

EE3801 Data Engineering Principles

Review

Optimizing Parallel Processing

Lab 8 Techniques

Lab 7

```
(env1) [ec2-user@ip-10-0-5-43 20181105]$ sbatch --  
dependency=afterok:11:12:13:14:15:16 /data/src/PyHipp/  
consol_jobs.sh
```

Lab 8

```
(env1) [ec2-user@ip-10-0-5-43 picasso]$ bash /data/src/PyHipp/  
consol_fsjobs.sh
```

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 8 Techniques

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
7	queue1	rpl1	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

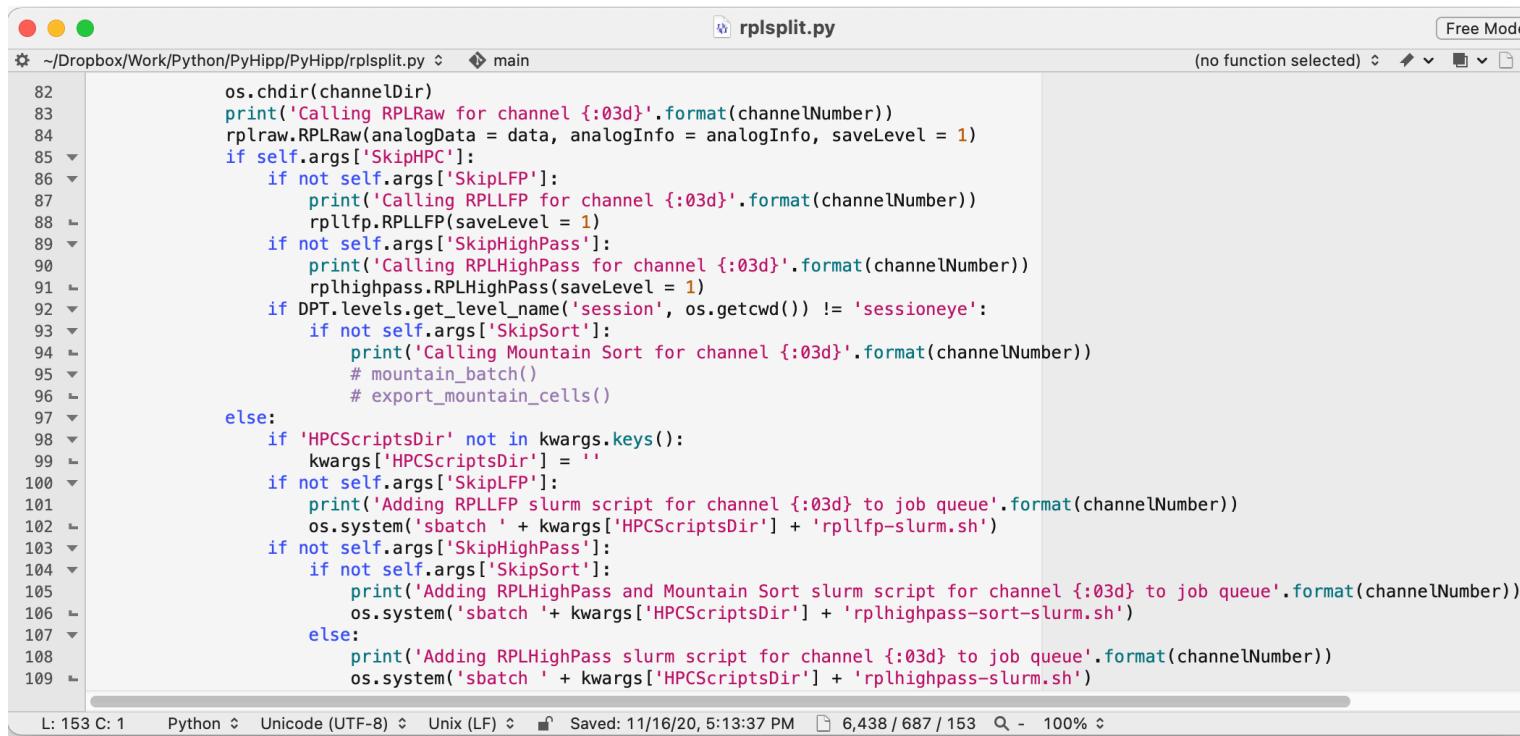
```
(env1) [ec2-user@ip-10-0-5-43 20181105]$ sbatch /data/src/PyHipp/consol_jobs.sh
```

