EE3801 Data Engineering Principles

Parallel Data Processing

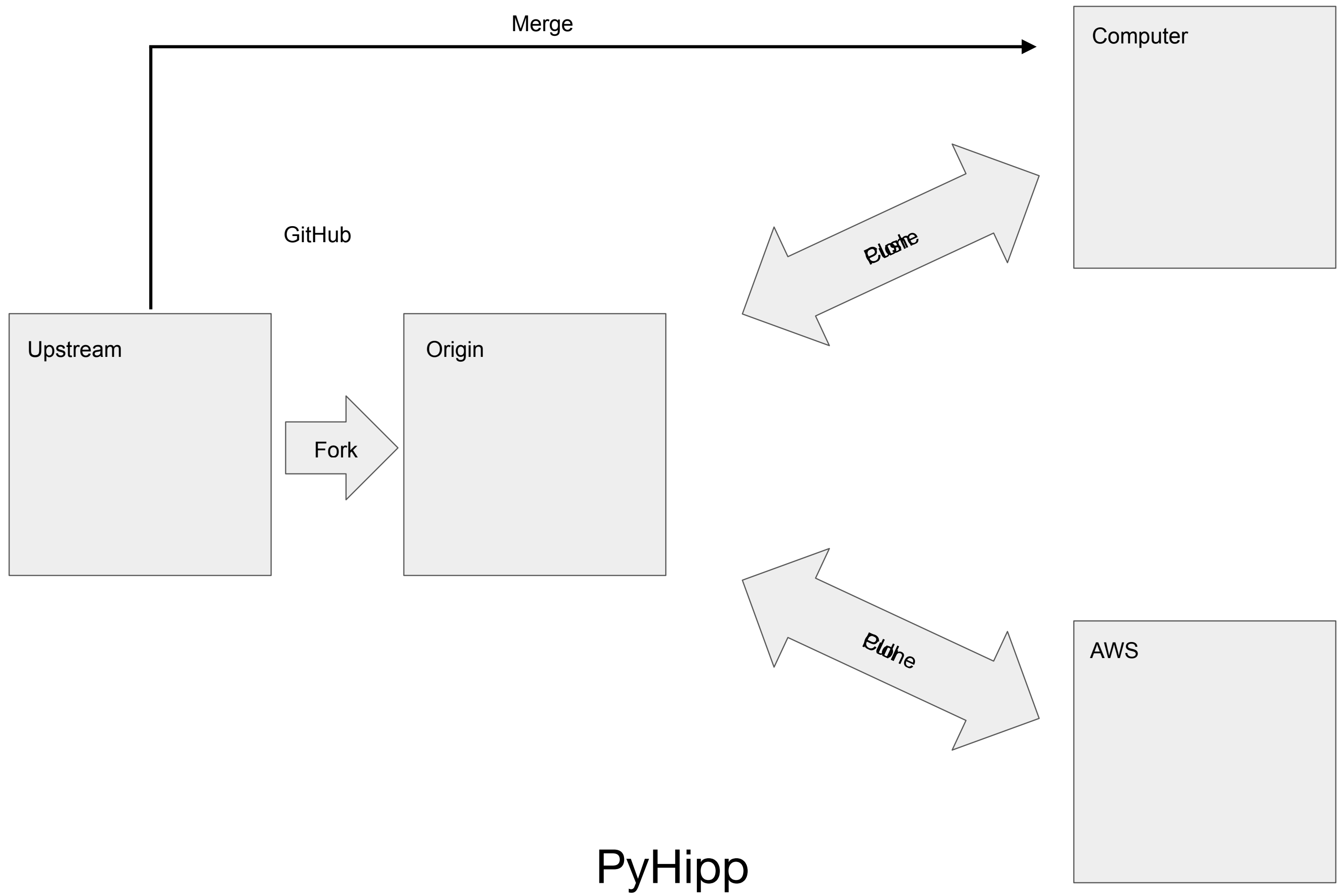
Parallel Data Processing

Parallel Processing

- Lab 5 techniques
- Parallelizing jobs
- Increasing memory per job
- Terminating cluster head nodes automatically
- Slurm job dependencies

GitHub Repositories

Forked Repositories

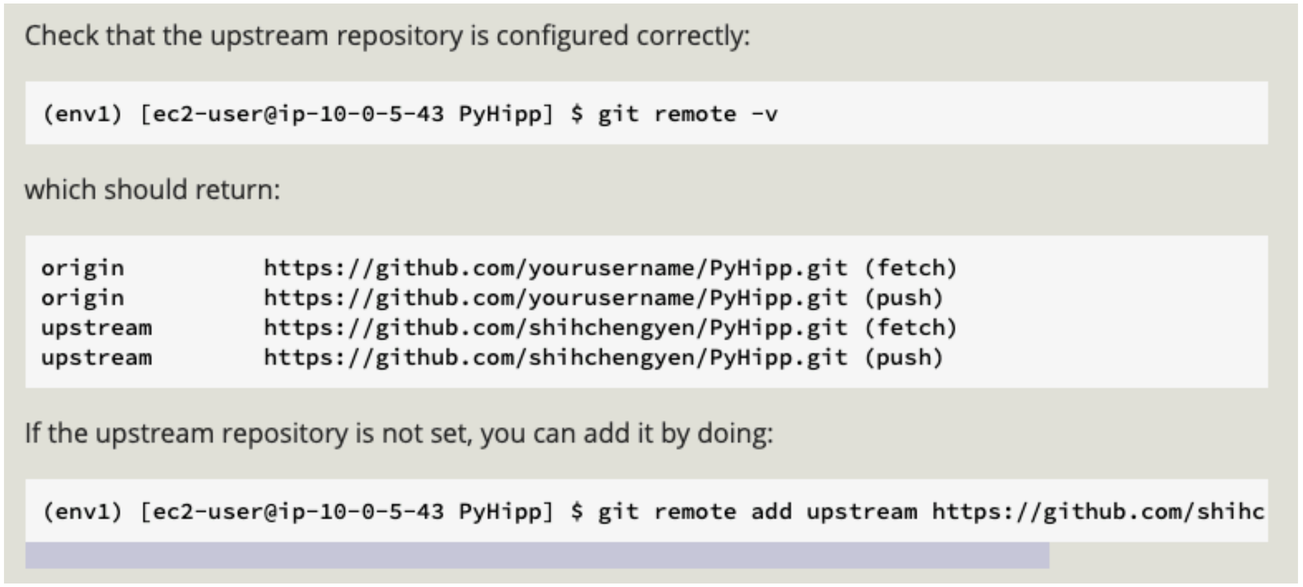


PyHipp

GitHub Repositories

Forked Repositories

On AWS



Check that the upstream repository is configured correctly:

```
(env1) [ec2-user@ip-10-0-5-43 PyHipp] $ git remote -v
```

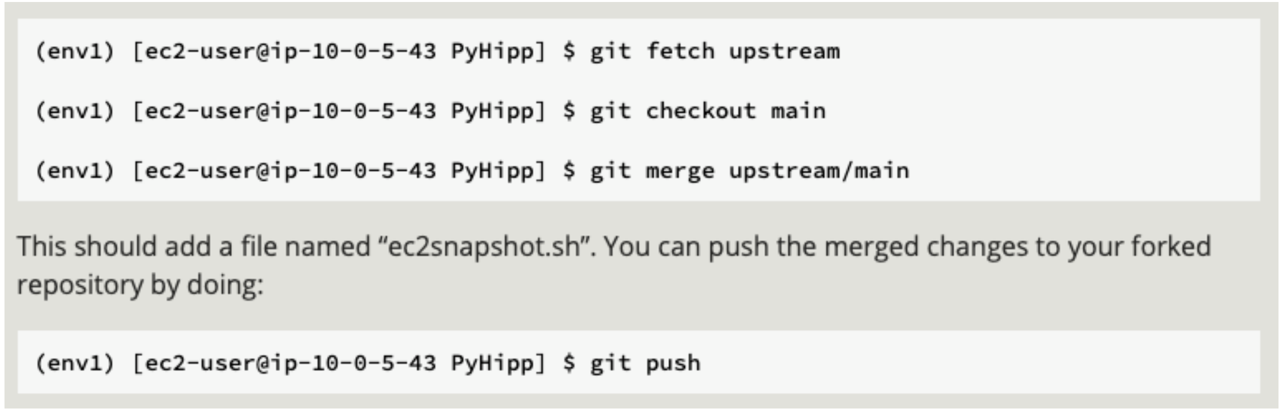
which should return:

```
origin          https://github.com/yourusername/PyHipp.git (fetch)
origin          https://github.com/yourusername/PyHipp.git (push)
upstream        https://github.com/shihchengyen/PyHipp.git (fetch)
upstream        https://github.com/shihchengyen/PyHipp.git (push)
```

If the upstream repository is not set, you can add it by doing:

```
(env1) [ec2-user@ip-10-0-5-43 PyHipp] $ git remote add upstream https://github.com/shihc
```

On AWS



```
(env1) [ec2-user@ip-10-0-5-43 PyHipp] $ git fetch upstream

(env1) [ec2-user@ip-10-0-5-43 PyHipp] $ git checkout main

(env1) [ec2-user@ip-10-0-5-43 PyHipp] $ git merge upstream/main
```

