An Exploration of Online-simulation-driven Portfolio Scheduling in Workflow Management Systems

Supplementary Material

Abstract

This document provides supplementary material for Section 5.4 in "An Exploration of Online-simulation-driven Portfolio Scheduling in Workflow Management Systems" by McDonald et al. (FGCS 2024).

1 Adapting to Dynamic Platform Changes

To conduct experiments for dynamic platform behaviors, we have augmented our experimental framework with two parameters that define a simple dynamic resource availability model:

- A probability p: a compute core is not available at the onset of the execution with probability p;
- A time T: an initially unavailable compute core will become available after a number of seconds sampled from a uniform distribution over the $(0, T)$ interval.

This is just one of the many possible ways to generate a dynamically evolving resource availability pattern, and likely different models are needed for different real-world use cases. But this admittedly simple model allows us to run experiments to evaluate the effectiveness of SDPS for adapting to platform changes.

We ran experiments for:

- $p \in \{0.1, 0.2, \ldots, 0.9\}$; and
- $T \in \{\alpha \times T' | \alpha \in \{0.1, 0.5, 1.0, 2.0, 10\}\}$, where T' is the makespan achieved using algorithm A_8 when all resources are available from the get go.

We use the above scaling approach for T values because our workflows all lead to very different makespans, making picking absolute values for T problematic (e.g., for a workflow that takes 3600s a T value of 100s means that most resources are available for most of the executions, but this is not the case for a workflow that takes 600s). We conducted experiments for platform configuration P_3 (the most heterogeneous) and assuming no simulation error (to ensure that we observe the net effect of adapting to platform changes). This leads to 405 experiments (9 workflows, 9 values of p, 5 values of α), for each of which we ran 100 trials (different seeds of the random number generator used to generate the initial machine state). For each trial we compare the one-algorithm solution (algorithm A_8, a.k.a., "one-alg") with SDPS using a single round of simulation ("no-adapt") and SDPS using a round of adaptation each time an additional 10% of the compute work has been completed ("adapt").

We first show and discuss results for two workflows and then discuss broad observations over all results.

Results for the W_1 workflow – Figure 1 shows results with the W_1 workflow for different values of p and α. Figure 1a is a "control" experiment where all resources are always available ($p = 0$). The

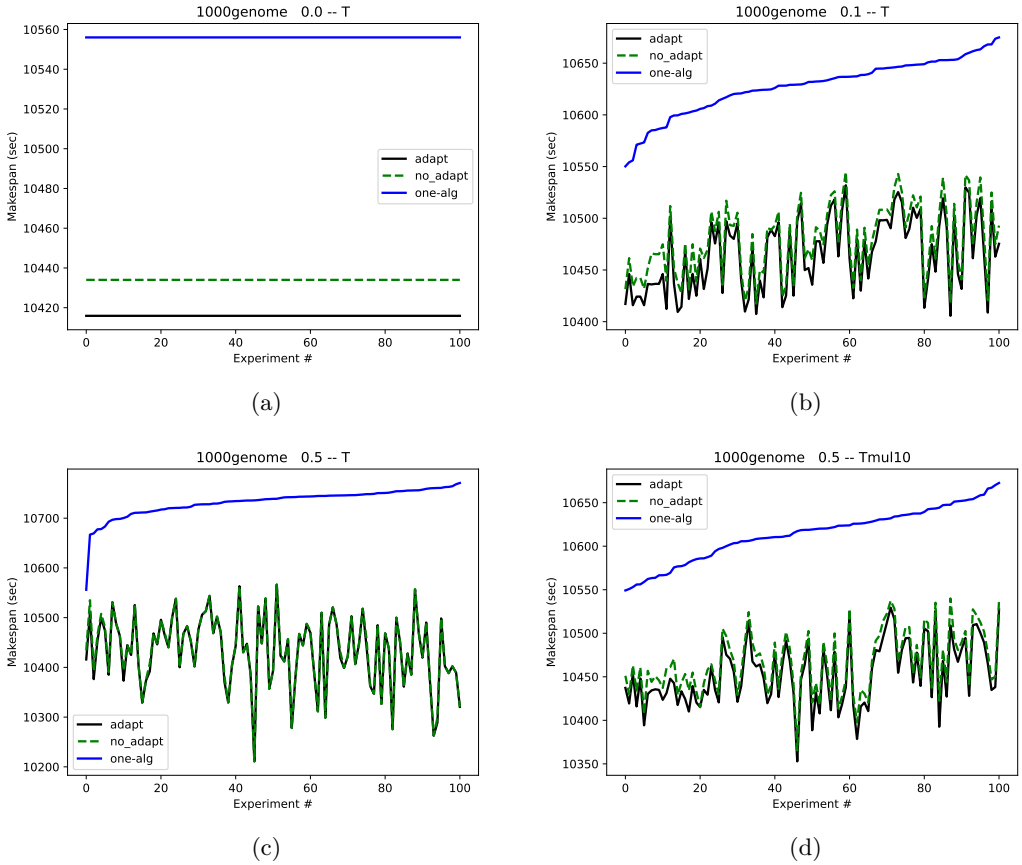

Figure 1: Makespan results for the W_1 workflow. The first data point on the horizontal axis is always for $p = 0$. Each subsequent data point it for a different random trials. Data points are sorted by increasing makespan as achieved by algorithm A_8.

