Overlall themes

- Current maintenance
  - Cedar down for system software updates (today only)
  - `/project` expansion (`/project` unavailable March 1-4); you can still use `/scratch` for running jobs during this time
- Our goals are to:
  - provide as much uptime as possible
  - have all resources (CPUs, GPUs, memory, storage, bandwidth) to be utilized as much as possible
  - minimize gaps in scheduling between jobs
  - minimize turnaround for your jobs
  - provide all information in the system status (in case of downtime)
- When hardware/etc problems occur, we want you to know how you can work around them
- We want to show you how certain workflows can lead to system instabilities

Major causes of system instability

- Node failures: a node needs rebooting
- File system problems: Lustre object storage servers can get overloaded with lots of small requests (more on this later); on Cedar we have
  - 4 object storage servers handling `/home` (slow) and `/scratch` (fast)
  - 10 object storage servers handling `/project`
  - these are paired into groups of two
    - one in a pair goes down $\Rightarrow$ the other one will take over
    - both go down $\Rightarrow$ the entire filesystem will hang
- Scheduler failures
  - Slurm can get overloaded with too many requests (more on this later)
- Oversubscription of nodes, GPUs
- No software stack synchronization between login and compute nodes
- Networking problems (within or outside our control)

What do you see?

- Sluggish jobs
- Jobs not starting / taking unusually long to start
  - also valid reasons why your job's starting time can be pushed into the future
- Slurm not responding, or producing unusual output
  - e.g. a job stuck in 'Prolog' R (running) state for a long time, not producing any output
- Shell not responding to simple commands or very slow
- Output files missing from your working directory
- Inside running jobs see "module not found"
  - typically requires manual intervention
- Cannot log in

What can you do about these instabilities?

- Report problems to `support@computecanada.ca` with details:
  - system you are using
  - job IDs of affected jobs
  - detailed description of the problem
  - time/date it was first encountered
  - full path to one of the directories with the script and error files
    - verify if you signed the consent that allow analysts to check your files (this will help resolve problems quickly instead of exchanging many emails)
- Pay attention to login messages
  - terminal output from anything in your `~/.bash_profile` or `~/.bashrc` (e.g. when loading a module or activating a virtual environment) might force important system messages scroll past the top of the terminal
  - these may contain both general system notices and `/scratch` purge notifications specifically for you
- Check `http://status.computecanada.ca` for updates and recent incidents

What can you do? (cont.)

- Sometimes you could work around a temporary filesystem problem by submitting jobs from another filesystem
  - on Cedar `/home`,`/scratch` files are handled by different servers than `/project` (may not be always possible: performance)
- Do *not* delete and resubmit jobs that have been waiting in a queue for a long time until confirming with `support@computecanada.ca`
  - otherwise we can't analyze why a job is waiting, and priority may be lost
- Expect a backlog of jobs after a system problem
  - do *not* swamp the system with a bunch of new jobs – be selective about what is most important to you
  - make sure that job parameters are chosen carefully to match the needs of particular jobs

These workflows will create problems

- Running anything CPU-intensive on the head node
- Submitting large number of jobs
- Issuing too many requests to the scheduler
  - classical example: running `watch squeue ...`
  - submitting thousands of jobs and then cancelling them
- Complex/unrealistic job dependencies can make Slurm unstable
- Not testing first on a small scale (and gradual scaling up)
  - large parallel jobs
  - many serial jobs and large job arrays
  - large computational problems in general
- Assuming perfect parallel scaling
  - your 64-core job may be slower than 32-core ...

Problematic workflows (cont.)

- Excessive and/or "bad" I/O, i.e. anything resulting in high load on Lustre object storage servers
  - avoid high IOps workflows: lots of small files, read/write in chunks smaller than 1MB, reading small blocks from large files
- Storing a large number of small files
  - organize your code's output
  - use tar or even better dar (`http://dar.linux.free.fr`, supports indexing, differential archives, encryption)
- Using nested parallelism in black-box pipelines
  - e.g. submitting serial jobs each of which launches multiple threads, sometimes asking for all cores on a node
  - your pipeline should be adapted to the cluster; if not sure, please talk to us
- Moving files from one filesystem to another (e.g. `/home` → `/scratch`) when close to quota can lead to data loss
  - use copy instead

Other best practices

- Implement/use checkpointing to be prepared for system failures
- Break your job into pieces (if possible)
- Read the documentation about scheduling, running jobs, using modules, other topics ✍ `https://docs.computecanada.ca`
- Know as much as possible about your application, e.g. whether it is serial or parallel, and how it was parallelized (threaded vs. MPI)
- Start with some tests before running extensive simulations
  - estimate the resources (especially memory, wall time)
  - test parallel scaling, scaling with problem size
- Only request resources (memory, running time) needed
  - with a bit of a cushion, maybe 115-120% of the measured values
  - use `sacct` or `seff` to estimate your completed code's memory usage

Other best practices (cont.)

- If you still need to do lots of small I/O from inside your job:
  - use Slurm-generated directory `$SLURM_TMPDIR` (pointing to `/localscratch/${USER}.${SLURM_JOBID}.0` on a node's SSD) for both input and output
    - don't forget to move files out before your job terminates: everything in `$SLURM_TMPDIR` will be deleted
  - use `$TMPDIR` RAM disk (pointing to `/tmp`)
    - don't forget to allocate additional memory to your job
    - don't forget to move the results before your job terminates
- Port your workflow to another CC's general-purpose cluster, to run it there in case of failures
  - data management part may not be so easy, but Globus should help
  - also try to port your workflows (have accounts, appropriate input data, programs installed) to local clusters where available (Grex, Orcinus, Plato)
- If you received a `/scratch` purge warning, do *not* wait until the last minute to transfer data to local systems or other clusters
  - always pay attention to `/scratch` purge notices (email and system login message)

Other best practices (cont.)

- Be aware that some filesystems are not backed up (e.g. `/scratch`), and some have a purge policy (`/scratch`) ✍ have a backup plan
- If a file's path changes, our backup system will interpret it as a new file ⇒ unnecessary load on the filesystems
  - be careful with renaming large directories
- In general, do *not* run jobs in `/home`
  - slow, not designed for high performance (unlike `/scratch`)
  - small quota (50GB/user)
  - lots of I/O makes difficult to do backups
- After your job finishes:
  - clean up (remove files that are no longer needed)
  - compress large files to reduce the disk space usage
  - archive (tar) the directories with many small files to reduce the file count
  - eventually move your data from `/scratch` to `/project`, `/nearline`, your own storage