This should add a file named "ec2snapshot.sh". You can push the merged changes to your forked repository by doing:

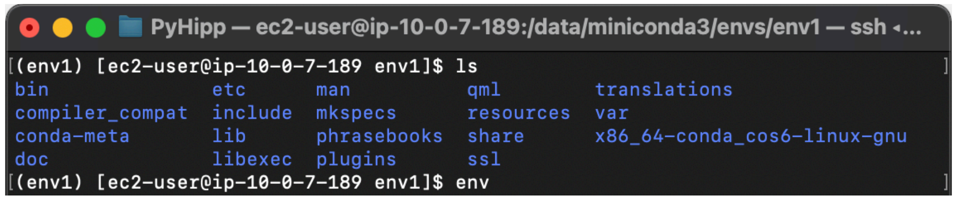
```
(env1) [ec2-user@ip-10-0-5-43 PyHipp] $ git push
```

Optimizing Parallel Processing

Lab 5 Techniques (Git and Conda)

- GitHub command line (git clone)

- Conda environments (e.g. /data/miniconda3/envs/env1)



PyHipp — ec2-user@ip-10-0-7-189:/data/miniconda3/envs/env1 — ssh ◂...

```
(env1) [ec2-user@ip-10-0-7-189 env1]$ ls
bin              etc      man          qml        translations
compiler_compat  include  mkspecs      resources  var
conda-meta       lib      phrasebooks  share      x86_64-conda_cos6-linux-gnu
doc              libexec  plugins      ssl
(env1) [ec2-user@ip-10-0-7-189 env1]$ env
```

- Software is installed in /data/miniconda3/envs/env1/lib/python3.8/site-packages

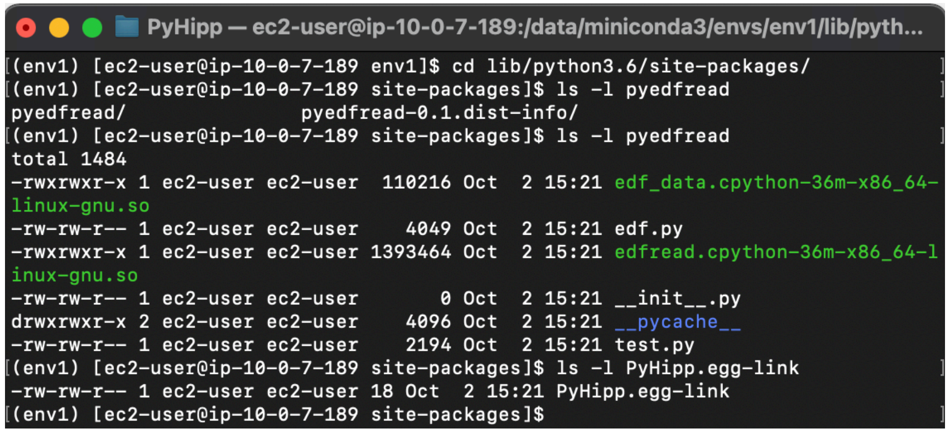
- Python checks this directory when functions are called

Optimizing Parallel Processing

Lab 5 Techniques (Conda)

- pip install .

- copies code to /data/miniconda3/envs/env1/lib/python3.8/site-packages, e.g. pyedfread


PyHipp — ec2-user@ip-10-0-7-189:/data/miniconda3/envs/env1/lib/pyth...

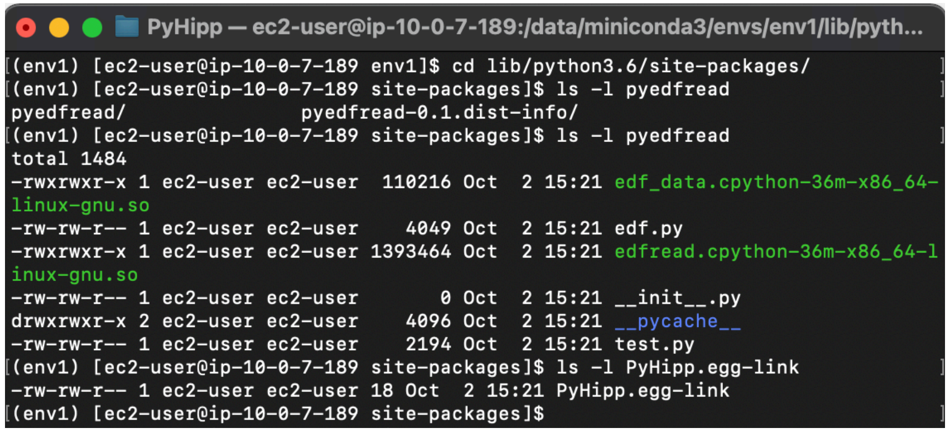
```
[(env1) [ec2-user@ip-10-0-7-189 env1]$ cd lib/python3.6/site-packages/
[(env1) [ec2-user@ip-10-0-7-189 site-packages]$ ls -l pyedfread
pyedfread/                 pyedfread-0.1.dist-info/
[(env1) [ec2-user@ip-10-0-7-189 site-packages]$ ls -l pyedfread
total 1484
-rwxrwxr-x 1 ec2-user ec2-user  110216 Oct  2 15:21 edf_data.cpython-36m-x86_64-
linux-gnu.so
-rw-rw-r-- 1 ec2-user ec2-user    4049 Oct  2 15:21 edf.py
-rwxrwxr-x 1 ec2-user ec2-user 1393464 Oct  2 15:21 edfread.cpython-36m-x86_64-l
inux-gnu.so
-rw-rw-r-- 1 ec2-user ec2-user       0 Oct  2 15:21 __init__.py
drwxrwxr-x 2 ec2-user ec2-user    4096 Oct  2 15:21 __pycache__
-rw-rw-r-- 1 ec2-user ec2-user    2194 Oct  2 15:21 test.py
[(env1) [ec2-user@ip-10-0-7-189 site-packages]$ ls -l PyHipp.egg-link
-rw-rw-r-- 1 ec2-user ec2-user 18 Oct  2 15:21 PyHipp.egg-link
[(env1) [ec2-user@ip-10-0-7-189 site-packages]$
```

- Each time the code changes, you have to do:
  - git pull $\longleftarrow$ (updates /data/src/pyedfread)
  - pip install . $\longleftarrow$ (copies to /data/miniconda3/envs/env1/lib/python3.8/site-packages
- You can then start using the new function(s)

Optimizing Parallel Processing

Lab 5 Techniques (Bash)

- pip install -e .
- Links to /data/miniconda3/envs/env1/lib/python3.8/site-packages, e.g. PyHipp


PyHipp — ec2-user@ip-10-0-7-189:/data/miniconda3/envs/env1/lib/pyth...

```
(env1) [ec2-user@ip-10-0-7-189 env1]$ cd lib/python3.6/site-packages/
(env1) [ec2-user@ip-10-0-7-189 site-packages]$ ls -l pyedfread
pyedfread/                 pyedfread-0.1.dist-info/
(env1) [ec2-user@ip-10-0-7-189 site-packages]$ ls -l pyedfread
total 1484
-rwxrwxr-x 1 ec2-user ec2-user  110216 Oct  2 15:21 edf_data.cpython-36m-x86_64-
linux-gnu.so
-rw-rw-r-- 1 ec2-user ec2-user    4049 Oct  2 15:21 edf.py
-rwxrwxr-x 1 ec2-user ec2-user 1393464 Oct  2 15:21 edfread.cpython-36m-x86_64-l
inux-gnu.so
-rw-rw-r-- 1 ec2-user ec2-user       0 Oct  2 15:21 __init__.py
drwxrwxr-x 2 ec2-user ec2-user    4096 Oct  2 15:21 __pycache__
-rw-rw-r-- 1 ec2-user ec2-user    2194 Oct  2 15:21 test.py
(env1) [ec2-user@ip-10-0-7-189 site-packages]$ ls -l PyHipp.egg-link
-rw-rw-r-- 1 ec2-user ec2-user 18 Oct  2 15:21 PyHipp.egg-link
(env1) [ec2-user@ip-10-0-7-189 site-packages]$
```

- Each time the code changes, all you have to do is:
  - git pull $\longleftarrow$ (updates /data/src/pyedfread)
- You can then start using the new function(s)

Optimizing Parallel Processing

Lab 5 Techniques (Compute Nodes)

Compute Node

Last Digit	Type	Last Digit	Type	Last Digit	Type
0	m5.4xlarge	4	r5n.2xlarge	8	r5.2xlarge
1	z1d.2xlarge	5	r5b.2xlarge	9	r5a.2xlarge
2	m5a.4xlarge	6	r5d.2xlarge		
3	r5dn.2xlarge	7	r5ad.2xlarge		

Instance Size	vCPU	Memory (GiB)	Instance Storage (GB)	Network Bandwidth (Gbps)***	EBS Bandwidth (Mbps)
m5.large	2	8	EBS-Only	Up to 10	Up to 4,750
m5.xlarge	4	16	EBS-Only	Up to 10	Up to 4,750
m5.2xlarge	8	32	EBS-Only	Up to 10	Up to 4,750
m5.4xlarge	16	64	EBS-Only	Up to 10	4,750

Instance	vCPU	Mem (GiB)	Networking Performance (Gbps)***	SSD Storage (GB)
z1d.large	2	16	Up to 10	1 x 75 NVMe SSD
z1d.xlarge	4	32	Up to 10	1 x 150 NVMe SSD
z1d.2xlarge	8	64	Up to 10	1 x 300 NVMe SSD

Lab 3

Snapshot

- Copy data snapshot

Click on the “Owned By Me” button to reveal the drop-down menu and select “Public Snapshots”. Copy and paste the following snapshot id: snap-05c0a250a5fa4d56d, and then hit “Return”. You should see a snapshot with the description “ee3801-2021-data” listed and selected.