Optimizing Parallel Processing

Fine-Grained Parallel Processing (226 jobs)

Just-in-time job submission

```
DPT.objects.processDirs(dirs=None, objtype=pyh.RPLSplit, channel=[*range(1,33)], SkipHPC=False, HPCScriptsDir = '/data/src/PyHipp/',  
SkipLFP=False, SkipHighPass=False, SkipSort=False);
```



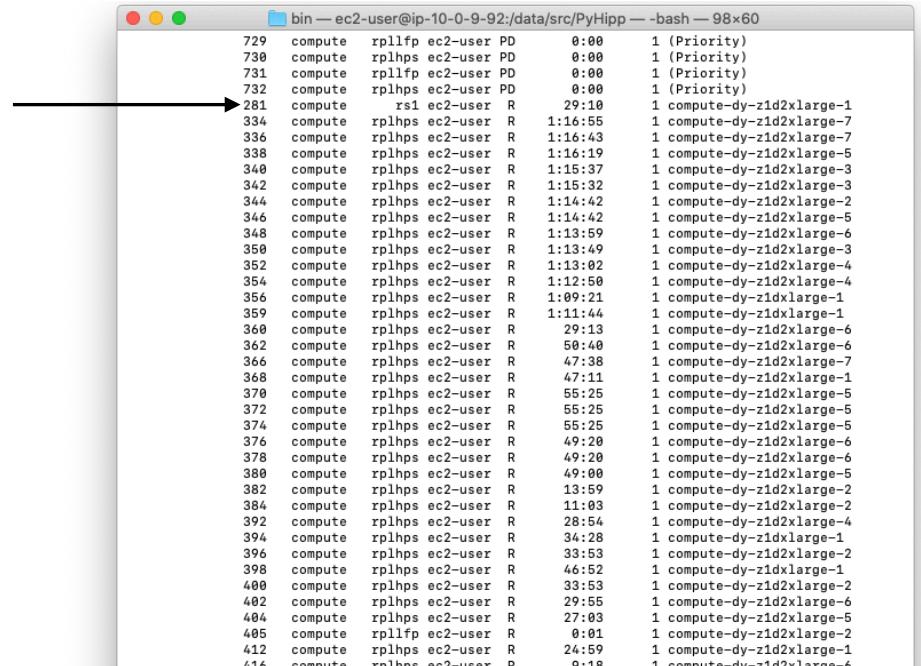
The screenshot shows a code editor window titled "rplsplit.py". The file path is indicated as "~/Dropbox/Work/Python/PyHipp/PyHipp/rplsplit.py". The code itself is a Python script with line numbers from 82 to 109. It performs various operations based on command-line arguments (self.args) and environment variables (os.environ). Key parts of the code include:

- Calling RPLRaw for each channel.
- Calling RPLLFP for each channel.
- Calling RPLHighPass for each channel.
- Handling session-specific logic (e.g., sessioneye).
- Performing Mountain Sort for each channel.
- Adding RPLLFP slurm scripts to a job queue.
- Adding RPLHighPass and Mountain Sort slurm scripts to a job queue.
- Adding RPLHighPass slurm scripts to a job queue.

The code uses standard Python libraries like os and print statements to log its progress.

Optimizing Parallel Processing

Fine-Grained Parallel Processing (226 jobs)



A screenshot of a terminal window titled "bin — ec2-user@ip-10-0-9-92:/data/src/PyHipp — bash — 98x60". The window displays a list of 226 parallel processing jobs. Each job entry consists of a job ID (e.g., 729, 730, 731, ..., 416), the command ("compute"), the user ("ec2-user"), the queue ("PD"), the start time ("0:00", "0:00", "0:00", ..., "0:18"), and the priority ("1 (Priority)", "1 (Priority)", "1 (Priority)", ..., "1 compute-dy-z1d2xlarge-1"). An arrow points from the left towards the terminal window.

