Loading of processing nodes ‘near full’

Work in progress update: Rubin’s use of PanDA

Rubin has been doing some ‘operations rehearsal’ testing recently, where we find that when we load a compute node near capacity, things slow down, sometimes by factors of 2x to 4x in terms of ‘wall clock’ processing.

We think this is because of contention for I/O on an individual node, and we are wondering if other PanDA users have had experience with this, and some things they might have done to more effectively use all available processing cores.

Example: Operations Rehearsal scale up test: 72 nodes (Linux boxes), each with 512 GB RAM, 128 cores (with 8 reserved, so 120 cores available for science jobs): 72*120 = 8640 cores.

SLURM is used to manage the 72 nodes, PanDA feeds the jobs to the SLURM managed cluster.

The default behavior is to ask for one core per ‘payload’ (science job), and if we have many jobs to submit, (more than 120), the way that the nodes are filled appears to be: 120 sent to the first node (or lowest load node at the time of job submission), until it is full, then onto the next node, and so on.

This results in a few nodes full to capacity, and many nodes with little or much less running on them.
When this happens, we observe that the processing time (wall time) of the ‘full-to-capacity’ nodes is 2x-4x slower (per job) than the nodes which are less full (¼ full or less).

Thus: If we have 800 jobs to run on 8640 cores on 72 nodes,
We are seeing 100 jobs on three nodes and 50 or 20 or 10 jobs spread over 20 or more nodes.

The jobs on the 10, 20 jobs per node systems run in a (good) time of two hours per jobs.

The jobs on the 100 jobs per node systems run in a (poor) time of four to eight hours per job -

Thus time end-to-end time to finish all the jobs ends up being eight hours, when it should be possible to do it in two hours, given the amount of nodes available.

Our question for other PanDA users is: Have you dealt with similar situations
Your self, and how?

One thing we are considering: Suggesting to SLURM that each job takes up 4 cores so that it only fills a node to 120/4 = 30 jobs/node before moving on to the next.

However, how then do we make use of these ‘empty cores’ ¾ on each node which might sit unused?

We could perhaps fill with ‘lower I/O intensive, more cpu intensive jobs’ where we set 1 core per jobs and let the machine be filled with these.

Do people have other suggestions?

Is there a way to ‘consider all 72 nodes = 8640 cores’ as a ‘large pool’ and fill nodes horizontally rather than vertically, i.e. fill 10 jobs on node 1, 10 on node 2 up to node 72 , then go back to node 1 and send 10 more jobs to node 1, etc.

In this case, if there are << 8640 jobs to run, the jobs would be more evenly distributed across the cluster.