Batch Computing on SubMIT: minimizing the horrors of batch computing to make your research go fast and make your collaborators happy

Luca Lavezzo

SubMIT User's Meeting

April 21st, 2025

You, most likely

Your problem:
Running things on your laptop is slow.

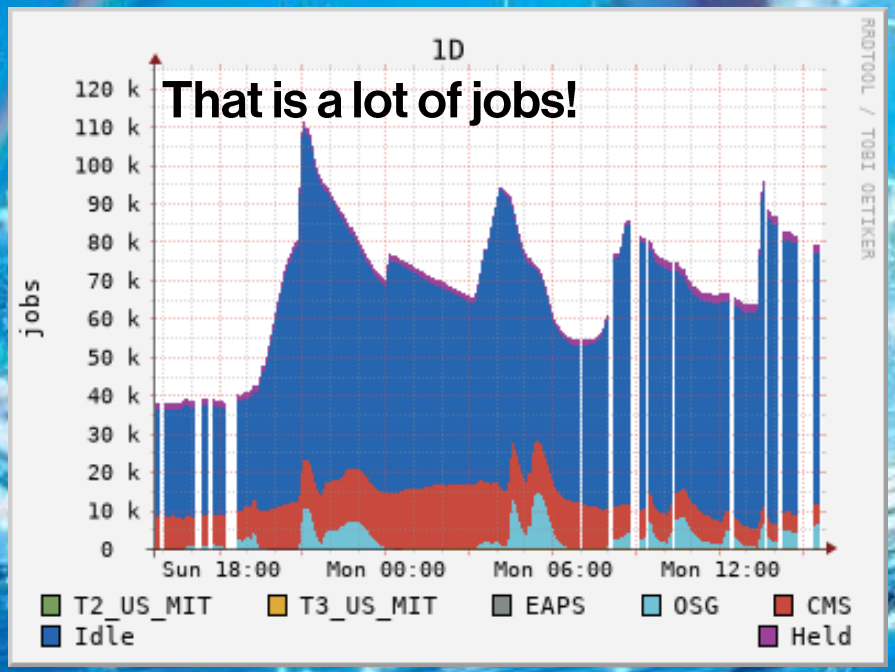
(Your collaborators are waiting for results, and you can't go our for a beer with friends until you finish your project.)

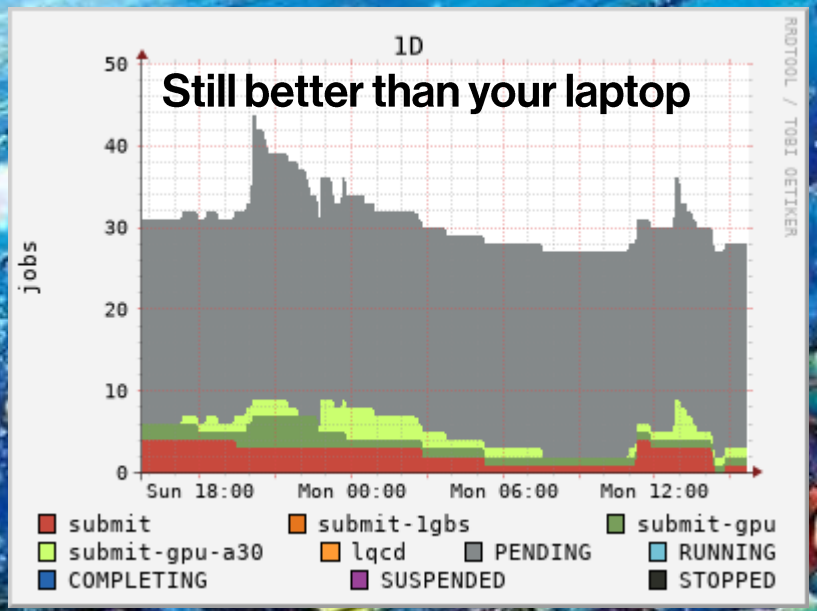
You, most likely

Your problem:
Running things on your laptop is slow.

(Your collaborators are waiting for results, and you can't go our for a beer with friends until you finish your project.)

