Segmenting Large Electron Microscopic Image Volumes

An Introduction to NBCR image analysis and segmentation tools

National Biomedical Computation Resource Summer Training Program @ UC San Diego

Chris Churas & Matthias Haberl
August 7th, 2017
churas@ncmir.ucsd.edu & mhaberl@ucsd.edu

Hands-On Session 2

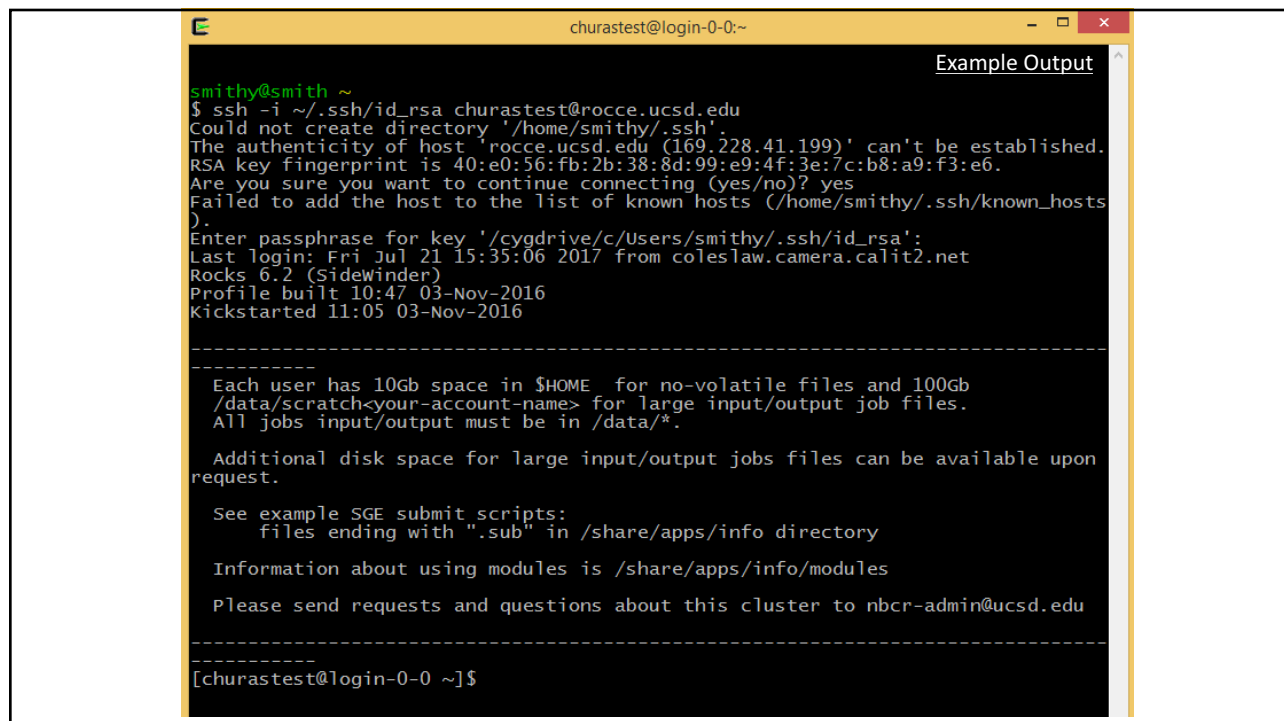Running CHM Train via chmutil on Rocce Cluster

Goals

By the end of this session, you will be able to:

1. Install chmutil on Rocce Cluster
2. Upload training data to Rocce Cluster
3. Autocontrast training images
4. Create CHM Train job
5. Submit CHM Train job
6. Check on CHM Train job status

