Cluster Jobs

Marcaus Cruz

July 2021

1 Preface

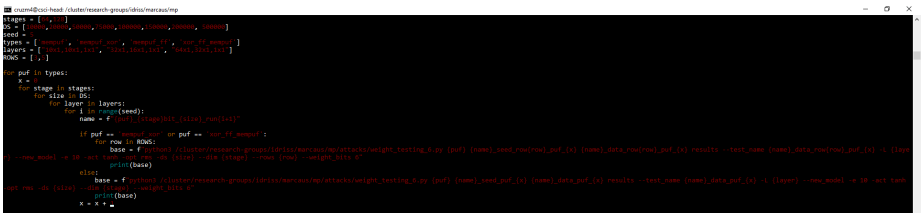
By this point, it's assumed that you have access to the cluster, your environment, WANDB, and the fly repository are all setup and ready to go. This document is meant to help you avoid a few bugs that caused many headaches in this Mempuf project.

2 Writing a job script

Writing a script is simpler than you'd think- at least with the way that Adrian showed me. We're going to be using a python file to help us out.

1. Create a python file with all of the different variable parameters that you want to test in lists.

2. Print out what you'd type on the command line when running a single attack.



You're then going to run this python file while piping the output to "some_file.sh" to convert the output into a bash script. i.e. *python3 script_maker.py >script.sh*

Once that script's created, you need to make sure that you have execute permissions so that fly.py can use the script to run the job on the cluster. The command to allow this is *chmod* in bash.

Note: Quadruple check that the commands printed have zero typos and that this python file has no bugs or syntax errors.

2.1 Other useful commands

- condor_q: Displays logistics of the jobs you have submitted like ID, date and time, and status.
- condor_rm {job number}: Sends a request to remove the job from the cluster.
- condor_status –submitters: Shows all running jobs on cluster and who's running them.

You can always use the -h flag and google to help you figure things out

3 Making sure WanDB logging is accurate

In your 'attack.py' file that initiates the keras model and wandb, there are a few things worth mentioning that will save you lots of headaches.

```
wandb.init(group=args.puf_type,
           project="MP_BM_0",
           entity="idriss-research",
           id=args.test_name,
           config=hyperparameter_defaults)

wandb.log({"Seed": args.seed_file,
           "Train Size": train_size,
           "Test Size": args.test_size,
           "Dim": args.dim,
           "Activation": args.act,
           "Arch": args.L,
           "MB Size": args.mb,
           "Rows": args.rows,
           "Weight Bits": args.weight_bits,
           "Loop": args.ctrl_loop})
```

The *project* parameter dictates what the project name is in wanDB. Each run that calls this file will be logged under the specified project. So if you want runs to be recorded in the same project to ease analysis, you're going to want this name constant throughout the entire job.

Another param to pay attention to is the *id*. If the id is the same for every run, then wanDB will not record any new data. You want to make sure that the id is unique to each run so that each run will be logged. The previous team created a handy method that generates random IDs in the *NetUtils* class if you don't really care what the runs are called.

Note: 'wandb on' (log results online) and 'wandb off' (log results offline) are helpful commands to know. When I'm testing, I make sure wandb is turned off before I run anything. If it's an actual job that I want to record and analyze, I turn it on.

4 Submitting the Job to the Cluster

I'm gonna refer you to Alex's paper because it's very well written and takes you step by step on how to submit a job **here**.