Job ID	Command	User	Queue	Start Time	Priority
729	compute	ec2-user	PD	0:00	1 (Priority)
730	compute	rp1lfp	ec2-user	0:00	1 (Priority)
731	compute	rp1lfp	ec2-user	0:00	1 (Priority)
732	compute	rp1lfp	ec2-user	0:00	1 (Priority)
281	compute	rs1	ec2-user	R 29:10	1 compute-dy-z1d2xlarge-1
334	compute	rp1lfp	ec2-user	R 1:16:55	1 compute-dy-z1d2xlarge-7
336	compute	rp1lfp	ec2-user	R 1:16:43	1 compute-dy-z1d2xlarge-7
338	compute	rp1lfp	ec2-user	R 1:16:19	1 compute-dy-z1d2xlarge-5
340	compute	rp1lfp	ec2-user	R 1:15:37	1 compute-dy-z1d2xlarge-3
342	compute	rp1lfp	ec2-user	R 1:15:32	1 compute-dy-z1d2xlarge-3
344	compute	rp1lfp	ec2-user	R 1:14:42	1 compute-dy-z1d2xlarge-2
346	compute	rp1lfp	ec2-user	R 1:14:42	1 compute-dy-z1d2xlarge-5
348	compute	rp1lfp	ec2-user	R 1:13:59	1 compute-dy-z1d2xlarge-6
350	compute	rp1lfp	ec2-user	R 1:13:49	1 compute-dy-z1d2xlarge-3
352	compute	rp1lfp	ec2-user	R 1:13:02	1 compute-dy-z1d2xlarge-4
354	compute	rp1lfp	ec2-user	R 1:12:50	1 compute-dy-z1d2xlarge-4
356	compute	rp1lfp	ec2-user	R 1:09:21	1 compute-dy-z1dxlarge-1
359	compute	rp1lfp	ec2-user	R 1:11:44	1 compute-dy-z1d2xlarge-1
360	compute	rp1lfp	ec2-user	R 29:13	1 compute-dy-z1d2xlarge-6
362	compute	rp1lfp	ec2-user	R 50:40	1 compute-dy-z1d2xlarge-6
366	compute	rp1lfp	ec2-user	R 47:38	1 compute-dy-z1d2xlarge-7
368	compute	rp1lfp	ec2-user	R 47:11	1 compute-dy-z1d2xlarge-1
370	compute	rp1lfp	ec2-user	R 55:25	1 compute-dy-z1d2xlarge-5
372	compute	rp1lfp	ec2-user	R 55:25	1 compute-dy-z1d2xlarge-5
374	compute	rp1lfp	ec2-user	R 55:25	1 compute-dy-z1d2xlarge-5
376	compute	rp1lfp	ec2-user	R 49:20	1 compute-dy-z1d2xlarge-6
378	compute	rp1lfp	ec2-user	R 49:20	1 compute-dy-z1d2xlarge-6
380	compute	rp1lfp	ec2-user	R 49:00	1 compute-dy-z1d2xlarge-5
382	compute	rp1lfp	ec2-user	R 13:59	1 compute-dy-z1d2xlarge-2
384	compute	rp1lfp	ec2-user	R 11:03	1 compute-dy-z1d2xlarge-2
392	compute	rp1lfp	ec2-user	R 28:54	1 compute-dy-z1d2xlarge-4
394	compute	rp1lfp	ec2-user	R 34:28	1 compute-dy-z1dxlarge-1
396	compute	rp1lfp	ec2-user	R 33:53	1 compute-dy-z1d2xlarge-2
398	compute	rp1lfp	ec2-user	R 46:52	1 compute-dy-z1dxlarge-1
400	compute	rp1lfp	ec2-user	R 33:53	1 compute-dy-z1d2xlarge-2
402	compute	rp1lfp	ec2-user	R 29:55	1 compute-dy-z1d2xlarge-6
404	compute	rp1lfp	ec2-user	R 27:03	1 compute-dy-z1d2xlarge-5
405	compute	rp1lfp	ec2-user	R 0:01	1 compute-dy-z1d2xlarge-2
412	compute	rp1lfp	ec2-user	R 24:59	1 compute-dy-z1d2xlarge-1
416	compute	rnlhns	ec2-user	R 0:18	1 compute-dy-z1d2xlarge-1

```
(env1) [ec2-user@ip-10-0-5-43 20181105]$ sbatch --dependency=afterok:280:281:282:283:284 /data/src/PyHipp/consol_jobs.sh
```

