Pydpiper on the SickKids HPF

This is mostly condensed from HPF documentation; see https://wiki.mouseimaging.ca/download/attachments/7897100/HPF%20User%20Documentation.pptx?api=v2, https://ccm.sickkids.ca/?page_id=61, and https://hpc.ccm.sickkids.ca.

- General outline
- Transfer data
- Troubleshooting a pipeline
- Example calls for MBM.py
- Disk Clean up
- Login
  - Login nodes
  - qlogin (interactive/submit) nodes
    - Surviving network interruptions with Tmux or GNU Screen
  - How long will your qlogin session still run for?
- Modules
- Run Pydpiper
- Known issues

General outline

- Transfer data onto /hpf/largeprojects/MICe.
- Start a new `qlogin` session with sufficient walltime and memory to run Pydpiper.
- Load modules.
- Run your Pydpiper job as you would at MICe.

Read on for details.

Transfer data

A directory for our projects on the HPF file system is mounted at MICe as /hpf/largeprojects/MICe, so it's trivial to move data back and forth - simply place it in this directory, and it'll be visible from both systems. You'll need to create a subdirectory /hpf/largeprojects/MICe/yourusername.

The HPF also has data nodes, but we don't really need these since data transfer is trivial and the qlogin nodes now have network access for installing software.

Troubleshooting a pipeline

See here for some detailed information on how to troubleshoot a pipeline: Troubleshooting pydpiper pipelines

Example calls for MBM.py

MBM.py cookbook

Disk Clean up

When you're finished with a registration, have the data analysed and are ready to archive the pipeline, you can remove more than just the tmp directories from the pipeline. See these notes on how to perform a thorough disk clean up after a MBM run

Login

Login nodes

First, `ssh` into a *login node*:

```
bdarwin@despereaux$ ssh -AX hpf.ccm.sickkids.ca
password:
[bdarwin@hpf23 ~]$
```

You may omit the "`username@`" if your local user is the same (which will usually be the case here). `-A` enables ssh agent forwarding and -X enables X forwarding; you may (even should) omit these if you don't plan to use them. No projects (`/hpf/projects`, etc.) or tools (`/hpf/tools`) directories are available on these nodes, so you can't do much useful from here.

qlogin (interactive/submit) nodes

Next, from a login node:

```
[bdarwin@hpclogin3 ~]$ # if your pipeline is very large, you can bump this up to a maximum of mem=16G,
walltime=120:00:00:
[bdarwin@hpclogin3 ~]$ qlogin -l mem=8G,walltime=72:00:00
qsub: waiting for job 1904303 to start
qsub: job 1904303 ready
[bdarwin@qlogin1 ~]$
```

From the qlogin nodes, you may `qsub` jobs (or run PydPiper) as usual.

The -l flag specifies resource requests for your qlogin session. If not specified, the defaults are 2 hours and 2G of RAM (for larger pipelines, consider requesting 8 or even 16G). You may have up to 6 interactive sessions of 16GB RAM or less, for a maximum of 5 days, as well as a single interactive session of up to 48 GB of RAM for a maximum of 24 hours. (There's apparently also a `ppn` option, but I don't know how to use it.) We might consider asking for even longer time limits.

Surviving network interruptions with Tmux or GNU Screen

Using a terminal multiplexer such as Screen or Tmux is useful to avoid losing your HPF session/pipeline due to a network interruption or local machine issues – particularly useful if you're working from a laptop.

I current recommend Tmux over Screen; see Remote working: directly via SSH for instructions. One simply starts a multiplexer session on the login nodes - but apparently not the qlogin nodes.

You could achieve the same result via screen as follows:

`[bdarwin@hpclogin3 ~]$ screen`

`[bdarwin@hpclogin3 ~]$ qlogin ...`

`[bdarwin@qlogin1 ~]$ MBM.py ...`

# now press Control-"a" (to send a command to screen instead of the shell) followed by Control-"d" (for detach)

[bdarwin@hpf23 ~]$ ^D

bdarwin@vinnie ~$ # at this point we have no ssh connection to HPF at all

# now a screen session is running independently on hpclogin3 ... to reattach:

`[bdarwin@hpclogin3 ~]$ screen -r   # reattach`

[bdarwin@qlogin1 ~]$ MBM.py .... # <-- lots of dots emitted by Pydpiper

# if you're satisfied with your pipeline's progress, you can detach again ...

HPF has several different login machines, currently `hpclogin1` through `hpclogin4`. When you log back into HPF you might end up on one of the other login nodes. You will find out by running the following command:

```
[matthijs@hpclogin4 ~]$ screen -ls
No Sockets found in /var/run/screen/S-matthijs.
```

