), 1992.
- E. Naroska and U. Schwiegelshohn. On an on-line scheduling problem for parallel jobs. Information Processing Letters, 81(6):297–304, 2002.
- B. Johannes. Scheduling parallel jobs to minimize the makespan. Journal of Scheduling, 9(5):433–452, 2006.
- J. L. Hurink and J. J. Paulus. Online algorithm for parallel job scheduling and strip packing. In Workshop on Approximation and Online Algorithms (WAOA), 2008
- D. Ye, X. Han, and G. Zhang. A note on online strip packing. Journal of Combinatorial Optimization, 17(4):417–423, 2009.
- A. Feldmann, M.-Y. Kao, J. Sgall, and S.-H. Teng. Optimal on-line scheduling of parallel jobs with dependencies. Journal of Combinatorial Optimization, 1(4):393–411, 1998.
- K. Li. Analysis of the list scheduling algorithm for precedence con- strained parallel tasks. Journal of Combinatorial Optimization, 3(1):73– 88, 1999.