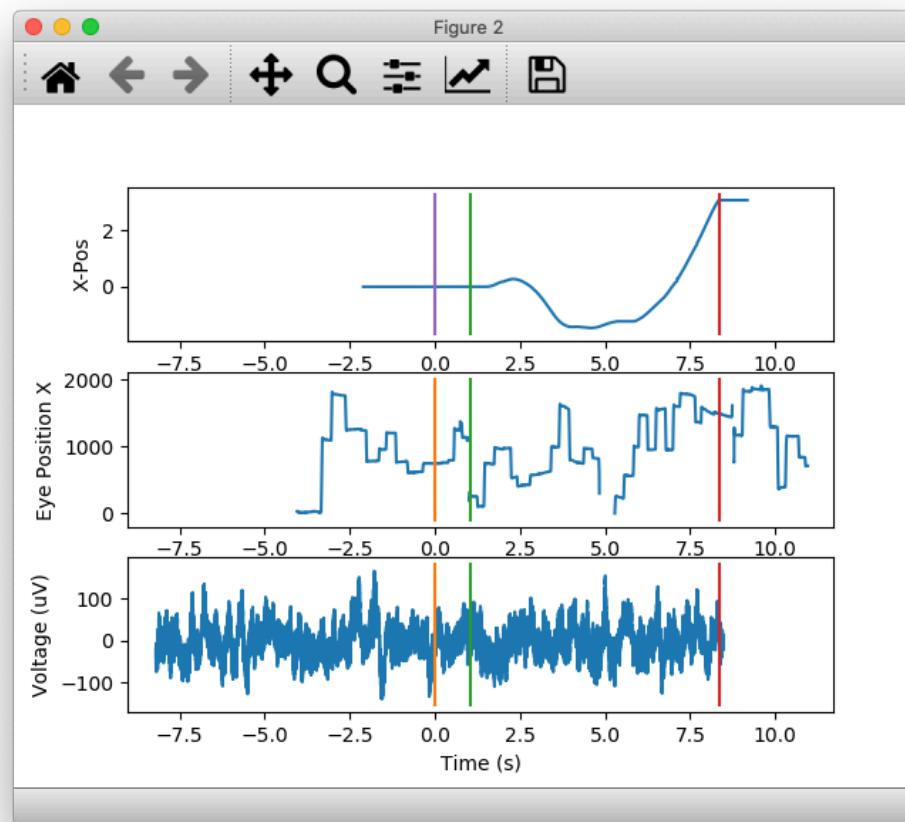
Data Engineering Principles

Review

- Aligning timestamps from different data
- Optimizing data processing pipelines by identifying dependencies
- Automatic head node deletion
- Using DPT to process hierarchically organized data
- Using PanGUI to view data and sets of data easily

Data Acquisition & Processing

Data Timelines



Data Engineering Principles

Review

- Aligning timestamps from different data
- Optimizing data processing pipelines by identifying dependencies
- Automatic head node deletion
- Using DPT to process hierarchically organized data
- Using PanGUI to view data and sets of data easily

Parallel Data Processing

Slurm Script

From Lab 5

```
In[ ]: DPT.objects.processDirs(dirs=None, objtype=pyh.RPLParallel, saveLevel=1)
In[ ]: DPT.objects.processDirs(dirs=None, objtype=pyh.RPLSplit, channel=[9, 31,
34, 56, 72, 93, 119, 120], saveLevel=1)
In[ ]: DPT.objects.processDirs(dirs=None, objtype=pyh.RPLLFP, saveLevel=1)
In[ ]: DPT.objects.processDirs(dirs=None, objtype=pyh.RPLHighPass, saveLevel=1)
In[ ]: DPT.objects.processDirs(dirs=None, objtype=pyh.Unity, saveLevel=1)
In[ ]: pyh.EDFSplit(saveLevel=1)
In[ ]: cd session01
In[ ]: pyh.aligning_objects()
```

```
#!/bin/bash

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

#SBATCH --time=1:00:00 # 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 -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);"
```

Called from /data/picasso/20181105

Parallel Data Processing

Parallel Processing

	RPL Parallel	RPL Split	RPLLFP	RPL HighPass	Unity	EDF Split
RPL Parallel						
RPL Split						
RPLLFP	✓					
RPL HighPass		✓				
Unity	✓					
EDF Split	✓					
<td>✓</td> <td></td> <td></td> <td>✓</td> <td>✓</td> <td></td>	✓			✓	✓	
raycast				✓	✓	
mountain_batch			✓			

