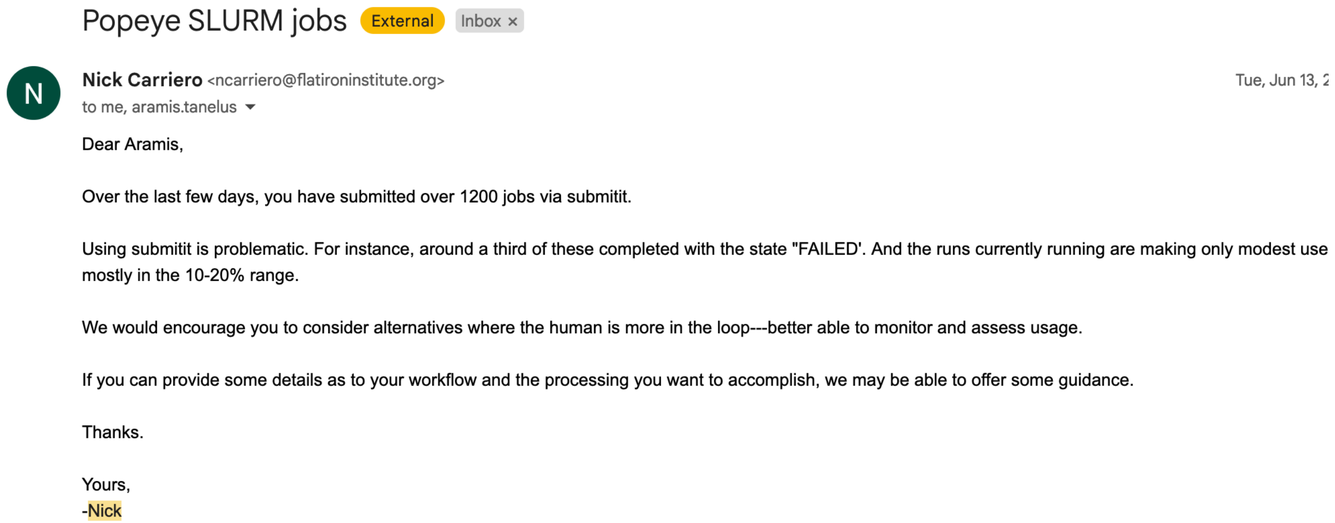
Intro to disBatch

Efficiently schedule and run parallel jobs

2023-08-06

What is disBatch?

- A tool for running many commands across a pool of nodes on the cluster
  - Only requires one resource allocation upfront (saves time in queue)
  - Saves time on initializing shared resources (e.g. copying datasets to local)
  - Saves time spent on reading concerned emails



Popeye SLURM jobs External Inbox ×

N

Nick Carriero <ncarriero@flatironinstitute.org> Tue, Jun 13, 2

to me, aramis.tanelus

Dear Aramis,

Over the last few days, you have submitted over 1200 jobs via submitit.

Using submitit is problematic. For instance, around a third of these completed with the state "FAILED'. And the runs currently running are making only modest use mostly in the 10-20% range.

We would encourage you to consider alternatives where the human is more in the loop---better able to monitor and assess usage.

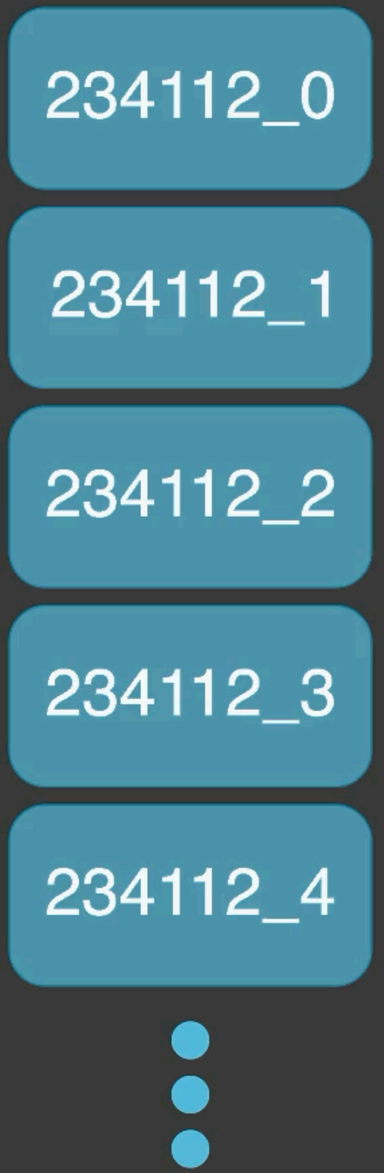
If you can provide some details as to your workflow and the processing you want to accomplish, we may be able to offer some guidance.

Thanks.

Yours,
-Nick

Running Jobs

Queued Jobs



How can I use it?

- If your workflow is scripted (you can type `python my_script.py args...` and it runs) it is trivial
- If your workflow is not scripted, convert it

Interlude: making Python scripts

- ✨Live demo✨ (what could go wrong?)

How can I use it?

- If your workflow is scripted (you can type `python my_script.py args...` and it runs) it is trivial
- If your workflow is not scripted, convert it
- Create a text file with commands to run each of your jobs, one per line
  - I usually make a simple python script for this
- Run disBatch on it

Tips

- You should redirect your logs to make errors and output visible
  - Wrap your commands in `` `( ... ) &>path_to_log.log` ``
  - Optionally, separate stdout and stderr: `` `( ... ) 1>out.log 2>err.log` ``

My formula:

Loop

String construction

Wrap & redirect

```
for base_config_path in base_config_dir.iterdir():
    ft_save_path = model_dir / f"{base_config_path.stem}_no_pretrain"
    log_name = f"{base_config_path.stem}_no_pretrain.log"
    command = (
        "python -u -m gerbilizer "
        f"--config {base_config_path} "
        f"--data {finetune_dataset_path} "
        f"--save-path {ft_save_path}"
    )
    command = f"({command}) &> {log_dir / log_name}"
    lines.append(command)
return lines
```

⋮

```
commands = get_commands()
with open("disbatch_script", "w") as ctx:
    ctx.write("\n".join(commands))
```

Tips

- You should redirect your logs to make errors and output visible
  - Wrap your commands in `` `( ... ) &>path_to_log.log` ``
  - Optionally, separate stdout and stderr: `` `( ... ) 1>out.log 2>err.log` ``
- You can do everything in one line...
  - But if you have to do a lot of setup (loading environments, etc.) it helps to offload everything into a separate script:

```
command = (
    "source ~/.bashrc && "
    "source ~/venvs/general/bin/activate && "
    f"python -u -m gerbilizer --config {base_config_path} --data {pretrain_dataset_path} --save-path {pt_save
)
command = f"( {command} ) &> {log_dir / log_name}"
```

```
command = f"./train_model.sh {base_config_path} {pretrain_dataset_path} {pt_save_path} {log_dir / log_name}"
```

Scheduling with SLURM

```
[atanelus@rustyamd1 ~]$ sbatch -p gpu --gpus-per-task=1 --mem=32GB
-c 6 -t 1-0 disBatch -p disbatch_logs/ disbatch_script
```

Slurm Args & Options:

- -p: Partition (gpu, gen, gen, etc.)
- -n: Number of tasks running in parallel
- -c: Num cores (per task)
- -t: Time limit (for everything)
- --mem: RAM (per task)
- --gpus-per-task: bonus points if you can guess this one

disBatch Args & Options:

- -p: Path for saving logs
- -t: max number of tasks running concurrently per node
- Path to script