Here is the promise





The horror of reality

Remote file reads

Unpredictable hardware allocation

Shipping environments

Re-writing framework to process data in parallel

Parsing through thousands of log files

Job resubmissions

Job monitoring

Our solution

To aid you in transitioning to batch computing, we have built an in-house, sophisticated, and effective neural-network-based system that you can chat with

Our solution

To aid you in transitioning to batch computing, we have built an in-house, sophisticated, and effective neural-network-based system that you can chat with

Training procedure:

Input data

Too many hours on

HTCondor

SUBMIT

slurm workload manager

→

Neural Network

→

Output weights

Infinite wisdom about batch jobs, computing, and life

Massachusetts Institute of Technology

Runs on most machines, best performance achieved with lightly roasted beans filtered with a pour-over

Let's get serious: Slurm

Slurm is a way to submit jobs directly to the SubMIT compute nodes

Easiest solution if you need something done quickly!

OK: how do we run a job?

Easy! We just need to:

1. Give some instructions to request resources from Slurm
2. Write a bash script that executes our workflow as a recipe

Let's get serious: Slurm

Slurm is a way to submit jobs directly to the SubMIT compute nodes

Easiest solution if you need something done quickly!

OK: how do we run a job?

Easy! We just need to:

1. Give some instructions to request resources from Slurm
2. Write a bash script that executes our workflow as a recipe
3. SubMIT it with:
   `sbatch submit.sh`

That's it!

submit.sh

```
#!/bin/bash
#
#SBATCH --job-name=job
#SBATCH --output=res_%j.txt
#SBATCH --error=err_%j.txt
#SBATCH --partition=submit
#SBATCH --time=10:00
#SBATCH --mem-per-cpu=100

# activate environment
source ~/.bashrc
conda activate my_env

# run analysis and transfer output
cd /home/submit/lavezzo/nobel-
winning-work/
python do_analysis.py
mv output.root /work/submit/lavezzo/
```

Workflows in Slurm

The power of batch computing is in **parallelizing** your workflows

No standard recipe, depends on your workflow! Typically, make another script to iterate over the parameters or data files you need to process, and submit a job for each

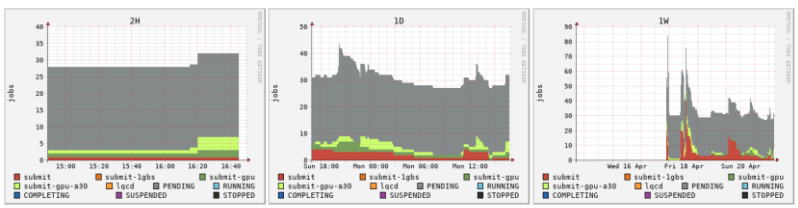
```
#!/bin/bash
#
#SBATCH --job-name=job
#SBATCH --output=res_%j.txt
#SBATCH --error=err_%j.txt
#SBATCH --partition=submit
#SBATCH --time=10:00
#SBATCH --mem-per-cpu=100

# activate environment
source ~/.bashrc
conda activate my_env

# run analysis and transfer output
cd /home/submit/lavezzo/nobel-winning-work/
python do_analysis.py --parameter N
mv output.root /work/submit/lavezzo/
```

Data, scripts, environments, software, etc. you access on SubMIT login nodes are also available on SubMIT compute notes on Slurm.

Available resources: ~1000 cores, ~20 GPUs

Slurm allocates resources fairly amongst users: do not SubMIT large jobs on the log-in nodes! I will kill them! And email you to run them on Slurm!

Mon Apr 21 2025 16:50:11 GMT-0400 (Eastern Daylight Time)



User	Pending	Completing	Suspended	Stopped	Running	MIT					Total
						submit	submit-1gbs	submit-gpu	submit-gpu-a30	lqcd	
lavezzo	0	0	0	0	0	0	0	0	0	0	0
mmould	0	0	0	0	0	0	0	0	0	0	0
amitit	0	0	0	0	0	0	0	0	0	0	0
svitale	0	0	0	0	4	0	0	1	3	0	4
tslatyer	0	0	0	0	1	1	0	0	0	0	1
dingq30	0	0	0	0	1	0	0	0	1	0	1
hughhig	0	0	0	0	1	0	0	1	0	0	1
xiuyuan	0	0	0	0	0	0	0	0	0	0	0
julia785	25	0	0	0	0	0	0	0	0	0	25
Total	25	0	0	0	7	1	0	2	4	0	32

monitoring page

HTCondor: the ‘H’ stands for ‘Horror’

HTCondor is the worst way to submit jobs

However, it has by far the most resources available:
the OSG, the MIT Tier 2 and the Tier 3 clusters can guarantee a *couple of thousand of cores at any given time, for free (!)*

HTCondor: the ‘H’ stands for ‘Horror’

HTCondor is the worst way to submit jobs

However, it has by far the most resources available:
the OSG, the MIT Tier 2 and the Tier 3 clusters can guarantee a *couple of thousand of cores at any given time, for free (!)*

OK: Why is it so bad?

