Getting the Most from Sharcnet

plan

- processors/nodes
- interconnect
- storage
- scheduler

Brief review

- Many clusters, spanning 8 years of tech
- Some intentionally specialized
- Some contributed
- No two identical:
  - cpu/memory node config
  - cluster interconnect and storage
  - connectivity to the WAN

Node hardware

- CPU models: instruction set and clock:
  - 2 DP flops/cycle (Requin)
  - 4 Saw, Orca
  - 16 Haswell
- Memory configuration (bandwidth, capacity)
  - 4*PC3200 (Requin, 3.2 GB/s/core)
  - 4*PC5300 (Saw, 2.7)
  - 8*PC10666 (Orca, 3.5)
  - 8*PC15000 (Haswell, 5.0)

Memory

- All systems except Saw are NUMA
- This means that a big-memory serial job will never be as fast as possible
- Ideally, use the amount of memory attached to the number of CPU die you’re using
- Approximated by memory-per-core

Node Interconnect

- Requin: 1GB/s, 1.4 us, full bisection
- Saw: 2.5 GB/s, 2 us, partial
- Orca: 5 GB/s, 2 us, partial

Storage

- Metadata
  - many/small files
  - file creation/rename/delete
  - colorized ls is expensive
- Large IO
  - fewer, big files
  - good for Lustre
  - blocks should be megabytes
  - global work and orca/saw scratch

Storage

- Please don’t use /home
  - very nice NFS servers but terrible for bandwidth
  - OK for metadata (compiling)
- Global work
    - terrible for anything but big-file-IO
    - striping
- Scratch
  - much faster normally
  - because it’s less contended

Node-local storage

- Don’t forget that nodes have their own disks
- Not very convenient
- Scales ideally

Storage Performance Numbers

- Lustre is about 150 MB/s per OSS
- local /scratch scales well
- global work not so well
  - interference
  - latency

Into and Out of Sharcnet

- sshfs
- DTN
  - globus: gridftp
  - rsync

Scheduling

- Jobs request CPU and memory resources for a specific length of time (GPU too).
- Scheduler chooses node where these can be provided exclusively.
- This means that all resources are conflated and mutually affect how soon a job may be scheduled

Scheduling

- Minimize your memory request
- Minimize your CPU request
- Minimize your runtime request
- Don’t turn a bunch of serial work into MPI

Scheduling

- Every cycle, jobs are examined to see which ones can be started.
- Order depends on priority, which is mainly NRAC > normal
- Fairshare is considered a goal: some priority advantage if you are far from your usage.
- Greedy: whether your job starts depends mostly on when/whether resources become available.