HPC data handling

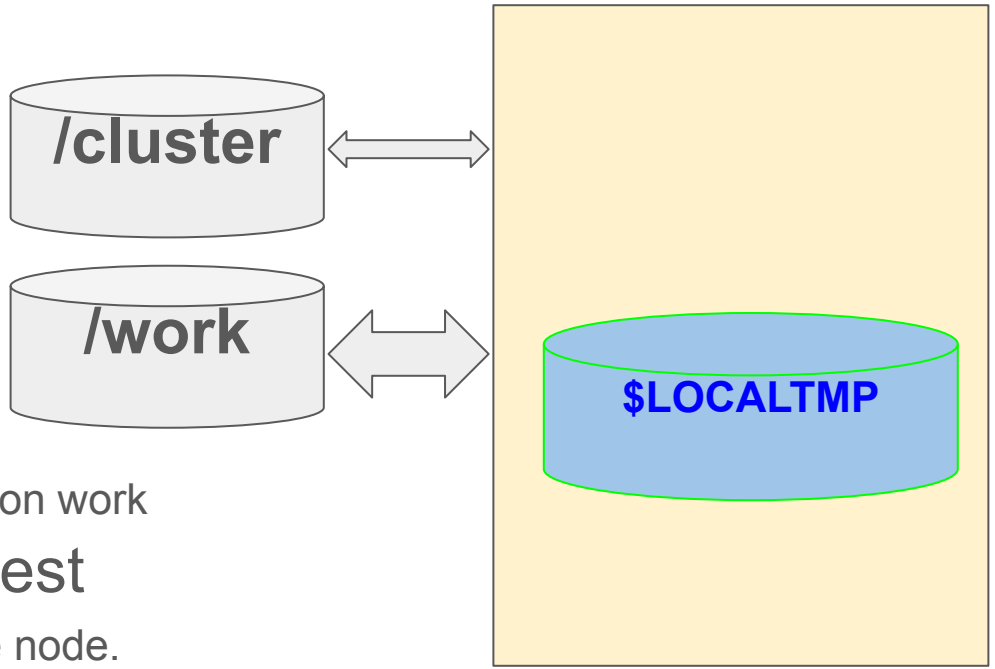
Sabry Razick

Topics

- Data staging
  - Using work
  - Using $SCRATCH
  - Using $TMP
- Check whether your job can run in parallel
  - Simple way using top
  - Investigate slurm log at the end
  - sacct
- Ask for the correct resources
- Make sure environment can be reproduced
- Investigate during the job run

Different locations

- To find out the mounts
  - df -h
- /cluster is slow
  - $HOME is on cluster
- /work is fast
  - /work/users/<USERNAME>
  - $SCRATCH is on work
  - Temp directory during a job is on work
- $LOCALTMP is the fastest
  - IT is a local disk on a compute node.
  - Not shared between nodes
  - Very good when there a millions of very small files to be processed.



Check the available directories

- Test job
  - mkdir $HOME/RCS_tutorial #If you did not so this on the first day
  - cp /cluster/teaching/abel_tutorial/NOV2018/location_test.slurm $HOME/RCS_tutorial
  - cd $HOME/RCS_tutorial
  - chmod +w location_test.slurm
  - nano location_test.slurm
    - Edit the project account
  - sbatch location_test.slurm

When there are hundreds of thousands files

- Test job
  - cd $HOME/RCS_tutorial
  - cp /cluster/teaching/abel_tutorial/MAY2019/millionfiles.slurm $HOME/RCS_tutorial
  - chmod +w millionfiles.slurm
  - nano millionfiles.slurm
    - Edit the project account
  - sbatch millionfiles.slurm
  - Investigate the output
    - tar -tvf outfiles.tar.gz

When there are hundreds of thousands jobs

- Create an archive (no compression)
  - tar -cvf <ARCHIVE_NM> <FILES>
  - tar -cvf file.tar *txt
- List content
  - tar -tvf <ARCHIVE_NM>
- Append a file
  - tar --append --file <ARCHIVE_NM> <NEW_FILES>
  - tar --append --file files.tar.gz 1.txt
- Extract all
  - tar -xvzf <ARCHIVE_NM>
- Extract one
  - tar -xvf <ARCHIVE_NM> <FILES>

Big file and lsof

- Test job
  - cd $HOME/RCS_tutorial
  - cp -r /cluster/teaching/abel_tutorial/MAY2019/IO $HOME/RCS_tutorial
  - cd $HOME/RCS_tutorial/IO
  - chmod +w file_IO.slurm
  - nano file_IO.slurm
    - Edit the project account
  - sbatch file_IO.slurm
  - Find the compute node
    - squeue -u $USER
  - ssh <compute_node>
  - Find the process
    - top -c -u $USER #Get PID of “python3 file_IO.py” process
  - Investigate IO access
    - lsof -p <PID>

sacct

- Investigate resource usage after the job is finished
- sacct -j <JOB_ID>
- sacct -j <JOB_ID> -o all

For long running compilations

- srun --account=<ACCOUNT> --ntasks=4 --mem-per-cpu=2G --time=1:00:00 /bin/bash

Interactive with GPU

- srun --account=staff --partition=accel --ntasks=8 --mem-per-cpu=6G --gres=gpu:2 --time=5:00 /bin/bash