Common causes of job failures

These are some of the most common causes of job failures on NCAR systems and some tips for how to avoid them. Keep an eye out for other tips, and change notices that you need to be aware of, in the CISL Daily Bulletin.

1. **Running close to the node or GPU memory limit.**
   If you experience failures early in your job, especially at times when the program is reading data from disk or allocating data arrays, you may be running out of memory on compute nodes or GPUs. You may simply need to request more memory per node. Other potential solutions are to either spread your job across more nodes/GPUs (with fewer processes per resource) or shrink your problem size. See Checking memory use to determine how much memory your program or script requires to run.

2. **Home or other directory filling up.**
   If you are seeing erratic job failures or experiencing failures for jobs that have run successfully many times in the past, you may have used up your disk quota. Run the **gladequota** command and clean up any storage spaces that are at or near 100% full.

3. **Specifying version-specific modules in your dotfiles.**
   The problem with specifying version-specific modules in your dotfiles (.bashrc and .tcshrc, for example) is that someday the version you specify will be removed, and so the reproducibility of your job is reduced. You'll probably look at your batch job, not see anything wrong with it, and be puzzled, not realizing the problem is rooted in your dotfiles. In fact, it is best not to specify *any modules* in your dotfiles. Instead, include your environment modifications – any **module load** commands, for example – in your job script.

4. **Failure to clean directories when remaking executables and binaries.**
   This failure can be puzzling because it looks like everything built correctly. When you run your application, it fails in a way that does not point to the root cause of the problem: that the build is using one or more binaries from previous, incompatible builds.

5. **Using old binaries with new module versions.**
   Some software libraries – for example, many components of the Cray Programming Environment and NVIDIA HPC Toolkit – do not guarantee forward-compatibility with newer versions as they are released. Attempting to run binaries compiled with older versions of these libraries can result in strange runtime failures at launch. If you switch to a new version of a compiler, MPI, or GPU toolkit, it is best to rebuild your application.

6. **Filling up temporary file space.**
   Using shared /tmp, /var/tmp or similar shared directories to hold temporary files can increase the risk of your own programs and other users' programs failing when no more space is available. Set TMPDIR to point to your GLADE scratch space (or the fast NVMe local scratch on Casper) in every script for all batch jobs. See Storing temporary files with TMPDIR.