Estimating an HPC Facility's Capacity For Accommodating Realtime Workflows

Eashan Adhikarla[1], Brian Austin[2]
[1]National Energy Research Scientific Computing (NERSC), [2]Lehigh University

Introduction

Supercomputers are valuable, highly contended resources that are typically managed by batch queue systems.

A growing number of workloads (including tightly-integrated experimental facilities and emergency decision making) require response-times that are not compatible with typical queue wait-times.

Any policy to accommodate real-time requirements will impact system utilization and disrupt other users.

This project estimates the consequences of three potential real-time scheduling policies by analyzing job logs from NERSC's Cori Supercomputer.

Methodology

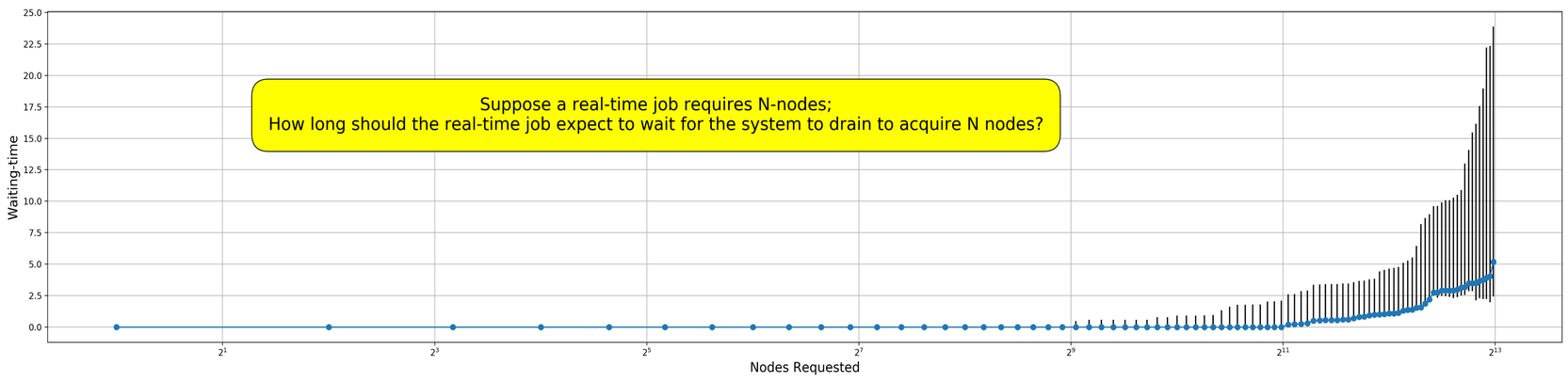
We obtained from NERSC, a log of the node count, start time, end time and memory footprint for all jobs that ran on Cori's KNL nodes during 2018. We randomly sampled 50-timepoints that correspond to the submission times for a set of hypothetical real-time jobs. The job logs were then analyzed to compute median wait-time, utilization-loss or memory availability for each of the proposed scheduling policies described below.

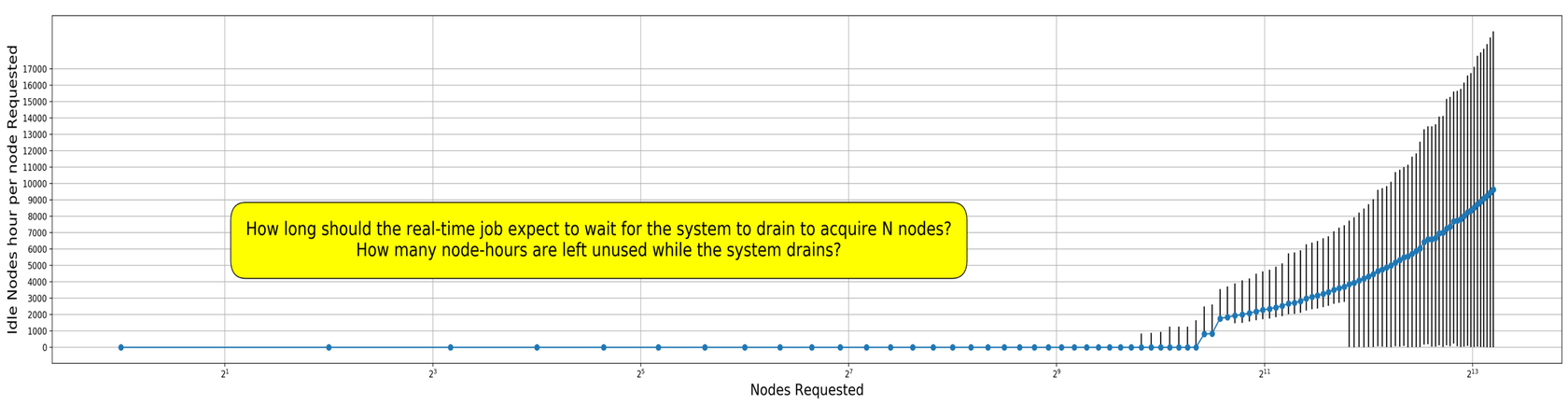
Cori and its Workload at a Glance

Compute Nodes	9688 (KNL), 2388 (Haswell)
Jobs per day	60,000
Nodes per Job	1-9688, Avg.= 716 hours
Time per Job	<1min – 8 days, Avg.=19 hours