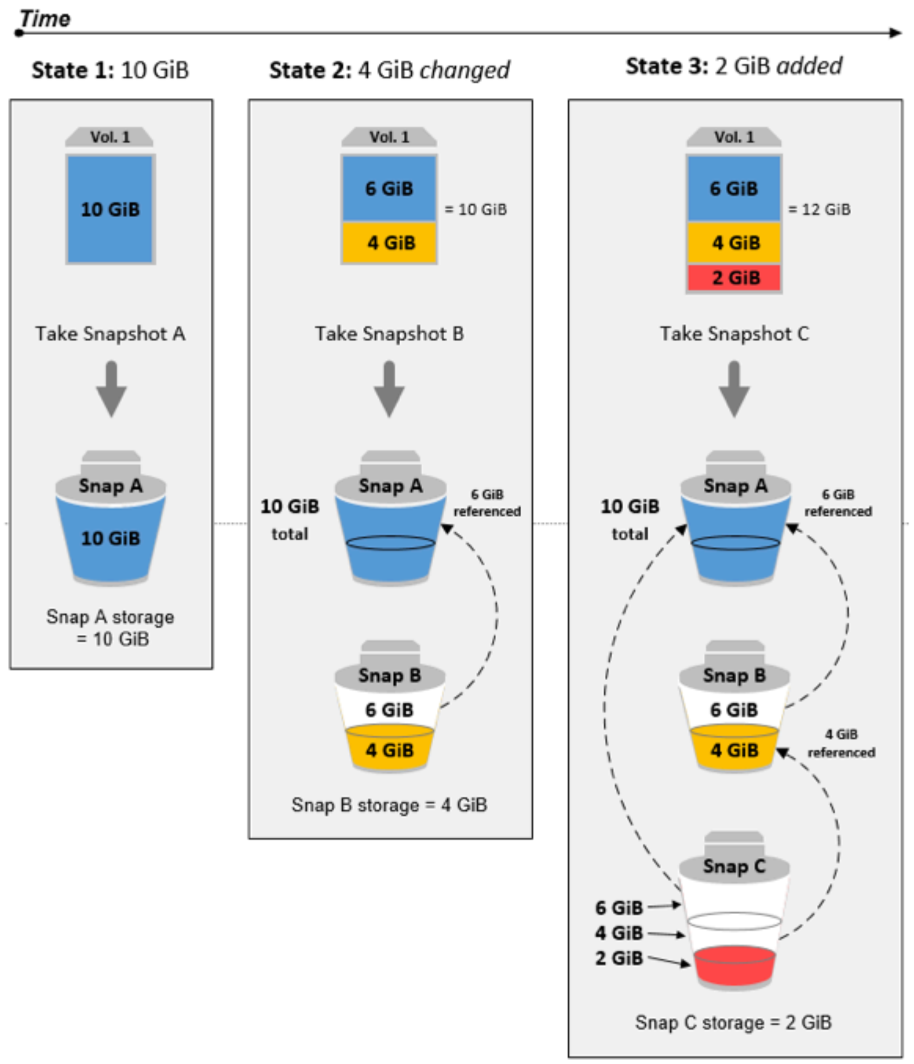
Click on the “Actions” button, and select the Copy command. In the window that appears, replace the description with “data”, make sure the “Encrypt this snapshot” option is NOT selected, and then click the “Copy” button. You should see a message that the snapshot is being copied.

Click on the “Public Snapshots” button to select the “Owned By Me” option, and then click on the “x” icon in the search field to remove the snapshot id. You should now see a snapshot with a size of 1000 GiB with the description “data”. If the snapshot is not selected, select it, and you should see more information for the snapshot shown in the panel at the bottom of the window. Move your cursor over the text “Snapshot ID”, which will cause a “Copy to clipboard” icon to appear. Click on it to copy the Snapshot ID.

- Subsequent snapshots will be faster

AWS ParallelCluster

EBS Snapshots

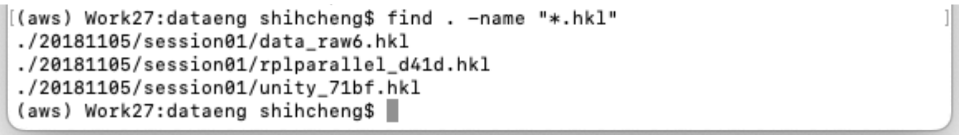


- Incremental backups
- Only blocks that have changed after your most recent snapshot are saved
- Unchanged blocks are referenced to previous snapshot
- New data will add to the storage required
- Total storage for 3 snapshots is 10 + 4 + 2 = 16 GiB

Optimizing Parallel Processing

Lab 5 Techniques (Bash)

- ipython (easier to use than regular Python, e.g. ls, cd)
- tail -f file (to display text as they are added to the file)
- find (to find files and directories)
    - find . -name “.hkl”

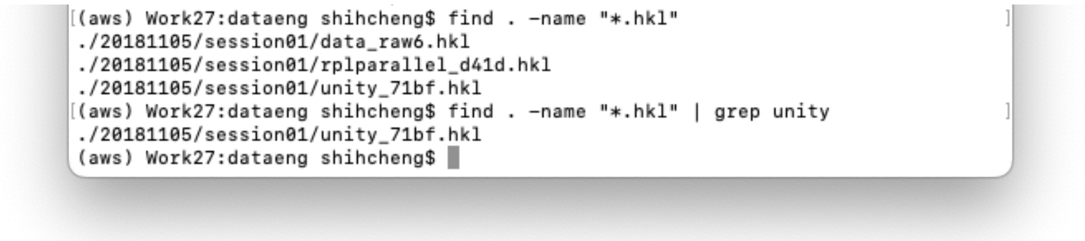


```
[(aws) Work27:dataeng shihcheng$ find . -name "*.hkl"
./20181105/session01/data_raw6.hkl
./20181105/session01/rplparallel_d41d.hkl
./20181105/session01/unity_71bf.hkl
(aws) Work27:dataeng shihcheng$ 
```

Optimizing Parallel Processing

Lab 5 Techniques (Bash)

- | (pipe output from one program to input of another program)

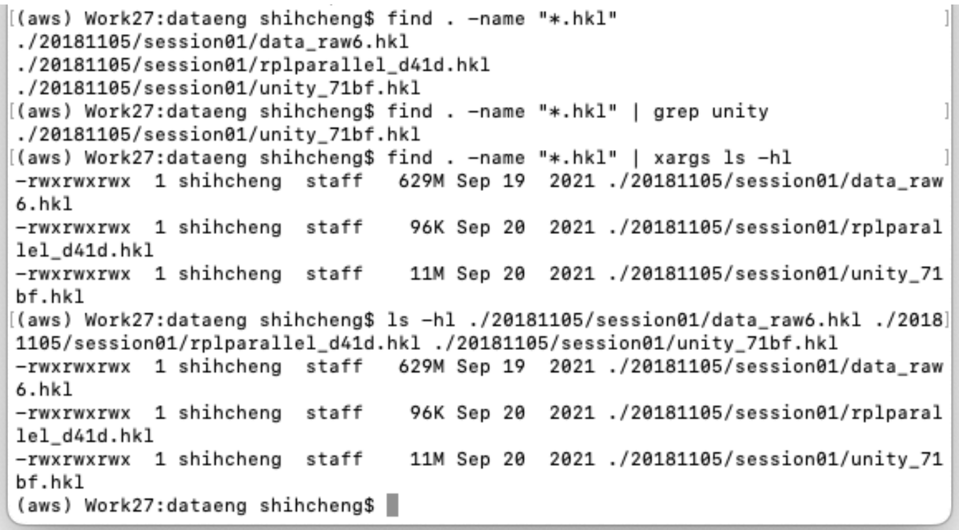


```
(aws) Work27:dataeng shihcheng$ find . -name "*.hkl"
./20181105/session01/data_raw6.hkl
./20181105/session01/rplparallel_d41d.hkl
./20181105/session01/unity_71bf.hkl
(aws) Work27:dataeng shihcheng$ find . -name "*.hkl" | grep unity
./20181105/session01/unity_71bf.hkl
(aws) Work27:dataeng shihcheng$ 
```

Optimizing Parallel Processing

Lab 5 Techniques (Bash)

