Best Practices for Managing Files

When using cluster storage and running parallel workloads, it is important to optimize your work to take advantage of available storage to ensure the speed and efficiency of your work as well as to avoid stressing the shared filesystem and negatively impacting performance for all users.

Storage Tiers explained

The DCC uses a hierachical organanization of storage media (called storage tiers) that differ in performance, cost, and capacity. Each option on the DCC has different intended use, performance, capabilities, and cost implications. See the full list of options here.

Using /scratch

Every DCC node has a local /scratch directory on NVMe, which is the highest throughput and fastest response time storage available on the DCC. Not only will this speed up your workloads, but it will also take pressure off the DCC shared filesystems. But beware: scratch space is limited, isolated per node, and is not backed up, so you should only use it for temporary file storage.

Examples for use: - When working with many small files, consider copying them to /scratch at the start of a run and reading them from there. - For STAR (mapping reads to genome indexes using STAR) mkdir -p /scratch/projectname/ --outTmpDir /projectname/${sample} --

Storing Files

- Avoid storing many files in a single directory – hundreds of files are ok, thousands are not. Our filesystem manages file locks (allows multiple nodes to access the same data concurrently safely) at the subdirectory level. Plan your data structure to use multiple sub-directories by default to minimize contention and locking events.
- Avoid operating on many small files – the performance may depend on how you handle the data on disk. Create a smaller number of larger files.

Be aware of I/O load

- when running multiple instance of the same job consider reading in all of the data at the beginning of the workflow, perform the entire analysis, then write the results at the end.
- Avoid opening/closing or creating/deleting files repeatedly in tight loops. Creating and deleting files requires locking certain filesystem structures, and these “synchronization points” are computationally expensive to enforce. Try to create files “as needed” through your workflow, and minimize the number created within a single directory. Similarly, delete files “as needed” (which is less “expensive” for directories that are “properly sized” in terms of number of files). Try not to re-use filenames in the same directory, when creating and deleting files in a loop. If possible, open files once at the beginning of your workflow/program, then close them at the end.

Watch your quotas

You are limited in capacity and exceeding your storage quotas will impact performance.

Choose the right command(s) for the jobs

- Avoid using ls -l in scripts – consider ‘find’ which is more appropriate for scripts, if you use ‘ls’ use it by itself or use ls -l filename if you want the long listing of a specific file.

A The NYU HPC center has published some suggestions you can try for your projects.