Now what do you do? The answer is that you can simply ssh into the login node you want:

```
[matthijs@hpclogin4 ~]$ ssh hpclogin3
Last login: Tue Sep 26 13:31:55 2017 from hpf24.cm.cluster
[matthijs@hpclogin3 ~]$

# now you will see your screen again:

[matthijs@hpclogin3 ~]$ screen -ls
There is a screen on:
    13900.pts-6.hpf23    (Detached)
1 Socket in /var/run/screen/S-matthijs.

# and you can re-attach to your screen like so:

[matthijs@hpclogin3 ~]$ screen -r 13900.pts-6.hpf23
```

Next up, multiple screens! After you have detached from a screen you do not have to reattach to it necessarily. You can also simply run screen again, and start a second one:

```
# starting empty:
screen -ls
No Sockets found in /var/run/screen/S-matthijs.

# now start a screen
[matthijs@hpclogin3 ~]$ screen

#.... do things in your screen and detach by using Ctrl+a and then Ctrl+d
# you can see that screen:

[matthijs@hpclogin3 ~]$ screen -ls
There is a screen on:
    13976.pts-6.hpf23    (Detached)
1 Socket in /var/run/screen/S-matthijs.

# you don't have to reattach to the same screen, but can start a new one just by typing screen again:
[matthijs@hpclogin3 ~]$ screen

# now after doing some work and detaching again, you'll see the following:

[matthijs@hpclogin3 ~]$ screen -ls
There are screens on:
    13998.pts-6.hpf23    (Detached)
    13976.pts-6.hpf23    (Detached)
2 Sockets in /var/run/screen/S-matthijs.

# you can reattach to the screen you want by specifying it specifically:
[matthijs@hpclogin3 ~]$ screen  -r 13998.pts-6.hpf23
```

TODO: move this section to a separate page and link to it from the HPF/SciNet/Graham wikipages ?

How long will your qlogin session still run for?

This section is out of date following the Centos 6 Centos 7 upgrade on March 1, 2021. You can use `/opt/qlogin_torque/bin/qstat` in a similar way as below but it doesn't seem to show the wall time elapsed, making this endeavour somewhat futile.

In the previous command you might have specified mem=8G,walltime=72:00:00. A day or two later you can find out how much time is left in that session using:

```
matthijs@mrjingles:~$ ssh hpf.ccm.sickkids.ca
Password:
[matthijs@hpclogin6 ~]$

# on this node (so prior to logging into a qlogin node, you can run the following:
[matthijs@hpclogin6 ~]$ /opt/qlogin/bin/qstat -u $USER
# In my case this shows:

qtorquemaster.hpf.cluster:
                                                                                  Req'd    Req'd       Elap

Job ID                  Username    Queue    Jobname          SessID  NDS   TSK   Memory   Time    S   Time

----------------------- ----------- -------- ---------------- ------ ----- ------ ------ --------- - ---------

101958.qtorquemaster.h  matthijs    qloginQ  STDIN            229398     1      1    2gb  02:00:00 R  00:54:45

# so I have roughly an hour left in the 2 hour qlogin session.
```