- xargs (converts output from one program to arguments for another program)



```
(aws) Work27:dataeng shihcheng$ find . -name "*.hkl"
./20181105/session01/data_raw6.hkl
./20181105/session01/rplparallel_d41d.hkl
./20181105/session01/unity_71bf.hkl
(aws) Work27:dataeng shihcheng$ find . -name "*.hkl" | grep unity
./20181105/session01/unity_71bf.hkl
(aws) Work27:dataeng shihcheng$ find . -name "*.hkl" | xargs ls -hl
-rwxrwxrwx  1 shihcheng  staff   629M Sep 19  2021 ./20181105/session01/data_raw
6.hkl
-rwxrwxrwx  1 shihcheng  staff    96K Sep 20  2021 ./20181105/session01/rplparal
lel_d41d.hkl
-rwxrwxrwx  1 shihcheng  staff    11M Sep 20  2021 ./20181105/session01/unity_71
bf.hkl
(aws) Work27:dataeng shihcheng$ ls -hl ./20181105/session01/data_raw6.hkl ./2018
1105/session01/rplparallel_d41d.hkl ./20181105/session01/unity_71bf.hkl
-rwxrwxrwx  1 shihcheng  staff   629M Sep 19  2021 ./20181105/session01/data_raw
6.hkl
-rwxrwxrwx  1 shihcheng  staff    96K Sep 20  2021 ./20181105/session01/rplparal
lel_d41d.hkl
-rwxrwxrwx  1 shihcheng  staff    11M Sep 20  2021 ./20181105/session01/unity_71
bf.hkl
(aws) Work27:dataeng shihcheng$ 
```

Optimizing Parallel Processing

Lab 5 Techniques (Slurm)

- srun --pty /bin/bash (get shell running on compute node)
- #SBATCH --time=24:00:00   # walltime (time limit for job)

Parallel Data Processing

Parallel Processing

- Lab 4 techniques
- Parallelizing jobs
- Increasing memory per job
- Terminating cluster head nodes automatically
- Slurm job dependencies

Data Processing on AWS

Slurm Script (Called from /data/picasso/20181105)

```
#!/bin/bash

# Submit this script with: sbatch <this-filename>

#SBATCH —time=24:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH -J "pipe"   # job name

## /SBATCH -p general # partition (queue)
#SBATCH -o pipe-slurm.%N.%j.out # STDOUT
#SBATCH -e pipe-slurm.%N.%j.err # STDERR

# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE

python -u -c "import PyHipp as pyh; \
import DataProcessingTools as DPT; \
import os; \
import time; \
t0 = time.time(); \
print(time.localtime()); \
DPT.objects.processDirs(dirs=None, objtype=pyh.RPLParallel, saveLevel=1); \
DPT.objects.processDirs(dirs=None, objtype=pyh.RPLSplit, channel=[9, 31, 34, 56, 72, 93, 119, 120]); \
DPT.objects.processDirs(dirs=None, objtype=pyh.RPLLFP, saveLevel=1); \
DPT.objects.processDirs(dirs=None, objtype=pyh.RPLHighPass, saveLevel=1); \
DPT.objects.processDirs(dirs=None, objtype=pyh.Unity, saveLevel=1); \
pyh.EDFSplit(); \
os.chdir('session01'); \
pyh.aligning_objects(); \
pyh.raycast(1); \
DPT.objects.processDirs(level='channel', cmd='import PyHipp as pyh; from PyHipp import mountain_batch; mountain_batch.mountain_batch(); from PyHipp
import export_mountain_cells; export_mountain_cells.export_mountain_cells();'); \
print(time.localtime()); \
print(time.time()-t0);"

aws sns publish --topic-arn arn:aws:sns:ap-southeast-1:123456789012:awsnotify --message "JobDone"
```

Parallel Data Processing

Parallel Processing

Serial Pipeline

Dependencies	RPLParallel	RPLSplit	RPLLFP	RPLHighPass	Unity	EDFSplit
RPLParallel						
RPLSplit						
RPLLFP		√				
RPLHighPass		√				
Unity	√					
EDFSplit	√					
aligning_objects	√				√	√
raycast					√	√
mountain_batch				√		

Parallel Pipeline

RPLParallel	RPLSplit
Unity	RPLLFP
EDFSplit	RPLHighPass
aligning_objects	mountain_batch
raycast	

Optimizing Parallel Processing

Parallel Processing (2 jobs)

Time

RPLParallel	RPLSplit		
Unity	session01	channel001	rplraw_xxxx.hkl
EDFSplit	sessioneye	channel002	rplraw_xxxx.hkl
aligning_objects		…	…
raycast		channel124	rplraw_xxxx.hkl
	RPLLFP		
	session01	channel001	rpllfp_xxxx.hkl
	sessioneye	channel002	rpllfp_xxxx.hkl
		…	…
		channel124	rpllfp_xxxx.hkl
	RPLHighPass		
	session01	channel001	rplhighpass_xxxx.hkl
	sessioneye	channel002	rplhighpass_xxxx.hkl
		…	…
		channel124	rplhighpass_xxxx.hkl
	mountain_batch		
	session01	channel001	firings.mda
		channel002	firings.mda
		…	…
		channel124	firings.mda

Optimizing Parallel Processing

Parallel Processing (5 jobs)

Time

RPLParallel	RPLSplit		RPLSplit		RPLSplit		RPLSplit	
Unity	session01	channel001	session01	channel033	session01	channel065	session01	channel097
EDFSplit	sessioneye	channel002	sessioneye	channel034	sessioneye	channel066	sessioneye	channel098
aligning_objects		…		…		…		…
raycast		channel032		channel064		channel096		channel124
	RPLLFP		RPLLFP		RPLLFP		RPLLFP	
	session01	channel001	session01	channel033	session01	channel065	session01	channel097
	sessioneye	channel002	sessioneye	channel034	sessioneye	channel066	sessioneye	channel098
		…		…		…		…
		channel032		channel064		channel096		channel124
	RPLHighPass		RPLHighPass		RPLHighPass		RPLHighPass	
	session01	channel001	session01	channel033	session01	channel065	session01	channel097
	sessioneye	channel002	sessioneye	channel034	sessioneye	channel066	sessioneye	channel098
		…		…		…		…
		channel032		channel064		channel096		channel124
	mountain_batch		mountain_batch		mountain_batch		mountain_batch	
	session01	channel001	session01	channel033	session01	channel065	session01	channel097
		channel002		channel034		channel066		channel098
		…		…		…		…
		channel032		channel064		channel096		channel124

Parallel Data Processing

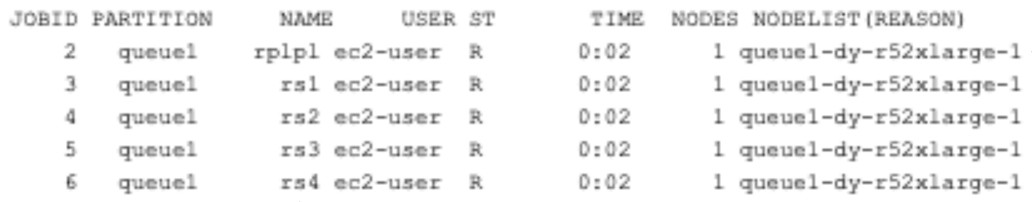
Parallel Processing

- Lab 4 techniques
- Parallelizing jobs
- Increasing memory per job
- Terminating cluster head nodes automatically
- Slurm job dependencies

Optimizing Parallel Processing

Parallel Processing (5 jobs)

Increasing memory per job



```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
    2    queue1    rplpl ec2-user  R       0:02      1 queue1-dy-r52xlarge-1
    3    queue1      rs1 ec2-user  R       0:02      1 queue1-dy-r52xlarge-1
    4    queue1      rs2 ec2-user  R       0:02      1 queue1-dy-r52xlarge-1
    5    queue1      rs3 ec2-user  R       0:02      1 queue1-dy-r52xlarge-1
    6    queue1      rs4 ec2-user  R       0:02      1 queue1-dy-r52xlarge-1
```

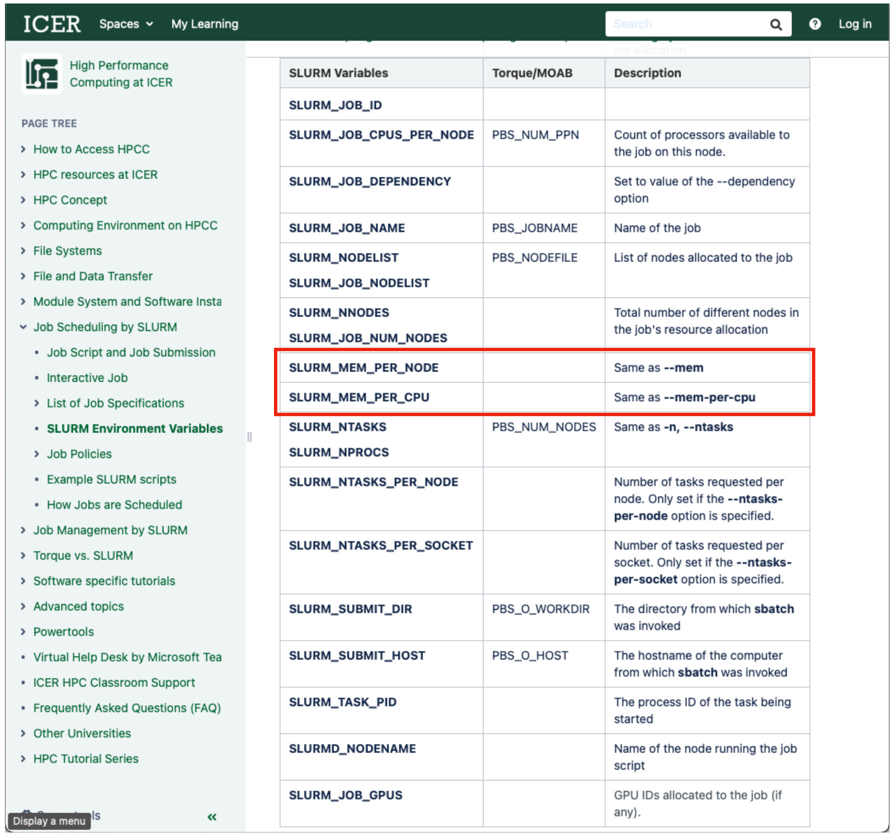
64 GB RAM

40 GB RAM

Optimizing Parallel Processing

Parallel Processing (5 jobs)

