RM-Replay for Cluster Scheduling Project Report 3/27-4/15

Zhen Huang, Blake Ehrenbeck

Our goal these past two weeks have been verifying that the wait times we get back from RM_Replay are in line with what we expect them to be.

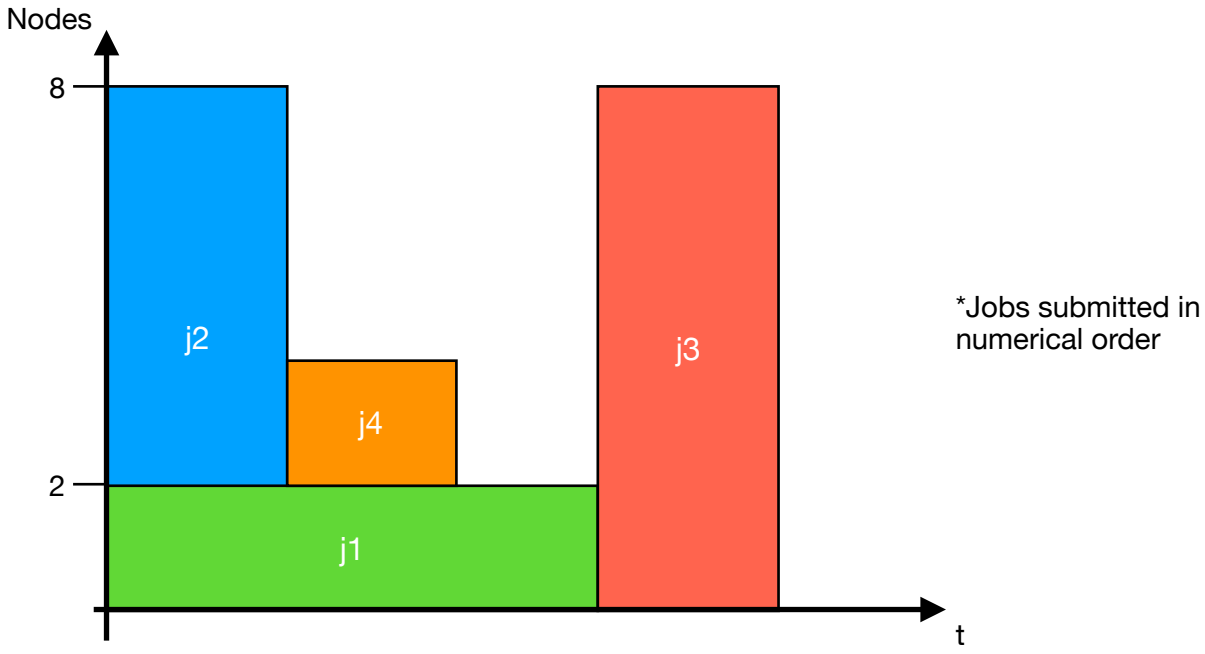
Progress

1. A Simple Test Case

We first tried a very simple test case. We ran a single job across three nodes on a machine with no currently pending or running jobs. We expected the output to be 0 seconds. When running with a replay ratio of 0.0001 we found that the wait time was 4 seconds. We learned that this is plausible considering there may be deviations in the output if the ratio is not one to one. Confirming this, once we set the ratio to be 1 replay second for every 1 second (1:1), we got a wait time of 0.5 seconds. This we believe is plausible. To verify this, we went back to the machine we originally ran the job on, and looked at the Slurm database that has records of the job runs. Since the granularity of this data is in seconds, we only know that the job was submitted and started within 1 second, making a time of 0.5 seconds a possible output.

2. A More Complicated Case

We also needed to try a more complicated test case. It's best illustrated with a graph:



We can expect that if the ratio is of replayed seconds to actual seconds is 1:1, j2 and j1 will experience no wait, j4 will actually start before j3 because of backfilling and will only wait for j2 to complete. j3 will start after j1 completes.

Initially, we were getting nonsensical wait times. As it turns out RM-Replay was deciding what nodes to allocate versus the nodes we asked for in our jobs via the nodelist. After combing through the source of RM-Replay we were able to find a flag that forces RM-Replay to listen to your node list.

We supplied -p 3 to ./start_replay.sh and it successfully used the nodes we wanted.

The output then was consistent with what we expected given the scenario in the graph pictured above.