Overlall themes

- (Our explanations should not sound like excuses)
- (We do *not* want to project an idea that Cedar is unstable and users should avoid it)
- 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
- 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: distinguish between failures and high traffic?
- Scheduler failures
- 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
  - many 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 will be 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
- 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
- Sometimes you could work around a temporary filesystem problem by submitting jobs from another filesystem
  - on Cedar `/scratch`,`/home` files are handled by different servers than `/project`

What can you do? (cont.)

- 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
- Excessive and/or "bad" I/O, i.e. anything resulting in high load on Lustre object storage servers (4 on Cedar handling `/home` and `/scratch`)
- Complex/unrealistic job dependencies can make Slurm unstable
- Issuing too many requests to the scheduler
  - classical example: running `watch squeue ...`
  - submitting thousands of jobs and then cancelling them
- 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.)

- 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
- Moving files from one filesystem to another (e.g. `/home` → `/scratch`) when close to the quota can lead to data loss
  - use copy instead
- 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)

Other best practices

- Checkpointing to be prepared for system failures
- Break your job into pieces (if possible)
- Only request resources (memory, running time) needed
  - with a bit of a cushion, maybe 115-120% of the measured values
  - use `sacct` to estimate your completed code's memory usage
- Be aware that some filesystems are not backed up (e.g. `/scratch`), and some have a purge policy (`/scratch`) ✍ have a backup plan
- 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 to transfer data to local systems or other clusters
  - always pay attention to `/scratch` purge notices (email and system login message)