Increasing memory per job



SLURM Variables	Torque/MOAB	Description
SLURM_JOB_ID		
SLURM_JOB_CPUS_PER_NODE	PBS_NUM_PPN	Count of processors available to the job on this node.
SLURM_JOB_DEPENDENCY		Set to value of the --dependency option
SLURM_JOB_NAME	PBS_JOBNAME	Name of the job
SLURM_NODELIST SLURM_JOB_NODELIST	PBS_NODEFILE	List of nodes allocated to the job
SLURM_NNODES SLURM_JOB_NUM_NODES		Total number of different nodes in the job's resource allocation
SLURM_MEM_PER_NODE		Same as **--mem**
SLURM_MEM_PER_CPU		Same as **--mem-per-cpu**
SLURM_NTASKS SLURM_NPROCS	PBS_NUM_NODES	Same as **-n, --ntasks**
SLURM_NTASKS_PER_NODE		Number of tasks requested per node. Only set if the **--ntasks-per-node** option is specified.
SLURM_NTASKS_PER_SOCKET		Number of tasks requested per socket. Only set if the **--ntasks-per-socket** option is specified.
SLURM_SUBMIT_DIR	PBS_O_WORKDIR	The directory from which **sbatch** was invoked
SLURM_SUBMIT_HOST	PBS_O_HOST	The hostname of the computer from which **sbatch** was invoked
SLURM_TASK_PID		The process ID of the task being started
SLURMD_NODENAME		Name of the node running the job script
SLURM_JOB_GPUS		GPU IDs allocated to the job (if any).

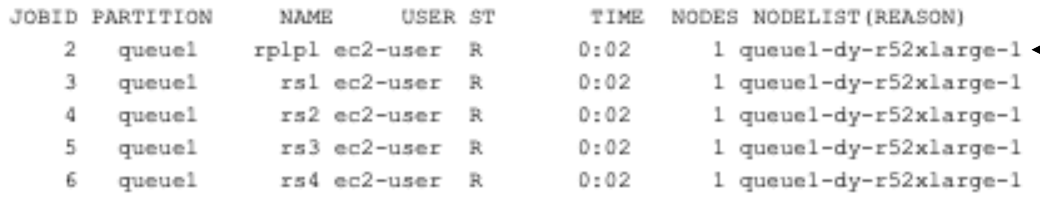
Does not work on AWS!

https://wiki.hpcc.msu.edu/display/ITH/SLURM+Environment+Variables

Optimizing Parallel Processing

Parallel Processing (5 jobs)

Increasing memory per job

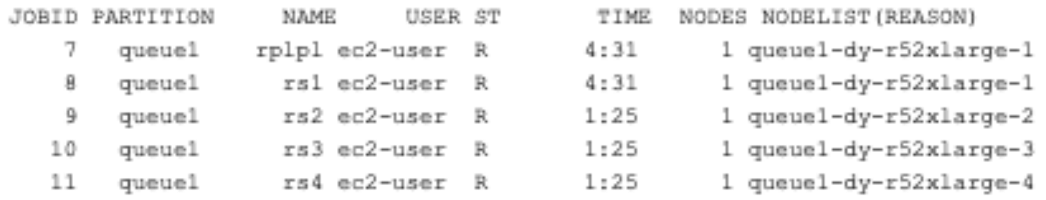


```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
    2    queue1    rplpl ec2-user  R       0:02      1 queue1-dy-r52xlarge-1
    3    queue1      rs1 ec2-user  R       0:02      1 queue1-dy-r52xlarge-1
    4    queue1      rs2 ec2-user  R       0:02      1 queue1-dy-r52xlarge-1
    5    queue1      rs3 ec2-user  R       0:02      1 queue1-dy-r52xlarge-1
    6    queue1      rs4 ec2-user  R       0:02      1 queue1-dy-r52xlarge-1
```

64 GB RAM

40 GB RAM

```
#SBATCH --time=24:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH --cpus-per-task=5             # number of processors per task
#SBATCH -J "rs1"   # job name
```



```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
    7    queue1    rplpl ec2-user  R       4:31      1 queue1-dy-r52xlarge-1
    8    queue1      rs1 ec2-user  R       4:31      1 queue1-dy-r52xlarge-1
    9    queue1      rs2 ec2-user  R       1:25      1 queue1-dy-r52xlarge-2
   10    queue1      rs3 ec2-user  R       1:25      1 queue1-dy-r52xlarge-3
   11    queue1      rs4 ec2-user  R       1:25      1 queue1-dy-r52xlarge-4
```

Parallel Data Processing

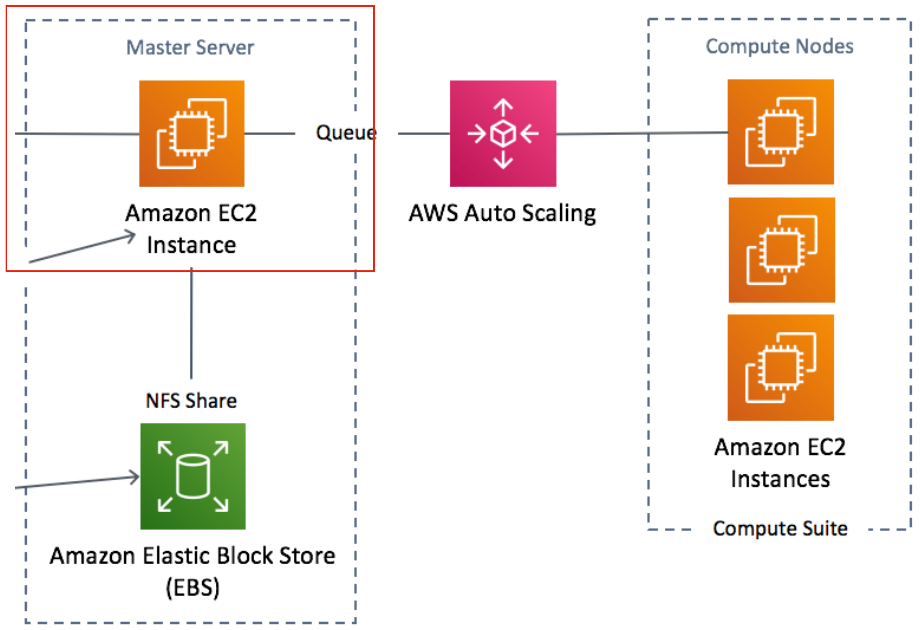
Parallel Processing

- Lab 4 techniques
- Parallelizing jobs
- Increasing memory per job
- Terminating cluster head nodes automatically
- Slurm job dependencies