Jobs are submitted to **remote** machines.

You do **not** have access to your data, environments, software, or scripts.

(Also: huge variability of hardware and software on the clusters, and each has its own kinks)

HTCondor: the ‘H’ stands for ‘Horror’

HTCondor is the worst way to submit jobs

However, it has by far the most resources available:
the OSG, the MIT Tier 2 and the Tier 3 clusters can guarantee a *couple of thousand of cores at any given time, for free (!)*

OK: Why is it so bad?

Jobs are submitted to **remote** machines.

You do **not** have access to your data, environments, software, or scripts.

(Also: huge variability of hardware and software on the clusters, and each has its own kinks)

OK: What am I to do?

We have to transfer everything {data, env., software, scripts} to the job, by hand.

This is painful.

Wisdom

We have tried to step through the conceptual hurdles in this tutorial, which has examples you can copy & paste.

There are also some real-list scripts available on our GitHub repo from real-life users, which can be starting points for your work.

I will now explain the concepts at a slightly higher level.

More details are explained in our User's Guide.

SubMITting to HTCondor

Similarly to Slurm, we need to:

1. Request HTCondor the resources we want
2. Give an executable script that runs through our workflow

In HTCondor, these two are different scripts:

submit.sub

```
universe       = vanilla
request_disk   = 1024
executable     = script.sh
arguments      = $(ProcId)
output         = $(ClusterId).$(ProcId).out
error          = $(ClusterId).$(ProcId).err
log            = $(ClusterId).$(ProcId).log
+DESIRED_Sites ="mit_tier3"
queue 1
```

script.sh

```
#!/bin/bash

echo "I am a HTCondor job!"
echo "I have landed in $(hostname)"
echo "I have received parameter $1"
echo "That's all!"
```

Wisdom 1: transferring inputs

You cannot read things from SubMIT directly! i.e. /home, /work, /ceph are not accessible from remote machines. You need to transfer whatever you need as input to the remote machine your job is running to.

Via submission script

You can add the following to your submission script. This is limited to 250MB. This will transfer your files to the compute node.

```
transfer_input_files    = <your comma-separated list of files>
```

Via XRootD

This allows you to read files remotely, but you need to set up certificates to authenticate yourself to the network. You can then read data remotely in your executable script. For example,

```
xrdcp root://submit50.mit.edu//data/user/l/lavezzo/inputs.txt .
```

Wisdom 2: transferring outputs

Presumably you're not running jobs as part of your evil plan to melt the ice caps and kill the polar bears, but you need the outputs of your job. You need to transfer these back to SubMIT.

Via submission script

Adding the following to your submission script will copy the outputs of your job back to SubMIT automatically.

```
should_transfer_files   = YES
when_to_transfer_output = ON_EXIT
```

Via XRootD

We can write out to SubMIT with XRootD as well:

```
xrdcp output.txt root://submit50.mit.edu//data/user/l/lavezzo/outputs/
```

Wisdom 3: software

Again since the HTCondor nodes don't have access to the subMIT storage areas, you need to distribute your software to the worker-node. This is further complicated that the OS on each worker-node or cluster may be different.

Via CVMFS

CVMFS is mounted on subMIT and all clusters connected to subMIT via HTCondor, and supports the distribution of containers. In order to use a container in your jobs, you can specify which image you want via, e.g.

```
+SingularityImage       = "/cvmfs/singularity.opensciencegrid.org/htc/rocky:9"
```

You can even distribute your own containers to CVMFS.

Via transfer

If you don't need a lot of software, and you can package it (perhaps by compiling it in a way that is self-contained), you can transfer it via the methods outlines in the previous section: either through the submission script or HTCondor.

What's best for me?

Run a long, heavy job, with O(10-100) cores

Iterate over O(10-1000) parameters or input files with a fairly lightweight job

Run anything with more than a handful of cores for more than a couple hours

→ **Slurm**

Iterate over many (> 1000) parameters or input files, and I'm willing to spend some extra time setting up the framework

→ **HTCondor**

Concluding thoughts

I hope to have transferred a fraction of my infinite wisdom about batch computing

More can be found on the SubMIT User's Guide, Tutorials, and Examples, including:

- Specific instructions to request resources
- Commands to monitor jobs
- Detailed recommendations for common hurdles in HTCondor jobs
- Example, toy scripts to learn about batch computing
- Specific, real-life examples of submission scrips provided by Users