1. Install chmutil on Rocce Cluster

A. Connect to Rocce Cluster via *ssh* by opening a terminal and invoking this command:

   ssh –i ~/.ssh/id_rsa <USER>@rocce.ucsd.edu

```
smithy@smith ~
$ ssh -i ~/.ssh/id_rsa churastest@rocce.ucsd.edu
Could not create directory '/home/smithy/.ssh'.
The authenticity of host 'rocce.ucsd.edu (169.228.41.199)' can't be established.
RSA key fingerprint is 40:e0:56:fb:2b:38:8d:99:e9:4f:3e:7c:b8:a9:f3:e6.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/smithy/.ssh/known_hosts
).
Enter passphrase for key '/cygdrive/c/Users/smithy/.ssh/id_rsa':
Last login: Fri Jul 21 15:35:06 2017 from coleslaw.camera.calit2.net
Rocks 6.2 (SideWinder)
Profile built 10:47 03-Nov-2016
Kickstarted 11:05 03-Nov-2016


-----------------------------------------------------------------------------
-----------
  Each user has 10Gb space in $HOME  for no-volatile files and 100Gb
  /data/scratch<your-account-name> for large input/output job files.
  All jobs input/output must be in /data/*.

  Additional disk space for large input/output jobs files can be available upon
request.

  See example SGE submit scripts:
       files ending with ".sub" in /share/apps/info directory

  Information about using modules is /share/apps/info/modules

  Please send requests and questions about this cluster to nbcr-admin@ucsd.edu

-----------------------------------------------------------------------------
-----------
[churastest@login-0-0 ~]$
```

1. Install chmutil on Rocce Cluster

These instructions are taken from https://github.com/CRBS/chmutil/wiki/chmutil-on-rocce-cluster

A. Run these two commands to download and install Miniconda

qrsh
wget https://repo.continuum.io/miniconda/Miniconda2-latest-Linux-x86_64.sh
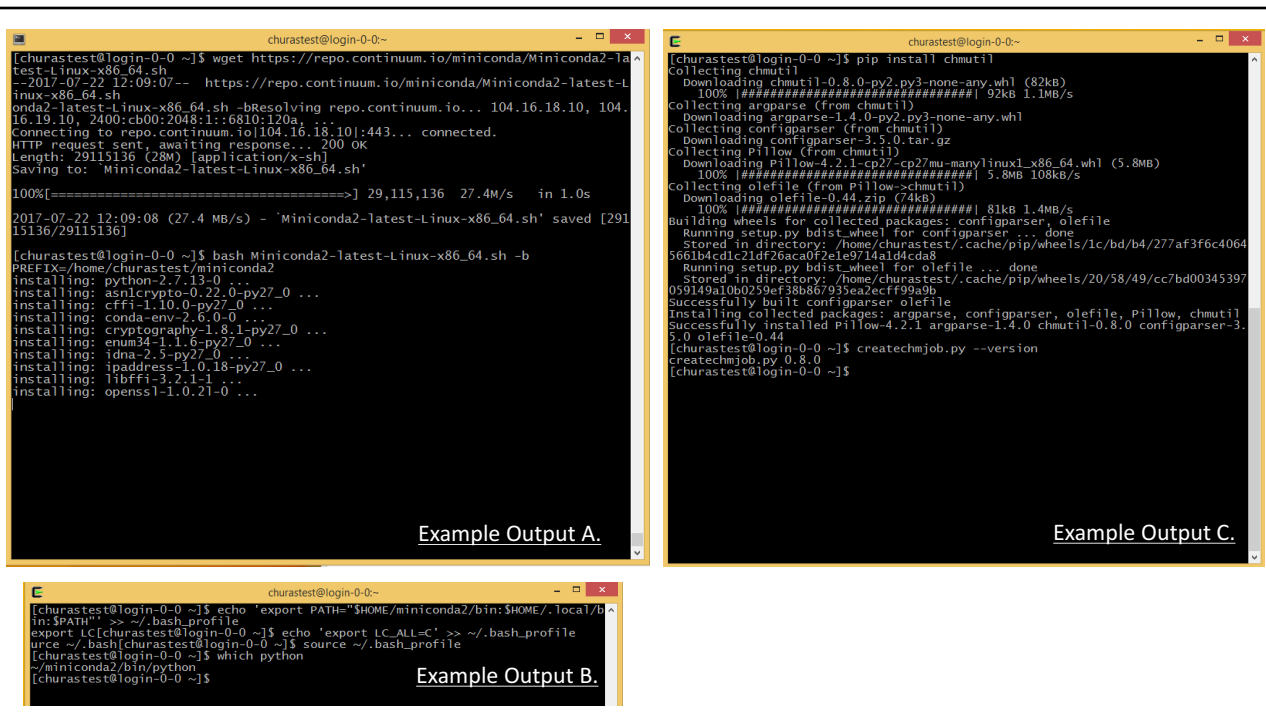*bash Miniconda2-latest-Linux-x86_64.sh –b*

