Getting CANDLE running on Biowulf

Whenever Singularity is used (as it is here), bind pertinent directories

```
export SINGULARITY_BINDPATH="/gs3,/gs4,/gs5,/gs6,/gs7,/gs8,/gs9,/gs10/
,/gs11,/gpfs,/spin1,/data,/scratch,/fdb,/lscratch"
```

This is something you might want to put in your ~/.bashrc or ~/.bash_profile so it's automatically loaded upon login.

Run a CANDLE benchmark

This is the most straightforward way to make sure everything is working; you don't have to run it to completion.

(1) Set variables

```
working_dir=<WORKING-DIRECTORY; e.g., ~/test>
gpu_type=<GPU-TYPE-ON-BIOWULF; e.g., k80>
```

(2) Clone CANDLE benchmarks from Github

```
mkdir ~/candle
cd ~/candle
git clone https://github.com/ECP-CANDLE/Benchmarks.git
```

(3) Run benchmark

```
cd $working_dir
echo '#!/bin/bash' > ./jobrequest.sh
echo "module load singularity" >> ./jobrequest.sh
echo "singularity exec --nv /data/classes/candle/candle-gpu.img python
 /data/`whoami`/candle/Benchmarks/Pilot1/P1B1/p1b1_baseline_keras2.py"
 >> ./jobrequest.sh
sbatch --partition=gpu --mem=50G --gres=gpu:$gpu_type:1 ./jobrequest.s
h
```

You should see your job queued or running in SLURM (e.g., squeue -u $(whoami)) and output being produced in $working_dir.

You can also SSH into the node on which the job is running (which is listed under “NODELIST (REASON)” of the squeue command) and even make sure the node’s GPU is being used by running the nvidia-smi command.

Now that you know everything is working you can kill the job using scancel <JOB-ID>, where <JOB-ID> is listed under JOBID of the squeue command. Or if you’re interested, you can let the job run; it should take about 30 min.

Run a grid search (a type of hyperparameter optimization) using output from a test model

In our case the test model just returns random numbers, but this allows you to test the complete workflow you’ll ultimately need for running your own model.

(1) Set variables

```
working_dir=<WORKING-DIRECTORY; e.g., ~/grid_search>
expt_name=<EXPERIMENT-NAME; e.g., random_loss_func>
ntasks=<NTASKS; e.g., 3> # should be greater than 2
job_time=<MAXIMUM-RUNTIME; e.g., 60>
memory=<MAXIMUM-MEMORY-NEEDED; e.g., 10G>
gpu_type=<GPU-TYPE; e.g., k80>
```

(2) Copy grid search template to working directory

```
cp -rp /data/classes/candle/grid-search-template/* $working_dir
```

(3) Edit one file

In $working_dir/swift/swift-job.sh change ./turbine-workflow.sh to swift/turbine-workflow.sh.

(4) “Compile” and run the grid search

```
cd $working_dir
echo '#!/bin/bash' > ./compile_job.sh
echo "module load singularity" >> ./compile_job.sh
echo "singularity exec /data/classes/candle/candle-gpu.img swift/stc-
workflow.sh $expt_name" >> ./compile_job.sh
sbatch -W --time=1 ./compile_job.sh
experiment_id=${expt_name:-experiment}
```

```
sbatch --output=experiments/$experiment_id/output.txt --error=experime
nts/$experiment_id/error.txt --partition=gpu --gres=gpu:$gpu_type:1 --
cpus-per-task=2 --ntasks=$ntasks --mem=$memory --job-name=$experiment_
id --time=$job_time --ntasks-per-node=1 swift/swift-
job.sh $experiment_id
```

Run a grid search using your own model

We already transferred the CANDLE scripts to a local directory (in the above example, to working_dir=~/grid_search). With this directory structure in place, we will now adapt some of these scripts to your own data and model.

(1) Set variables

```
expt_name=<EXPERIMENT-NAME; e.g., my_model>
ntasks=<NTASKS; e.g., 3> # should be greater than 2
job_time=<MAXIMUM-RUNTIME; e.g., 60>
memory=<MAXIMUM-MEMORY-NEEDED; e.g., 10G>
gpu_type=<GPU-TYPE; e.g., k80>
```

(2) Copy over new grid search template scripts

Warning: This will overwrite the two scripts in $working_dir/scripts.

```
cp -f /data/BIDS-
HPC/public/grid_search_template/* $working_dir/scripts
```

(3) Edit files

- **$working_dir/scripts/run_model.sh:** this is a helper shell script that accepts the hyperparameters as command line arguments and calls the model via a Python script, train_model.py, below. Typically you only need to edit the setting of $ml_model_path.
- **$working_dir/scripts/train_model.py:** this is the main, customizable Python script that calls the machine learning model with a particular set of hyperparameters. Its inputs should be a string defining a dictionary of hyperparameters (which is automatically generated in run_model.sh) and the text file containing the result of the model on the data with the current set of hyperparameters.
- **$working_dir/data/dice-params.txt:** text file containing a hyperparameter combination on every line. Can be generated with a script e.g. $working_dir/data/data-generator.py.

(4) “Compile” and run the grid search

These are the same steps as for the test model.

```
cd $working_dir
echo '#!/bin/bash' > ./compile_job.sh
echo "module load singularity" >> ./compile_job.sh
echo "singularity exec /data/classes/candle/candle-gpu.img swift/stc-
workflow.sh $expt_name" >> ./compile_job.sh
sbatch -W --time=1 ./compile_job.sh
experiment_id=${expt_name:-experiment}




sbatch --output=experiments/$experiment_id/output.txt --error=experime
nts/$experiment_id/error.txt --partition=gpu --gres=gpu:$gpu_type:1 --
cpus-per-task=2 --ntasks=$ntasks --mem=$memory --job-name=$experiment_
id --time=$job_time --ntasks-per-node=1 swift/swift-
job.sh $experiment_id
```