Modules

We use `modulecmd(1)` to set paths and other environment variables for our software:

```
# use our own modules; almost certainly safe to put this line in your ~/.bashrc:
[bdarwin@qlogin1 ~]$ module use /hpf/largeprojects/MICe/tools/modulefiles
[bdarwin@qlogin1 ~]$ module load mice-env
# to see what this is loading:
[bdarwin@qlogin1 ~]$ module show mice-env  # or, e.g., mice-env/1.0.4 to fix a specific version
Currently Loaded Modulefiles:
  1) gcc/5.2.0               8) minc-stuffs/0.1.21     15) pcre/8.38
  2) octave/4.0.0            9) imagemagick/6.8.9-5    16) netcdf/4.3.2
  3) minc-toolkit/1.9.14    10) pydpiper/2.0.10        17) curl/7.49.1
  4) python/3.6.2           11) MICe-lab/0.17          18) openMPI/2.1.1
  5) parallel/20170922      12) zlib/1.2.8             19) R/3.3.2
  6) qbatch/1.0.1-20180123  13) bzip2/1.0.6            20) RMINC/1.5.1.0
  7) pyminc/0.51            14) xz/5.2.2               21) mice-env/1.0.4
# you can also use the old method of loading individual modules by hand, in which case you'll need to specify
some (now out-of-date) versions:
[bdarwin@qlogin1 ~]$ module purge
[bdarwin@qlogin1 ~]$ module load gcc/5.2.0 octave minc-toolkit/1.9.14 python/3.6.2 pyminc/0.51 minc-stuffs/0.
1.21 imagemagick parallel qbatch/1.0.1-20170829 pydpiper/2.0.9
```

(In the future, we intend to make manual loading of the prerequisite modules unnecessary.)

The Pydpiper module also sets some system-specific settings via a config file at `$PYDPIPER_CONFIG_FILE` which you can `cat`.

Run Pydpiper

Make sure you have enough time remaining in your qlogin session!

See `/hpf/largeprojects/MICe/tools/initial-models/` and `/hpf/largeprojects/MICe/tools/protocols` for relevant files.

You generally don't need to specify `--mem`, --proc, --time, --queue-type, &c. since this is done in the configuration file.

As an example, a simple MBM pipeline should look something like this, if you choose not to run MAGeT:

```
MBM.py --pipeline-name=Pipeline_Date --subject-matter mousebrain --num-executors 30 --time 48:00:00 --init-
model /hpf/largeprojects/MICe/tools/initial-models/Pydpiper-40-micron-basket-dec-2014
/basket_mouse_brain_40micron.mnc --no-common-space-registration --no-run-maget --maget-no-mask --lsq12-protocol
/hpf/largeprojects/MICe/tools/protocols/linear/Pydpiper_testing_default_lsq12.csv --files *.mnc
```

You should see something like the following:

```
Couldn't import dot_parser, loading of dot files will not be possible.
Total number of stages in the pipeline:  155359
Number of stages already processed:      103220
Daemon is running at: 192.168.100.72:37503

The pipeline's uri is: PYRO:obj_863e48321092496ba5c578f892c1b1ec@192.168.100.72:37503

3725292[]
```

The `[]` indicates that Pydpiper has submitted an "array job" - i.e., an array of nearly identical jobs. You can get detailed information about the individual elements of the array using `qstat -t 3725292[]` (note the brackets). (Note: this is standard at MICe as well, but I don't know where to put general Pydpiper information yet ...)

Known issues

- You may occasionally get sporadic errors about executors dying, shortly after they start. This happens when a node's ramdisk (`/dev/shm/`) is full, causing the executor to crash. For now, `--max-failed-executors` has been set to some large number in the config file.
- If executors are dying *systematically* or a specific stage is repeatedly failing, this probably indicates stages are being cancelled by the scheduler for using too much virtual memory ('vmem') due to its very conservative accounting scheme. Try a 'grep vmem *-e.log*' if you suspect this. As a temporary workaround I've increased the default job memory in the configuration file, but this means executors may be slower to start running.

- The server itself doesn't use much memory for all but very large pipelines, but the scheduler sees double that amount at the moment when jobs are being submitted. This could be fixed if it's problematic.
- If the scheduler is not accepting jobs, you'll see some further errors when trying to submit. Currently Pydpiper will just repeatedly try to submit, but this isn't really tested yet. If Pydpiper gets into a state where it believes some executors are still queued but `qstat -u ...` shows otherwise and restarting your pipeline is time-consuming, you may submit the appropriate number of executors manually via `pipeline_executor.py --uri-file=/full/path/to/build-model-jan27_uri --num-executors=n`.
- Pydpiper >= 2.0.8 by default resamples your data to consensus space and requires a common space template. This is good practice as it allows for future meta-analysis. With that said, this feature may not be appropriate for many studies (for example studies with neonatal brains). Add the "--no-common-space-registration" flag to skip this registration step.