B.
Run these three commands to update startup scripts with the path to Miniconda

echo 'export PATH="$HOME/miniconda2/bin:$HOME/.local/bin:$PATH"' >> ~/.bash_profile
echo 'export LC_ALL=C' >> ~/.bash_profile
source ~/.bash_profile

C.
Run these two commands to install chmutil and verify installation by calling *createchmjob.py*

pip install chmutil
createchmjob.py –-version
exit

Note: The above commands only have to be run once

3

```
churastest@login-0-0:~
[churastest@login-0-0 ~]$ wget https://repo.continuum.io/miniconda/Miniconda2-la
test-Linux-x86_64.sh
--2017-07-22 12:09:07--  https://repo.continuum.io/miniconda/Miniconda2-latest-L
inux-x86_64.sh
onda2-latest-Linux-x86_64.sh -bResolving repo.continuum.io... 104.16.18.10, 104.
16.19.10, 2400:cb00:2048:1::6810:120a, ...
Connecting to repo.continuum.io|104.16.18.10|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 29115136 (28M) [application/x-sh]
Saving to: 'Miniconda2-latest-Linux-x86_64.sh'

100%[===========================================>] 29,115,136  27.4M/s    in 1.0s

2017-07-22 12:09:08 (27.4 MB/s) - 'Miniconda2-latest-Linux-x86_64.sh' saved [291
15136/29115136]

[churastest@login-0-0 ~]$ bash Miniconda2-latest-Linux-x86_64.sh -b
PREFIX=/home/churastest/miniconda2
installing: python-2.7.13-0 ...
installing: asn1crypto-0.22.0-py27_0 ...
installing: cffi-1.10.0-py27_0 ...
installing: conda-env-2.6.0-0 ...
installing: cryptography-1.8.1-py27_0 ...
installing: enum34-1.1.6-py27_0 ...
installing: idna-2.5-py27_0 ...
installing: ipaddress-1.0.18-py27_0 ...
installing: libffi-3.2.1-1 ...
installing: openssl-1.0.2l-0 ...
```

Example Output A.

```
churastest@login-0-0:~
[churastest@login-0-0 ~]$ pip install chmutil
Collecting chmutil
  Downloading chmutil-0.8.0-py2.py3-none-any.whl (82kB)
    100% |################################| 92kB 1.1MB/s
Collecting argparse (from chmutil)
  Downloading argparse-1.4.0-py2.py3-none-any.whl
Collecting configparser (from chmutil)
  Downloading configparser-3.5.0.tar.gz
Collecting Pillow (from chmutil)
  Downloading Pillow-4.2.1-cp27-cp27mu-manylinux1_x86_64.whl (5.8MB)
    100% |################################| 5.8MB 108kB/s
Collecting olefile (from Pillow->chmutil)
  Downloading olefile-0.44.zip (74kB)
    100% |################################| 81kB 1.4MB/s
Building wheels for collected packages: configparser, olefile
  Running setup.py bdist_wheel for configparser ... done
  Stored in directory: /home/churastest/.cache/pip/wheels/1c/bd/b4/277af3f6c4064
5661b4cd1c21df26aca0f2e1e9714a1d4cda8
  Running setup.py bdist_wheel for olefile ... done
  Stored in directory: /home/churastest/.cache/pip/wheels/20/58/49/cc7bd00345397
059149a10b0259ef38b867935ea2ecff99a9b
Successfully built configparser olefile
Installing collected packages: argparse, configparser, olefile, Pillow, chmutil
Successfully installed Pillow-4.2.1 argparse-1.4.0 chmutil-0.8.0 configparser-3.
5.0 olefile-0.44
[churastest@login-0-0 ~]$ createchmjob.py --version
createchmjob.py 0.8.0
[churastest@login-0-0 ~]$
```

Example Output C.

```
churastest@login-0-0:~
[churastest@login-0-0 ~]$ echo 'export PATH="$HOME/miniconda2/bin:$HOME/.local/b
in:$PATH"' >> ~/.bash_profile
export LC[churastest@login-0-0 ~]$ echo 'export LC_ALL=C' >> ~/.bash_profile
urce ~/.bash[churastest@login-0-0 ~]$ source ~/.bash_profile
[churastest@login-0-0 ~]$ which python
~/miniconda2/bin/python
[churastest@login-0-0 ~]$
```

Example Output B.