AWS ParallelCluster

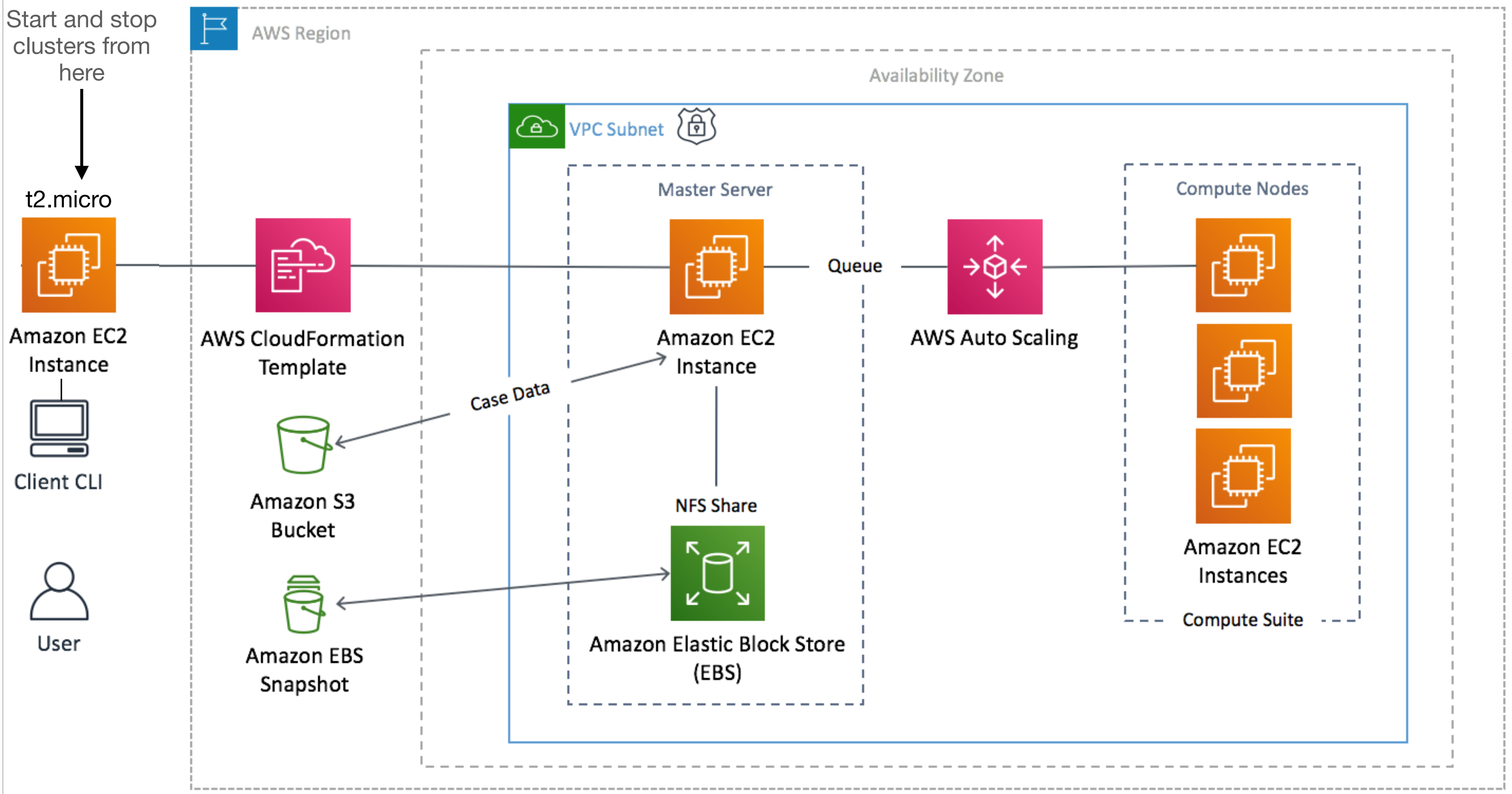
Master Server

- Master node
- Main login interface to the cluster
- Jobs are submitted and managed here
- Always running once the cluster is created
- Configured using the file ~/cluster-config.yaml



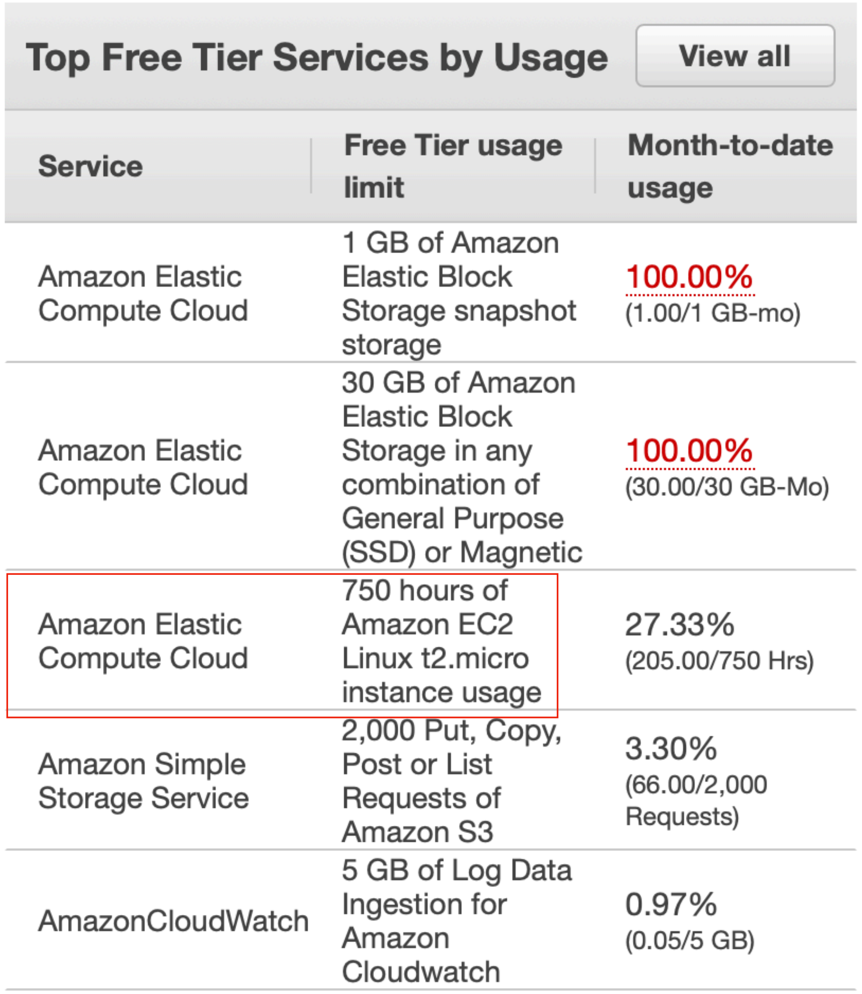
Data Processing on AWS

Cluster Deletion



Data Processing on AWS

Cluster Deletion



Top Free Tier Services by Usage

View all

Service	Free Tier usage limit	Month-to-date usage
Amazon Elastic Compute Cloud	1 GB of Amazon Elastic Block Storage snapshot storage	100.00% (1.00/1 GB-mo)
Amazon Elastic Compute Cloud	30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic	100.00% (30.00/30 GB-Mo)
Amazon Elastic Compute Cloud	750 hours of Amazon EC2 Linux t2.micro instance usage	27.33% (205.00/750 Hrs)
Amazon Simple Storage Service	2,000 Put, Copy, Post or List Requests of Amazon S3	3.30% (66.00/2,000 Requests)
AmazonCloudWatch	5 GB of Log Data Ingestion for Amazon Cloudwatch	0.97% (0.05/5 GB)

750/24 = 31.25 days

Data Processing on AWS

Cluster Deletion

- Normal procedure:
  - receive notification that jobs have been completed
  - update_snapshot.sh data 2
  - receive snapshot completion notification from CloudWatch
  - pcluster delete-cluster -n MyCluster01

Data Processing on AWS

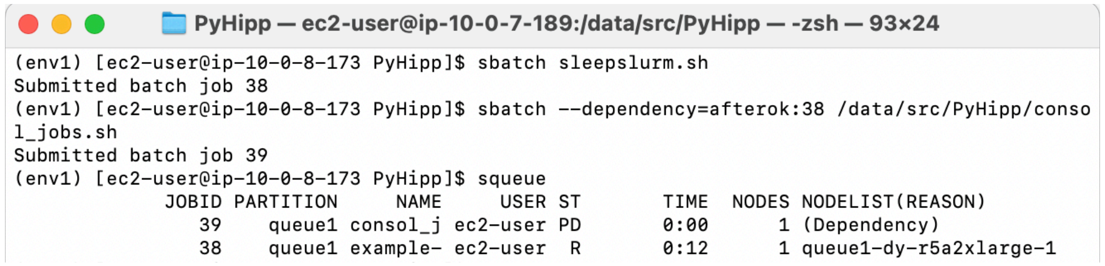
Cluster Deletion

- Could have EC2 instance constantly check on queue in Head Node to see if it is empty
  - pcluster ssh -i ~/MyKeyPair.pem -n MyCluster01 “squeue”
  - and then create snapshot and delete cluster
  - Very inefficient

Data Processing on AWS

Cluster Deletion

- Can use slurm job dependencies to ssh to EC2 instance to issue pcluster delete-cluster command



PyHipp — ec2-user@ip-10-0-7-189:/data/src/PyHipp — -zsh — 93×24

```
(env1) [ec2-user@ip-10-0-8-173 PyHipp]$ sbatch sleepslurm.sh
Submitted batch job 38
(env1) [ec2-user@ip-10-0-8-173 PyHipp]$ sbatch --dependency=afterok:38 /data/src/PyHipp/conso
l_jobs.sh
Submitted batch job 39
(env1) [ec2-user@ip-10-0-8-173 PyHipp]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                39    queue1 consol_j ec2-user PD       0:00      1 (Dependency)
                38    queue1 example- ec2-user  R       0:12      1 queue1-dy-r5a2xlarge-1
```

