- Current maintenance
  - Cedar is down for system software updates (today only)
  - `/project` expansion March 1st - 4th, Lustre file metadata will be copied over to new SSDs, `/project` unavailable at this time, you can still use `/scratch` for running jobs during this time

- Cedar is a very complex system: lots of components, latest hardware and highly-optimized software (with small install base), shared filesystems, very broad user mix
  - $\sim 1,600$ nodes on Cedar, each with 24-48 cores, local storage
  - $\sim 66,000$ cores
  - 100Gb/s Omnipath interconnect linking all nodes and storage
  - three (Lustre) parallel file systems with $\sim 30$ PB combined storage
    - `/home`, `/scratch`, `/project`
    - each with its own policies, 2/3 backed up
  - 584 NVIDIA P100 Pascal GPUs
  - $\sim 60$ Slurm partitions, for long / short / GPU / large-memory / interactive jobs / CPU architecture
    - 3h, 12h, 1d, 3d, 7d, 28d maximum runtimes
    - trying to accommodate a large variety of job types
    - at the cost of efficiency and simplicity

- Our goals are to:
  - provide as much uptime as possible
    - we constantly monitor our clusters
    - work as quickly as possible to repair problems and return nodes to production
    - in case of downtime or other problems, provide frequent system status updates
  - accommodate a wide spectrum of jobs
  - maximize resource (CPUs, GPUs, memory, to smaller extent storage) utilization
  - minimize turnaround for your jobs

1. When hardware/etc problems occur, we want you to know how *in some cases* you can work around them
2. We want to show you how certain workflows can lead to problems on HPC clusters
   - and share with you best practices for working on these systems

# Major causes of system instability

## Node failures: a node needs rebooting or other work

- $\sim 1,600$ nodes on Cedar
- Of these, $\sim 30$ nodes have actual hardware failures at any one time
  - these get gradually replaced through a rather onerous return merchandise authorization (RMA) process
  - we are working with the vendor to simplify this process
- Marked offline by Slurm, for any number of reasons: not communicating, incorrect reports, low memory, cannot terminate the job, etc.
  - requires manual intervention
- Over-subscription of nodes, GPUs
  - e.g., too many threads
- Does not pass other checks and taken offline
  - GPUs get stuck in a strange state: *"Only EGL 1.4 and greater allows OpenGL as client API"*, requires reboot

# Major causes of system instability (cont.)

## File system problems

- Lustre object storage servers (OSS) can get overloaded with lots of small requests
  - example: this past Tuesday a user was running 90 jobs, all with high I/O in `/project` bringing it to a halt
  - putting these jobs on hold *did not* fix the system
  - one of the OSS servers had to be rebooted due to thread exhaustion (very heavy load requesting too many threads and eventually dead-locking)
  - end result: `/project` was not available to all users for $\sim$ 3 hours
- 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, but high I/O jobs might take much longer than expected
  - both go down $\Rightarrow$ the entire filesystem will hang
  - any downed server will have to be rebooted

more on high I/O later ☞

# Major causes of system instability (cont.)

## Scheduler (Slurm) failures

- Can get overloaded with too many requests
- Bugs ...

more on scheduler later ☞

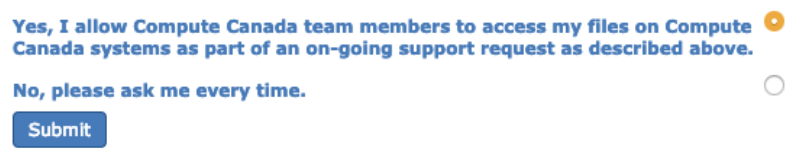
- No software stack synchronization between login and compute nodes
- Networking problems (within or outside our control)

# What do you see?

- Sluggish jobs
  - file system problem? node over-subscription? low memory? saturated network?
- Jobs not starting / taking unusually long to start
  - also valid reasons why your job's estimated start time could be pushed into the future
- Slurm not responding, or producing unusual output
  - e.g. last year's infamous Slurm bug leading to jobs stuck in 'Prolog' R (running) state for a long time, not producing any output
- Shell not responding to simple commands or very slow
  - could be per individual filesystem/command
- 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?

- Pay attention to login messages (system's MOTD = message of the day)
  - 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
- 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
    - check if you signed the consent that allow analysts to check your files (this will help resolve problems quickly instead of exchanging many emails), by logging in to `http://ccdb.computecanada.ca` and selecting My Account ➪ Agreements



# 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, input data)
- 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
  - priority may be lost (grows slowly with the waiting time for each job)
- 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 ...` (never do this!)
  - using a script to submit thousands of jobs and then cancelling them
- Complex/unrealistic job dependencies can make Slurm unstable

- Not testing first on a small scale, and not scaling up gradually
  - 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 lots of small reads/writes: many small files, frequent read/write in chunks smaller than 1MB, reading multiple small blocks from large files
- Storing a large number of small files
  - Lustre is very different from your laptop's drive
  - 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

# Problematic workflows (cont.)

- Using `mv` command to move files `/home,/scratch` → `/project` will result in an overquota error message in the middle of moving
  - this is expected behaviour!
  - not so much a problem for the cluster, but certainly will be a problem for you ...
  - in `/project` the 1TB (or higher) quota is applied to all files with the group ID `def-group`
    - so that all your group members are able to write there
    - any new file you write to `/project` will have `def-group` group ID
    - you can find this group ID by running `id` and looking for 'def-...'
  - by default, all files in `/home,/scratch` have group ID `username`
  - `mv` command preserves group ID, i.e. effectively `mv` acts as `cp -a`
  - the quota for group ID `username` is almost zero in `/scratch`
  - solution: use `cp` instead, followed by `rm`

# Other best practices

- Implement/use checkpointing to be prepared for system failures
- Break your job into pieces, if possible (time-wise, processor-wise)
- Read the documentation about scheduling, running jobs, using modules, other topics ✍ `https://docs.computecanada.ca`
- Know as much as possible about your application (serial vs. parallel), and how it was parallelized (threaded vs. MPI)
  - very important for creating the correct job submission script!
- Start with some tests before running extensive simulations
  - estimate the resources (especially memory, wall time)
  - use `sacct` or `seff` to estimate your completed code's memory usage
  - 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
  - otherwise your job will be queued much longer

# Other best practices (cont.)

- If you still need to do lots of small I/O from inside your job:
  - *use on-node SSD:* Slurm-generated directory `$SLURM_TMPDIR` points to `/localscratch/${USER}.${SLURM_JOBID}.0`
    - for both input and output
    - don't forget to move files out before your job terminates: everything in `$SLURM_TMPDIR` will be deleted
  - *use RAM disk:* `$TMPDIR` points 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, system's MOTD)
  - exercise care when transferring data close to quota in destination
  - when moving to `/project`, replace `mv` with `cp + rm`

# 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 `/home` and `/project`
- 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` (will be available on Cedar soon), your own storage