2. Upload training data to Rocce

A. If still connected, disconnect from Rocce by typing *exit*

exit

B. **Either** download the *download_dataset_2.zip* file from the BBDTC portal and unzip and enter it as before **OR** if you want, enter *download_dataset_1* directory where training data created from previous exercise.

C. The directory will contain one item (more if you entered *download_dataset_1* folder*)*:

i. A directory, *train*, which contains two directories *images* and *labels* that contain training data created in previous exercise.

D. The training data is already in correct format, so we just need to upload it to Rocce:

scp –i ~/.ssh/id_rsa –r train <USER>@rocce.ucsd.edu:/data/scratch/<USER>/.

This will recursively upload the *train* directory to */data/scratch/<USER>* directory on Rocce.
Note: Be sure to replace <USER> above with your Rocce username.

Directory Structure:

```
download_dataset_2
  └── train
       ├── images
       │     ├── x.001.png
       │     └── x.002.png
       └── labels
             ├── x.001.png
             └── x.002.png
```

Example Output

```
[churastest@login-0-0 churastest]$ exit
logout
Connection to rocce.ucsd.edu closed.

smithy@smith ~
$ ls Downloads/download_dataset_2.zip
Downloads/download_dataset_2.zip

smithy@smith ~
$ unzip Downloads/download_dataset_2.zip
Archive:  Downloads/download_dataset_2.zip
   creating: download_dataset_2/
   creating: download_dataset_2/train/
   creating: download_dataset_2/train/images/
  inflating: download_dataset_2/train/images/x.000.png
  inflating: download_dataset_2/train/images/x.001.png
   creating: download_dataset_2/train/labels/
  inflating: download_dataset_2/train/labels/x.000.png
  inflating: download_dataset_2/train/labels/x.001.png

smithy@smith ~
$ cd download_dataset_2

smithy@smith ~/download_dataset_2
$ ls
train

smithy@smith ~/download_dataset_2
$ scp -i ~/.ssh/id_rsa -r train churastest@rocce.ucsd.edu:/data/scratch/churaste
st
Could not create directory '/home/smithy/.ssh'.
The authenticity of host 'rocce.ucsd.edu (169.228.41.199)' can't be established.
RSA key fingerprint is 40:e0:56:fb:2b:38:8d:99:e9:4f:3e:7c:b8:a9:f3:e6.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/smithy/.ssh/known_hosts
).
Enter passphrase for key '/cygdrive/c/Users/smithy/.ssh/id_rsa':
x.000.png                                    100%  131KB 131.2KB/s   00:00
x.001.png                                    100%  133KB 133.4KB/s   00:00
x.000.png                                    100% 1242    1.2KB/s   00:00
x.001.png                                    100% 3638    3.6KB/s   00:00

smithy@smith ~/download_dataset_2
$
```

3. Autocontrast training images

When displaying images IMOD auto contrasts the data, to Train CHM it helps to apply a similar auto contrast on the images. The commands below do just that with the chmutil command *createchmimage.py*
https://github.com/CRBS/chmutil/wiki/createchmimage.py

A. Connect to Rocce Cluster via *ssh*, start an interactive job, & change directory to
/data/scratch/<USER>/train/images Be sure to replace <USER> with you Rocce username.

ssh –i ~/.ssh/id_rsa <USER>@rocce.ucsd.edu
qrsh
cd /data/scratch/<USER>/train/images

B. Use a bash for loop & chmutil command *createchmimage.py* to auto contrast images

for Y in `/bin/ls` ; do createchmimage.py $Y $Y --autocontrast ; done

Note: The ` character is a back tick (it is usually above the tab key on keyboard)

C. Exit from interactive job

exit

Directory Structure:

```
|
|____ <USER>
         |
         |____ train
```

4. Create CHM Train job

We will be running the chmutil command *createchmtrainjob.py* to create CHM train job on the Rocce cluster. Here is a description of the arguments used in the command invocation below

--chmbin	Path to CHM Singularity binary (use path as shown below)
--cluster	Tells tool what cluster to generate job for. (currently Rocce & Comet are supported)
--stage	CHM parameter, sets number of stages (default 2)
--level	CHM parameter, sets number of levels (default 4)
--maxmem	Lets caller define amount of ram in gigabytes needed by job (default 90)

A.   Change to */data/scratch/<USER>* and run *createchmtrainjob.py* with arguments as shown below

cd /data/scratch/<USER>

createchmtrainjob.py train/images train/labels mytrain --chmbin /data/churas/chm_s22.img --cluster rocce --stage 2 --level 1 --maxmem 12

The above command will create a new directory named *mytrain*.