RPLParallel	RPLSplit
Unity	RPLLFP
EDFSplit	RPLHighPass
aligning_objects	mountain_batch
raycast	

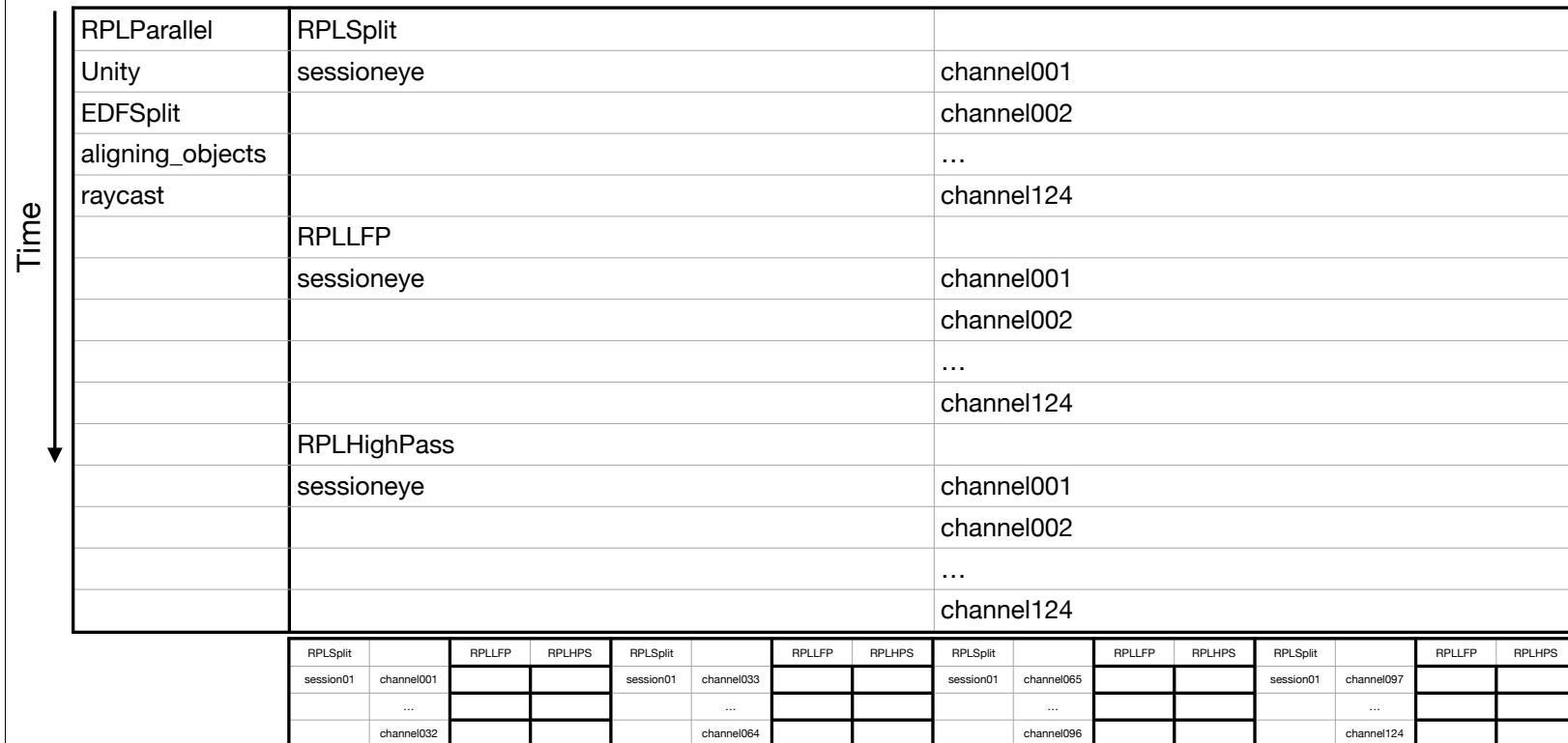
Optimizing Parallel Processing

Coarse-Grain Parallel Processing (5 jobs)

