Plans & Progress Report on Bullet cluster batch configuration

Intro

The bullet cluster has 2928 cores (183*16) that are to be shared by jobs from multiple queues and projects. Atlas and Fermi are the biggest users of single slot jobs. KIPAC and theory folks run parallel jobs at several sizes and time-scales. Crafting an efficient way to share the machine between the single slot and parallel jobs is the goal of this plan. The single slot (ss) jobs should run as unlimited as possible when there is no parallel work and they should leave as much room as possible for parallel jobs. Parallel jobs should be able to aquire hosts and run in a timely fashion. Generally parallel jobs work best using whole hosts. Small "development test" parallel jobs generally need quick access and larger runs can wait a correspondingly longer period.

This plan was initially developed during a meeting on 31 October attended by most of the stake holders (see Shirley's notes). The knobs to turn are considered below along with comments on implementation and a summary report of progress so far is given. Finally an attempt is made to measure success as we proceed.

LSF configuration tuning

At present, we have identified the following areas to tune batch performance.

1.) Job dispatch distribution

LSF defaults to disperse jobs widely across available hosts. The opposite is desired for ss jobs. Changes to affect this so far are setting the order[] parameter to "order[-slots,-maxslots]" and JOB_ACCEPT_INTERVAL=0. The first causes ss jobs to aggregate onto the smallest (#cores) available host with the fewest unused cores. In our system this means the older smaller hosts fill with jobs first in an unloaded system. The second causes jobs to aggregate in time with consecutive jobs being assigned to the same host if there are unused cores remaining on that host. Initial tests on the orange cluster are promising. This works very well as long as the cluster is not full. Once full there is no choice but for jobs to get more widely dispersed. Keeping some "head room" seems to help.

2.) Establish the use of backfill

Backfill refers to running time limited jobs on cores that are reserved for a larger job that is not yet able to run. This keeps the utilization of the cluster high. Backfill is enabled and tested on all the queues. It is not yet in widespread use as it requires planning by the users.

3.) Set up limits on running jobs per queue on bulletfarm hosts

LSF can be configured to limit the total number of slots used collectively by the general queues on bullet nodes. This has not yet been tried but it is up next. In particular allowing a maximum of ~2500 jobs from non-parallel queues would effectively provide a minimum set of cores to MPI jobs. Similarly we may limit the number of slots used in the mpi queue.

4.) Parameters that affect the reservation of cores for parallel jobs

Ideally the reservation system should collect cores and converge to a set of machines meeting the job requirements in a mostly linear fashion. So far that is not the case. Parameters that affect this are being investigated. Some help from IBM may be appropriate here.

5.) cpu/memory/network binding using cpu affinity, cgroups limits

A medium term goal is to limit jobs to the number of cores and amount of memory or network that they requested. There are modern capabilties now that we are not yet using.

6.) Job migration:

Virtual machines for long running jobs and triggers for migration could be used to move jobs off of low occupancy hosts and onto higher occupancy hosts to free up more hosts for parallel jobs. This is a medium term goal and requires significant local tooling effort. It is effort appropriate for a modern scientific data center and will follow naturally from existing use of vm's by CD now for smaller projects.

7.) Lessor forms of partition

Advanced or guaranteed reservations (also see #3 above) are being considered.

8.) re-architect the whole LSF setup

A ground up "redo" is appropriate but not with out first understanding the settings that would define the new set up. The current slac system has evolved and is not really "designed". A formal design effort is warranted after initial quick fixes.

9.) re-architect with another batch system such as SLURM

This should be considered also before or with #8.

Summary of Progress

At this time (5 November) only (1) and (2) above are set up. The job dispatch changes are working well when the cluster is not completely full (~few hosts empty). After that new jobs replace finished jobs in semi-random fashion and fragment in time. Backfill is expected to be very useful in future but is not being enabled at submission time by users yet. It is potentially useful for Atlas and for smaller parallel jobs that are quick turn around tests of new codes.

Job count limits on general queues are likely to be the next test as well as changes to parameters that may affect the reservation of cores for parallel jobs.

Beyond LSF itself, it may be necessary to provide ways for user jobs to be "bundled" into units that match full hosts (16 cores in this case). This would then put the bundled single slot jobs on par with the parallel and keep the cluster occupancy pattern consistent across all the queues. This has been reported to work at other institutions.

The orange and Pinto clusters will be used as test beds for many of these ideas including the job migration. Given the budget environment we expect to keep the orange cluster going as long as the maintenance effort is not a burden.

Measuring success

Success is expected to be incremental and user dependent. Job statistics can be prepared from the LSF logs and we can chart the throughput and latency for various classes of parallel jobs. Interpreting this will require input from the parallel job users. Information is being gathered now for this purpose. A review of these statistical measures can be made in early December.

strawman for success metric:

- throughput: ~1000 parallel slots in use when demand is sufficient
- latency: 1-2 hours for <=128 cores; <1day for <=512 cores; <2days for ~1024 cores (TBD if special arrangements needed)
- runtime limits: if you wait 2 days to run and only get a short time the duty cycle is poor so some thought needed here