Analysis Pipeline to GCP Slurm
Issues and Resolution
May 17, 2020

1. **Early adopters of Slurm on GCP**
   There were miscellaneous issues associated with being one of the earliest adopters of Slurm on GCP. For example, the support of multiple partitions in Slurm had some bugs but these were addressed very early on.
2. **Docker performance**
   Pulling a new docker image (e.g., uwgac/topmed-roybranch ~8GB) from the public docker hub (https://hub.docker.com) was slow which often resulted in the docker daemon timing out.

   We had a bit better performance when using GCP's repository (based in Google Storage) but occasionally still had the same problem.

   The solution was to preload and pull the desired docker image on the custom VM used for Slurm's compute nodes. I still had the option to pull down a newer image if needed (which worked better when updating the newer image when only a small number image layers have been modified).
3. **Mounting NFS projects problems**
   When submitting jobs, Slurm would start the job before the mount to our NFS volume completed. We fixed this issue by editing the slurm compute-node setup script (setup.py) and adding our 'projects' mount dependency to the code:

   After=network.target munge.service home.mount apps.mount etc-munge.mount **projects.mount**

4. **gcsfuse mount issues**
   Similar to issue #3, In /etc/fstab, gcsfuse mount (which requires the network to be up) did not complete before the Slurm job started. This time, Jonathon found a fix. Evidently, the mount was attempted before the network was up. By adding an option to the gcsfuse mount command in /etc/fstab, fixed the problem. There are two different mount commands in /etc/fstab that can be used:

   gcp-xxx-bucket /fuse gcsfuse rw,**x-systemd.requires=network-online.target**, allow_other, implicit_dirs
   or
   gcp-xxx-bucket /fuse gcsfuse rw,**_netdev**, allow_other, implicit_dirs

   We used the second mount command (i.e., with the option **_netdev**)
5. **Operating system updates during job executions (causing jobs to fail)**
   All of our jobs run via docker. We were getting sporadic job failures with the docker daemon failing.

   By looking at the system logs for a Slurm compute instance that failed and tracing back from the failed time, we eventually noticed that docker daemon was restarting. With further investigation into the logs, we noticed that yum was updating system packages and causing docker to restart.

   Evidently, by default, GCP Centos VM's enable automatic yum updates. Disabling this automatic update fixed our problem.
6. **Lack of GCP resource quotas**

GCP has resource quotas associated with the maximum number of cores/vcpu's. The initial limit of 2,400 was insufficient for our pipeline and resulted in Slurm job failures. We requested a bump to 35,000 to accommodate the size of the data, number of expected jobs, and number of analysts running the pipeline at one time.

Google's engineering team really pushed back on this request (they prefer having smaller increments in resource quotas; demonstrating that you are running into a resource quota limit; etc). Eventually after a week or so our request for the quotas was approved.

7. **Dependent jobs not starting for job arrays**
   A job depending on the successful completion of a job array was started after the job array completed successfully. This was a bug and was fixed.
8. **Cannot run multiple jobs on the same compute node (when there are sufficient resources on the compute node)**
   For most jobs in the analysis pipeline, each job runs on a dedicated compute node. Because of cost, we only select a compute node of sufficient resources with very little resources available for running other jobs. However, there are some cases where there may be available resources to run another job on the same compute node. Slurm has this capability but we had difficulty enabling this. With the help Google (Eric), we eventually found a configuration of Slurm on GCP that provided this capability - it was just a matter of editing the slurm config file (/apps/slurm/current/etc/slurm.conf) and restarting the Slurm Controller daemon. See the *Cluster-Deployment Guide* for details.
9. **Idiosyncrasies submitting a Python script to Slurm (script copied to /var/...)**
   When submitting a script to Slurm to run a batch job (via sbatch command), Slurm copies the script to a directory under /var for execution.

   For most scripts (either shell or Python), this would not be a problem. However, if a Python script is being executed and the Python script is importing a module in its original directory, then an error will occur. Python can only import a module from the standard system location or the location of the script. By copying the Python script to /var/..., the script will fail.

   The solution was to execute a shell script with parameters specifying the full path of the Python script and its arguments. The shell script would then execute the Python script correctly.