vertical axis is the achieved makespan, and the horizontal axis correspond to 100 random trials. In this figure, since $p = 0$, all trials are identical (all compute resources are available) hence the flat lines. We see that effect that adaptation can bring some benefit even when everything is static, due to the workflow proceeding in different phases with different characteristics (see Section **??**). Figure 1b is for $p = 0.1$ and $\alpha = 1$. In this and other figures, data points are sorted along the horizontal axis with respect to the makespan achieved by the one-algorithm approach ("one-alg"). For reference, the leftmost data point is always for $p = 0$. We now see variations, and marginal improvements due to running multiple rounds of adaptation. This improvement is a combination of selecting a new algorithm because the resource pool has changed and the above effect due to the workflow proceeding in different phases. Figure 1c is for $p = 0.5$ and $\alpha = 1$, and here there are almost no improvements for "adapt" over "no-adapt". Finally, Figure 1d shows results for $p = 0.5$ and $\alpha = 10$, with more improvement of "adapt" over "no-adapt", but still marginal. All other results for different p and α values look very similar. Overall, for this workflow, running multiple rounds of simulations does not have a large effect. This is because the structure of the workflow renders initial scheduling decisions crucial. When comparing SDPS to "one-alg" in these results, a case can be made, as the reviewer thought, that the one-algorithm approach is more sensitive to the initial resource states. This is a strength of SDPS over the traditional approach, but the use of multiple rounds of simulation does not bring much benefit.

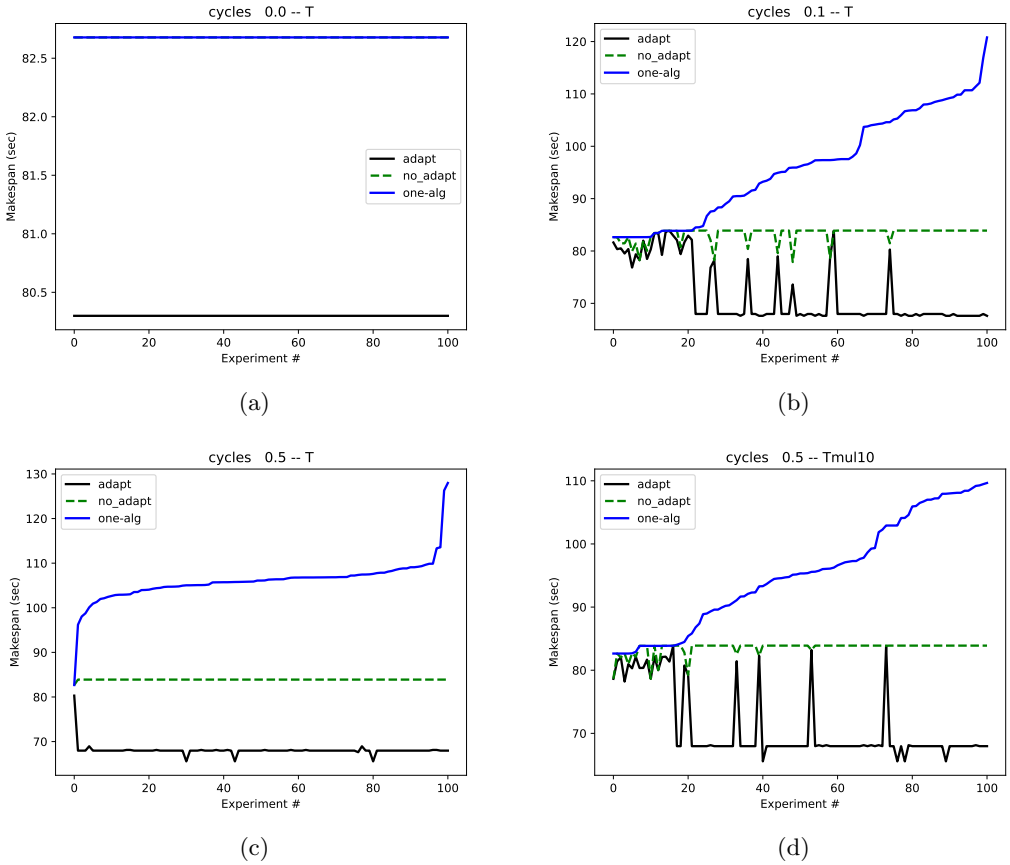

Figure 2: Makespan results for the W_4 workflow. The first data point on the horizontal axis is always for $p = 0$. Each subsequent data point it for a different random trials. Data points are sorted by increasing makespan as achieved by algorithm A_8.

Results for the W_4 workflow – Figure 2 is similar to Figure 1, but for workflow W_4. For this workflow, we see that using multiple rounds of simulation with $p = 0$ does not bring any significant benefit (Figure 2a). The other results show that, unlike for workflow W_1, using multiple rounds of simulations can bring benefits ($\sim$17% makespan reduction). Furthermore, it is clear that the performance achieved is more stable than, in particular, the traditional OneAlg approach.

Overall results – Overall, although results in Figure 2 show that running multiple rounds of adaptation can bring some benefit, this behavior is not seen for most workflows. For instance, out of our 9 workflows, for $p = 0.5$ and $\alpha = 1$, significant improvements are seen for only 1 workflow (W_4). For $p = 0.5$ and $\alpha = 10$, significant improvements are seen for that same workflow and only minor improvements are seen for another workflow (W_5). Out of all our results, the use of multiple rounds of simulation leads to significant improvements in only a handful of cases. There are likely real-world workflow configurations for which the improvement could be large. We could easily craft synthetic platform configurations, initial availability patterns, and workflow configurations specifically so that improvements are large. But, within the scope of our study of 9 real-world workflow configurations and a typical real-world platform architecture, large improvements are rare at best.