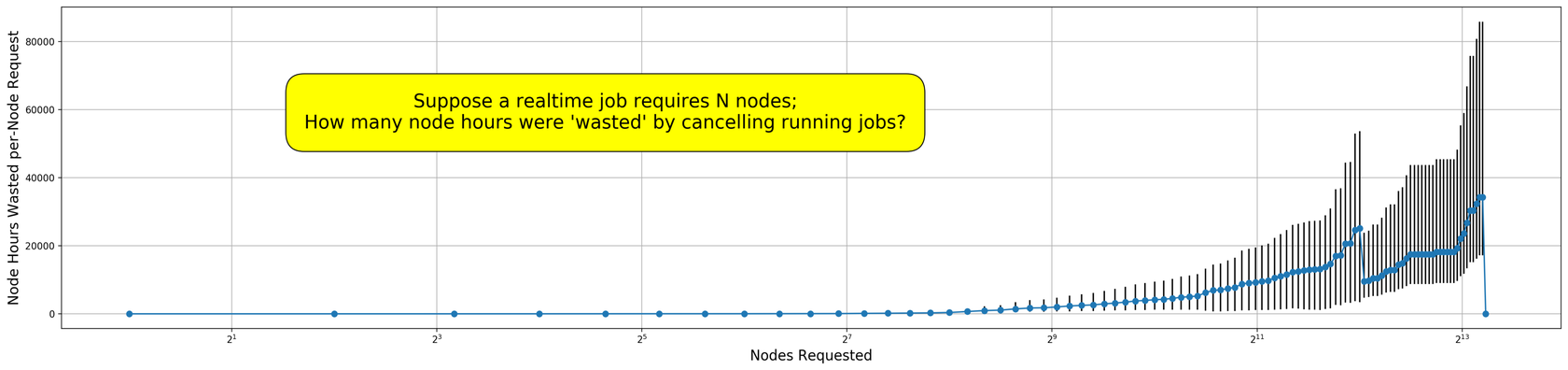
Results

Draining: Upon submission of a real-time job, the compute nodes are drained by allowing currently running batch jobs to complete normally and prohibiting batch jobs from starting until the real-time job begins.

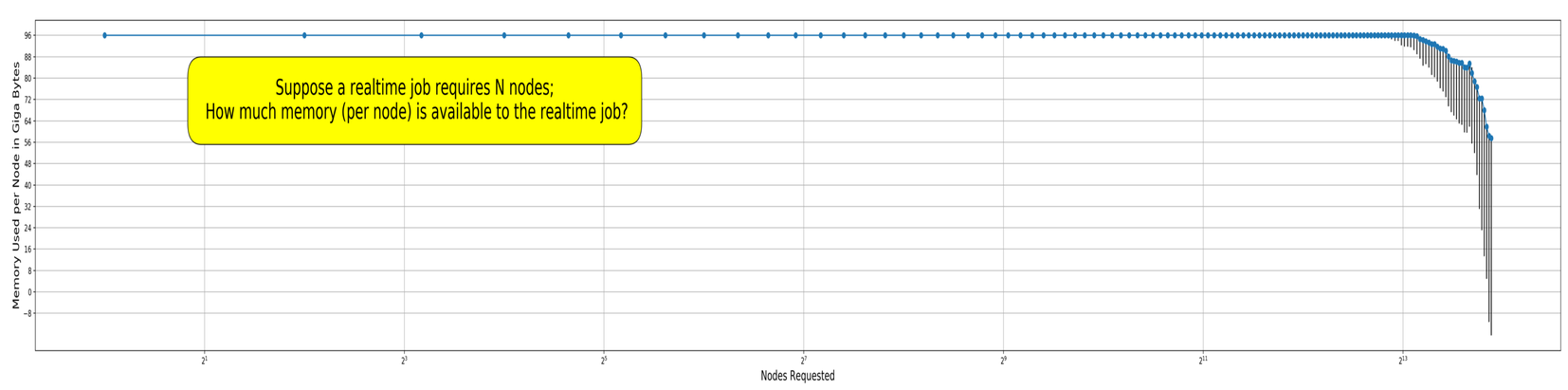




Job Cancellation: Upon submission of a real-time job, a fraction of the currently running batch jobs are cancelled to allow the real-time job to start immediately. Jobs are selected for cancellation to minimize waste of computational resources.
This policy is extremely disruptive to users whose jobs are cancelled.



Job Pausing: Upon submission of a real-time job, a fraction of the currently running batch jobs are paused (using, for example, the `nice` command) to allow the real-time job to start immediately. The paused jobs maintain their memory footprint, so jobs are selected for pausing to maximize the memory available to the real-time job.



Conclusions

Because HPC resources are finite, no scheduling policy can provide sufficient compute elasticity to run arbitrarily large jobs on demand. All the policies evaluated here "passes the buck" in a different direction. System-draining delays the start of the real-time job and squanders HPC resources with idle nodes. Job-cancellation gives sufficient priority to real-time jobs, but at the expense of severe disruption to other batch users whose jobs are cancelled. "Pausing" batch jobs is a promising option that allows real-time jobs to start immediately and avoids wasting system resources but requires more elaborate changes to the scheduler and may increase memory pressure, especially for high-concurrency real-time jobs.

Future Work

- Introduce supervised machine learning model to the set of policies so that the model can make internally decisions on which policy to be allotted to the incoming job.
- Improve these results by accounting Backfill.

References

- Time-sharing redux for large-scale HPC systems : ieeexplore.ieee.org/document/7828392
- NERSC-2014 Workload Analysis : http://portal.nersc.gov/project/mpccc/baustin/nersc2014workloadanalysis_v1.1

Contact: eadhikarla@lbl.gov |eaa418@lehigh.edu
Lehigh University, Mountain Top 113 Building C
(650) 695-9272