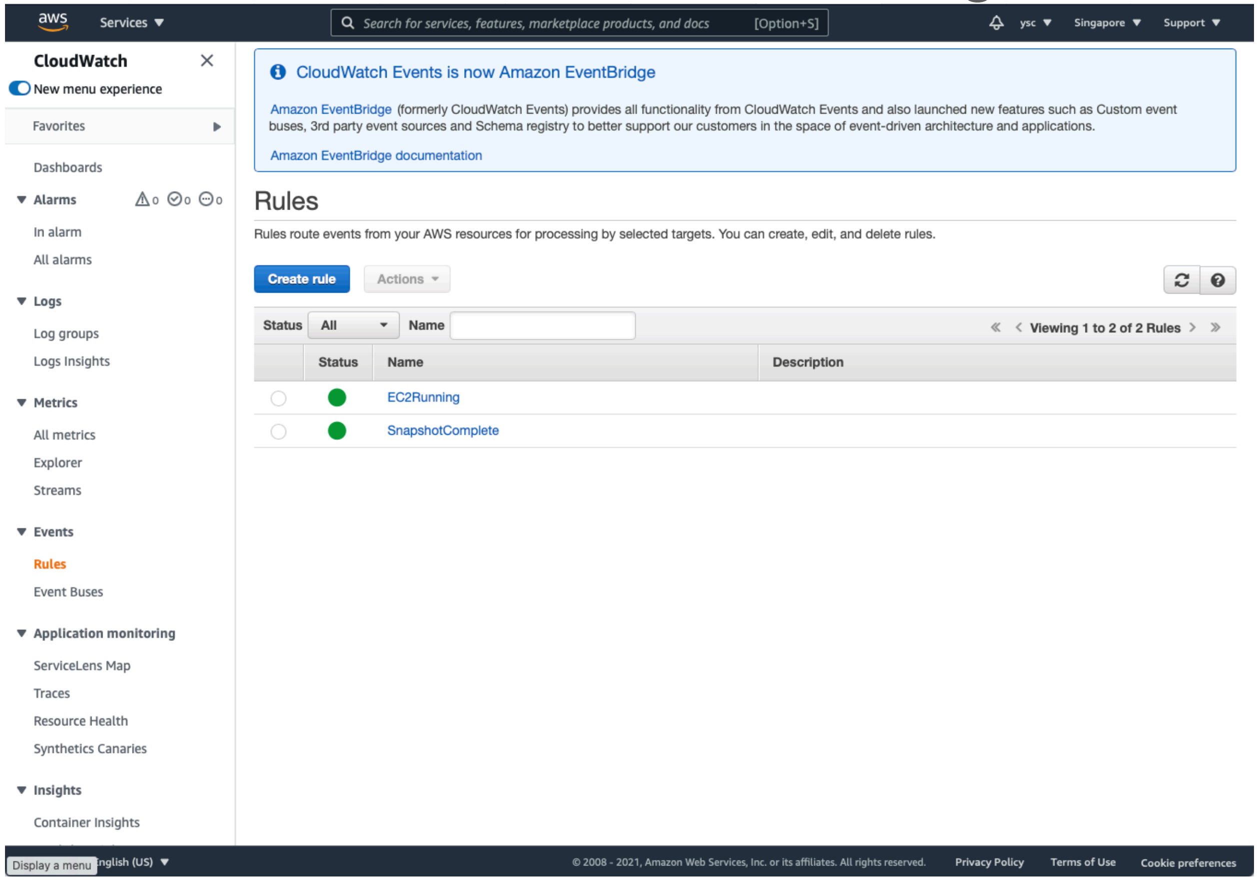
  - ssh -i /data/MyKeyPair.pem ec2-user@xx.xx.xx.xx “~/update_snapshot.sh data 2 MyCluster01; pcluster delete-cluster -n MyCluster01”

  - but we want to delete the cluster only after the snapshot is complete

Data Processing on AWS

Cluster Deletion

Can we use CloudWatch/EventBridge instead?



CloudWatch Events is now Amazon EventBridge

Amazon EventBridge (formerly CloudWatch Events) provides all functionality from CloudWatch Events and also launched new features such as Custom event buses, 3rd party event sources and Schema registry to better support our customers in the space of event-driven architecture and applications.

Amazon EventBridge documentation

Rules

Rules route events from your AWS resources for processing by selected targets. You can create, edit, and delete rules.

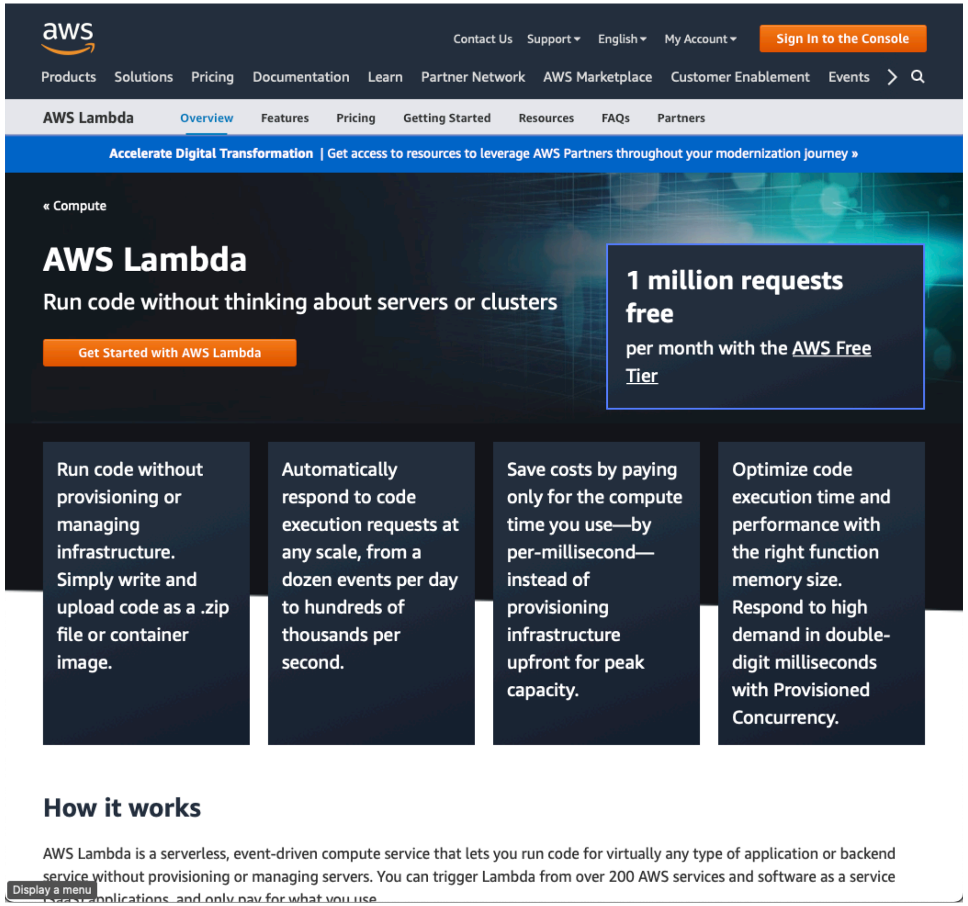
Status: All | Name: | Viewing 1 to 2 of 2 Rules

	Status	Name	Description
○	●	EC2Running	
○	●	SnapshotComplete	

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data Processing on AWS

Cluster Deletion



AWS Lambda

Run code without thinking about servers or clusters

1 million requests free

per month with the AWS Free Tier

Run code without provisioning or managing infrastructure. Simply write and upload code as a .zip file or container image.

Automatically respond to code execution requests at any scale, from a dozen events per day to hundreds of thousands per second.

Save costs by paying only for the compute time you use—by per-millisecond—instead of provisioning infrastructure upfront for peak capacity.

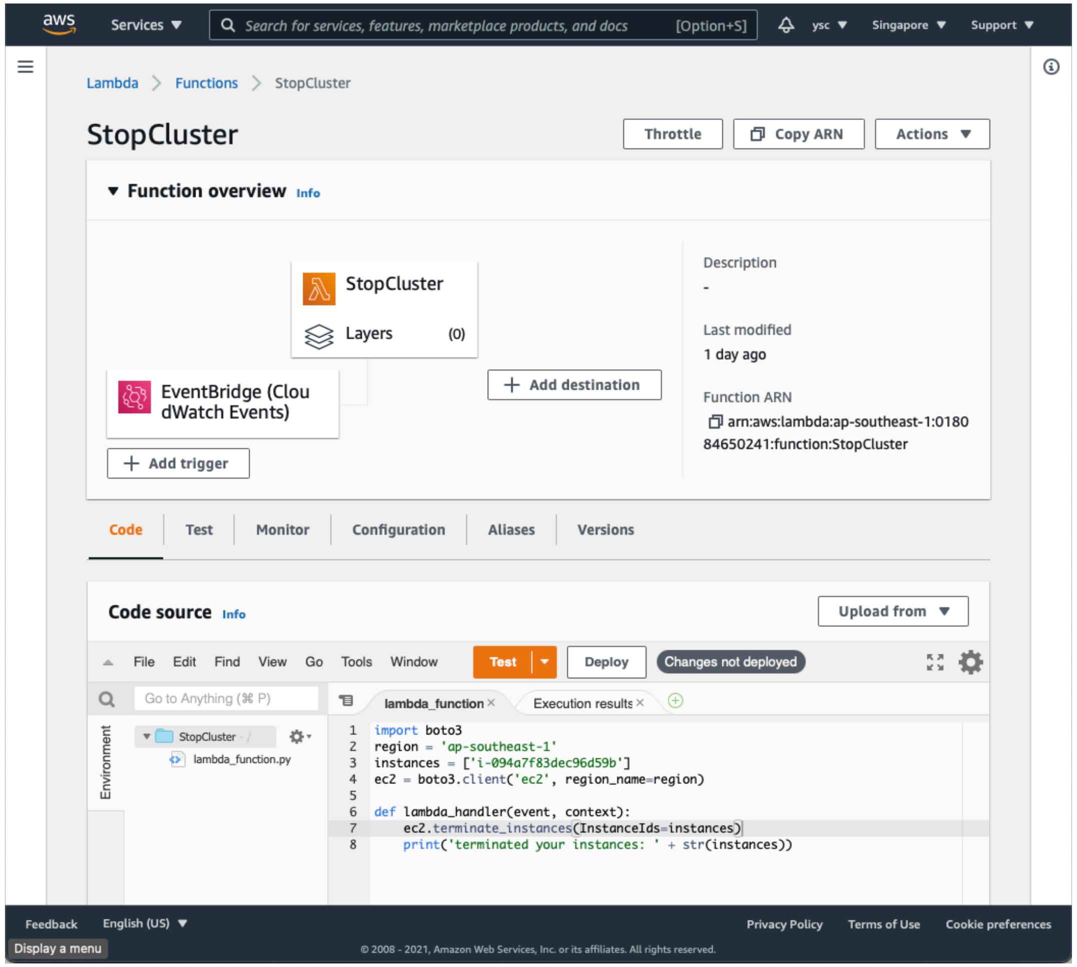
Optimize code execution time and performance with the right function memory size. Respond to high demand in double-digit milliseconds with Provisioned Concurrency.