For more information on how to use *createchmtrainjob.py* just run the command with no arguments or with *–h* flag (Example: *createchmtrainjob.py --help*)

https://github.com/CRBS/chmutil/wiki/createchmtrainjob.py

Directory Structure:

```
└── <USER>
     ├── train
     ├── mytrain
     │    ├── stdout
     │    ├── tmp
     │    ├── readme.txt
     │    └── runtrain.rocce
```
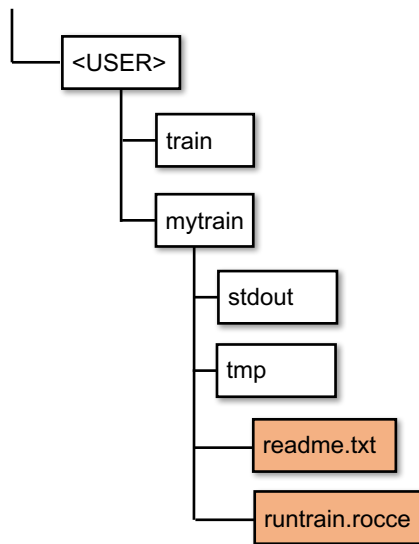
```
                    churastest@login-0-0:/data/scratch/churastest              –  □  ✕
[churastest@login-0-0 ~]$ cd /data/scratch/churastest
[churastest@login-0-0 churastest]$ createchmtrainjob.py ./train/images ./train/l
abels ./mytrain --chmbin /data/churas/chm_s22.img --cluster rocce --stage 2 --le
vel 1 --maxmem 12

To submit run: cd /data/scratch/churastest/mytrain; qsub /data/scratch/churastes
t/mytrain/runtrain.rocce

[churastest@login-0-0 churastest]$
```

Example Output

---

5. Submit CHM Train job

Running *createchmtrainjob.py* from previous step creates a *mytrain* folder with a couple files and directories described here:

stdout/   Directory that will contain standard error and output file for training job. The file will be named <JOBID>.out

tmp/   Directory that will temporarily hold output from CHM Train. As the job finishes this directory will be renamed to *model/*

readme.txt   Text file with information about job (parameters passed to *createchmtrainjob.py*) as well as information on how to check on job status and get more help

runtrain.rocce Generated script to run CHM Train job on Rocce cluster. This file is passed to SGE *qsub* command

A.   When run successfully *createchmtrainjob.py* outputs a command to run to actually submit the job.

In the terminal copy the text to the right of "*To submit run:*" and paste it in the terminal to submit the job.

The text to copy will look similar to the text below:

cd /data/scratch/<USER>/mytrain; qsub /data/scratch/<USER>/runtrain.rocce

Directory Structure:

```
mytrain
├── stdout
│   └── <JOBID>.out
├── tmp
├── readme.txt
└── runtrain.rocce
```
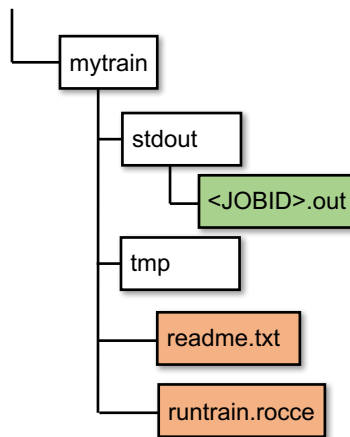
churastest@login-0-0:/data/scratch/churastest/mytrain

```
[churastest@login-0-0 churastest]$ cd /data/scratch/churastest
[churastest@login-0-0 churastest]$ createchmtrainjob.py ./train/images/ ./train/
labels/ ./mytrain --chmbin /data/churas/chm_s22.img --cluster rocce --stage 2 --
level 1 --maxmem 12

To submit run: cd /data/scratch/churastest/mytrain; qsub /data/scratch/churastes
t/mytrain/runtrain.rocce

[churastest@login-0-0 churastest]$  cd /data/scratch/churastest/mytrain; qsub /d
ata/scratch/churastest/mytrain/runtrain.rocce
Your job 14706 ("chmtrainjob") has been submitted
[churastest@login-0-0 mytrain]$
```

<JOBID>

Job name, can be set with
--jobname flag passed
to *createchmtrainjob.py*

Example Output

6. Check on CHM Train job status

When you run the *qsub* command as done in previous step, SGE will output a line of this format:

Your job <JOBID> ("<JOBNAME>") has been submitted

The <JOBID> is the identifier for the job and is used as the file name for the file containing standard error/output of job which is stored in *stdout/<JOBID>.out*. Contents of this text file can be examined for debugging purposes.