How it works

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. You can trigger Lambda from over 200 AWS services and software as a service (SaaS) applications, and only pay for what you use.

Data Processing on AWS

Cluster Deletion



StopCluster

Description
-

Last modified
1 day ago

Function ARN
arn:aws:lambda:ap-southeast-1:018084650241:function:StopCluster

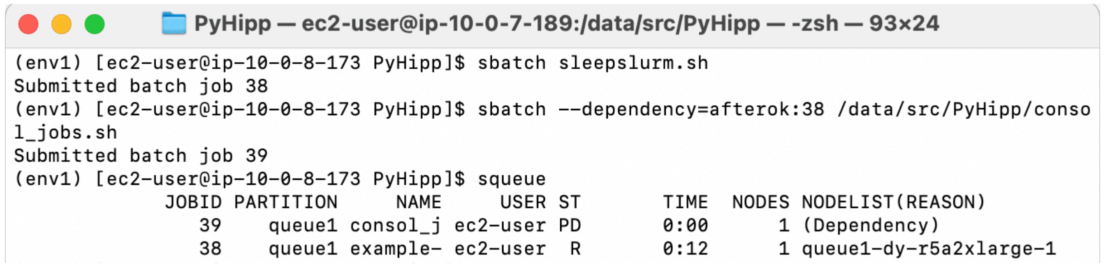
```python
import boto3
region = 'ap-southeast-1'
instances = ['i-094a7f83dec96d59b']
ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
    ec2.terminate_instances(InstanceIds=instances)
    print('terminated your instances: ' + str(instances))
```

Data Processing on AWS

Cluster Deletion

- Use slurm job dependencies to initiate snapshot:


PyHipp — ec2-user@ip-10-0-7-189:/data/src/PyHipp — -zsh — 93×24

```
(env1) [ec2-user@ip-10-0-8-173 PyHipp]$ sbatch sleepslurm.sh
Submitted batch job 38
(env1) [ec2-user@ip-10-0-8-173 PyHipp]$ sbatch --dependency=afterok:38 /data/src/PyHipp/conso
l_jobs.sh
Submitted batch job 39
(env1) [ec2-user@ip-10-0-8-173 PyHipp]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                39    queue1 consol_j ec2-user PD       0:00      1 (Dependency)
                38    queue1 example- ec2-user  R       0:12      1 queue1-dy-r5a2xlarge-1
```

- ssh -i /data/MyKeyPair.pem ec2-user@xx.xx.xx.xx "~/update_snapshot.sh data 2 MyCluster01"

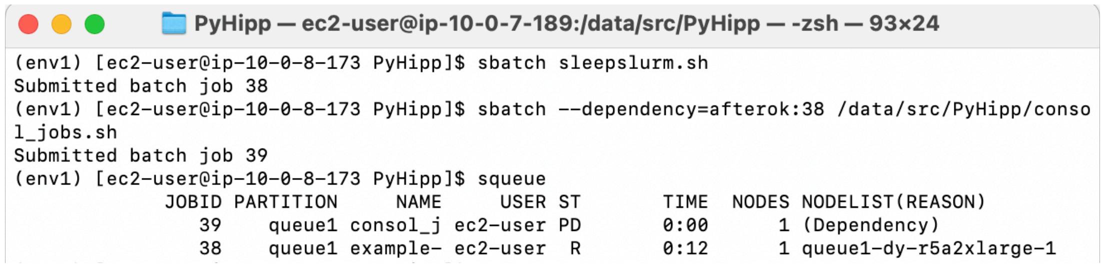
- CloudWatch watch for snapshot completion

- Lambda function used to delete head node

Data Processing on AWS

Cluster Deletion

- Use slurm job dependencies to initiate snapshot:



PyHipp — ec2-user@ip-10-0-7-189:/data/src/PyHipp — -zsh — 93×24

```
(env1) [ec2-user@ip-10-0-8-173 PyHipp]$ sbatch sleepslurm.sh
Submitted batch job 38
(env1) [ec2-user@ip-10-0-8-173 PyHipp]$ sbatch --dependency=afterok:38 /data/src/PyHipp/conso
l_jobs.sh
Submitted batch job 39
(env1) [ec2-user@ip-10-0-8-173 PyHipp]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                39    queue1 consol_j ec2-user PD       0:00      1 (Dependency)
                38    queue1 example- ec2-user  R       0:12      1 queue1-dy-r5a2xlarge-1
```

- ssh -o StrictHostKeyChecking=no -i /data/MyKeyPair.pem ec2-user@xx.xx.xx.xx "source ~/.bash_profile; pcluster update-compute-fleet --status STOP_REQUESTED -n MyCluster01; ~/update_snapshot.sh data 2 MyCluster01"

- CloudWatch watch for snapshot completion

- Lambda function used to delete head node

- Still have to call pcluster delete-cluster to remove from list

Parallel Data Processing

Parallel Processing

- Lab 4 techniques
- Parallelizing jobs
- Increasing memory per job
- Terminating cluster head nodes automatically
- Slurm job dependencies

Data Processing on AWS

Cluster Deletion

consol_jobs.sh

```
#!/bin/sh
temp1=($(squeue))
cmd1="sbatch --dependency=afterok:"
counter1=0
for i in "${temp1[@]}"; do
        if [[ "$i" == "queue1" ]]; then
                        id1=${temp1[$counter1-1]}
                        cmd1="${cmd1}${id1}:"
        fi
        counter1=$((counter1+1))
done
cmd1=${cmd1::-1}
cmd1="${cmd1} /data/src/PyHipp/ec2snapshot.sh"

echo $($cmd1)
eval $cmd1
```

```
(env1) [ec2-user@ip-10-0-3-231 20181105]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               233    queue1    rplpl ec2-user CF       0:02      1 compute-dy-r5large-1
               234    queue1      rse ec2-user CF       0:02      1 compute-dy-r5large-1
(env1) [ec2-user@ip-10-0-3-231 20181105]$
```

Data Processing on AWS

Cluster Deletion

consol_jobs.sh

```
#!/bin/sh
temp1=($(squeue))
cmd1="sbatch --dependency=afterok:"
counter1=0
for i in "${temp1[@]}"; do
       if [[ "$i" == "queue1" ]]; then
                      id1=${temp1[$counter1-1]}
                      cmd1="${cmd1}${id1}:"
       fi
       counter1=$((counter1+1))
done
cmd1=${cmd1::-1}
cmd1="${cmd1} /data/src/PyHipp/ec2snapshot.sh"

echo $($cmd1)
eval $cmd1
```

```
(env1) [ec2-user@ip-10-0-9-92 data]$ for i in "${temp1[@]}"; do echo $i; done
JOBID
PARTITION
NAME
USER
ST
TIME
NODES
NODELIST(REASON)
233
queue1
rplpl
ec2-user
CF
0:02
1
compute-dy-r5large-1
234
queue1
rse
ec2-user
CF
0:02
1
compute-dy-r5large-1
(env1) [ec2-user@ip-10-0-9-92 data]$
```

```
cmd1="sbatch —dependency=afterok:
cmd1="sbatch —dependency=afterok:233:
cmd1="sbatch —dependency=afterok:233:234:
cmd1="sbatch —dependency=afterok:233:234
cmd1="sbatch —dependency=afterok:233:234 /data/src/PyHipp/ec2snapshot.sh
```

Optimizing Parallel Processing

Lab 6 Techniques

- Bash
  - GitHub command line (git pull, git push, git fetch upstream, etc.)
  - Create shell scripts for frequently performed series of functions
  - Run commands remotely via ssh, e.g.

    ```
    pcluster ssh -i ~/MyKeyPair.pem -n MyCluster01 squeue
    ```
- Slurm
  - Use file dependencies to parallelize into 2 jobs (RPLParallel, RPLSplit)
  - Use directory hierarchy to parallelize into 5 jobs (e.g. session01/array01)
  - Use slurm parameter --cpus-per-task=5 to increase memory available to individual jobs
  - Use slurm job dependencies to run jobs in sequence
- AWS
  - Use AWS Lambda to run scripts without servers
  - Use AWS Lambda, EventBridge, and small EC2 instance to terminate head node automatically

Optimizing Parallel Processing

Lab Instructions

- Lab 6 Instructions:
  - https://ee3801.github.io/Lab6/instruction.html
- Submit to Canvas (Lab 6->Lab 6A & Lab 6->Lab 6B)
- Submit in PDF format
- Name the files Lab6A_YourName.pdf and Lab6B_YourName.pdf
- Part A due on Monday (Oct 23) 2 pm
- Part B due on Wednesday (Oct 25) 9 pm

Questions?