A. To check on job status run *qstat* command that is a SGE command that outputs all jobs your account is running
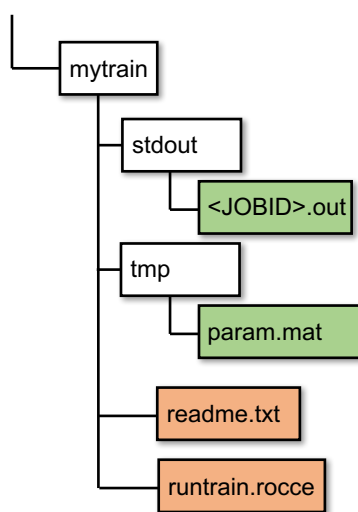
qstat

qstat will output jobs, one per row with current status.

If the job has finished or failed it will **NOT** be displayed in output.

Run *man qstat* for more information

Directory Structure:

6. Check on CHM Train job status

When *qstat* no longer displays the job, it is best to check the *stdout<JOBID>.out* file to verify correct operation.

This can be done with *tail* command which lets one view text files.

ls stdout
tail stdout/<JOBID>.out

A successful job will have the following on the last line:

chm_s22.img exited with code: 0

CHM Train writes output to *tmp* directory and when job completes the script **renames** the directory to *model.* It is a good idea to check for valid output in this *model* directory using the *ls* command

ls model

A successful run will have these files plus three *output_level..* directories:

param.mat, MODEL_level0_stage1.mat, MODEL_level0_stage2.mat, MODEL_level1_stage1.mat
For more information go to: https://github.com/CRBS/chmutil/wiki/createchmtrainjob.py

Directory Structure:

```
mytrain
├── stdout
│   └── <JOBID>.out
└── model
    ├── param.mat
    ├── MODEL_level0_stage1.mat
    ├── MODEL_level0_stage2.mat
    ├── MODEL_level1_stage1.mat
    └── output_level0_stage1
            :
```

Example Output
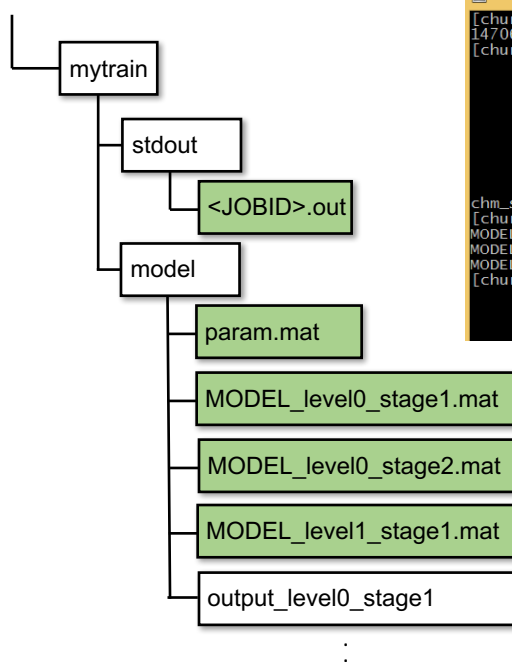
```
[churastest@login-0-0 mytrain]$ ls stdout/
14706.out
[churastest@login-0-0 mytrain]$ tail stdout/14706.out
        Involuntary context switches: 7200
        Swaps: 0
        File system inputs: 262730
        File system outputs: 129952
        Socket messages sent: 0
        Socket messages received: 0
        Signals delivered: 0
        Page size (bytes): 4096
        Exit status: 0
chm_s22.img exited with code: 0
[churastest@login-0-0 mytrain]$ ls model/
MODEL_level0_stage1.mat  output_level0_stage1  param.mat
MODEL_level0_stage2.mat  output_level0_stage2
MODEL_level1_stage1.mat  output_level1_stage1
[churastest@login-0-0 mytrain]$
```

More information on checking job completion can be found here:
https://github.com/CRBS/chmutil/wiki/createchmtrainjob.py