RPLParallel	RPLSplit		RPLSplit		RPLSplit		RPLSplit	
Unity	session01	channel001	session01	channel033	session01	channel065	session01	channel097
EDFSplit	sessioneye	channel002	sessioneye	channel034	sessioneye	channel066	sessioneye	channel098
<td>...</td> <td></td> <td>...</td> <td></td> <td>...</td> <td></td> <td>...</td> <td></td>	
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

Optimizing Parallel Processing

Fine-Grain Parallel Processing



Jobs: $6 + 110 + 110 = 226$

Optimizing Parallel Processing

Fine-Grain Parallel Processing (226 jobs)

```
bin — ec2-user@ip-10-0-9-92:/data/src/PyHipp — bash — 98x60
729 compute rpllfpc ec2-user PD 0:00 1 (Priority)
730 compute rplhps ec2-user PD 0:00 1 (Priority)
731 compute rpllfpc ec2-user PD 0:00 1 (Priority)
732 compute rplhps ec2-user PD 0:00 1 (Priority)
281 compute rs1 ec2-user R 29:18 1 compute-dy-zid2xlarge-1
334 compute rplhps ec2-user R 1:16:55 1 compute-dy-zid2xlarge-7
336 compute rplhps ec2-user R 1:16:43 1 compute-dy-zid2xlarge-7
338 compute rplhps ec2-user R 1:16:19 1 compute-dy-zid2xlarge-5
340 compute rplhps ec2-user R 1:15:37 1 compute-dy-zid2xlarge-3
342 compute rplhps ec2-user R 1:15:32 1 compute-dy-zid2xlarge-3
344 compute rplhps ec2-user R 1:14:42 1 compute-dy-zid2xlarge-2
346 compute rplhps ec2-user R 1:14:42 1 compute-dy-zid2xlarge-5
348 compute rplhps ec2-user R 1:13:59 1 compute-dy-zid2xlarge-6
350 compute rplhps ec2-user R 1:13:49 1 compute-dy-zid2xlarge-3
352 compute rplhps ec2-user R 1:13:02 1 compute-dy-zid2xlarge-4
354 compute rplhps ec2-user R 1:12:50 1 compute-dy-zid2xlarge-4
356 compute rplhps ec2-user R 1:09:21 1 compute-dy-zidxlarge-1
359 compute rplhps ec2-user R 1:11:44 1 compute-dy-zidxlarge-1
360 compute rplhps ec2-user R 29:13 1 compute-dy-zid2xlarge-6
362 compute rplhps ec2-user R 50:40 1 compute-dy-zid2xlarge-6
366 compute rplhps ec2-user R 47:38 1 compute-dy-zid2xlarge-7
368 compute rplhps ec2-user R 47:11 1 compute-dy-zid2xlarge-1
370 compute rplhps ec2-user R 55:25 1 compute-dy-zid2xlarge-5
372 compute rplhps ec2-user R 55:25 1 compute-dy-zid2xlarge-5
374 compute rplhps ec2-user R 55:25 1 compute-dy-zid2xlarge-5
376 compute rplhps ec2-user R 49:20 1 compute-dy-zid2xlarge-6
378 compute rplhps ec2-user R 49:20 1 compute-dy-zid2xlarge-6
380 compute rplhps ec2-user R 49:00 1 compute-dy-zid2xlarge-5
382 compute rplhps ec2-user R 13:59 1 compute-dy-zid2xlarge-2
384 compute rplhps ec2-user R 11:03 1 compute-dy-zid2xlarge-2
392 compute rplhps ec2-user R 28:54 1 compute-dy-zid2xlarge-4
394 compute rplhps ec2-user R 34:28 1 compute-dy-zidxlarge-1
396 compute rplhps ec2-user R 33:53 1 compute-dy-zid2xlarge-2
398 compute rplhps ec2-user R 46:52 1 compute-dy-zidxlarge-1
400 compute rplhps ec2-user R 33:53 1 compute-dy-zid2xlarge-2
402 compute rplhps ec2-user R 29:55 1 compute-dy-zid2xlarge-6
404 compute rplhps ec2-user R 27:03 1 compute-dy-zid2xlarge-5
405 compute rpllfpc ec2-user R 0:01 1 compute-dy-zid2xlarge-2
412 compute rplhps ec2-user R 24:59 1 compute-dy-zid2xlarge-1
416 compute rplhps ec2-user R 9:18 1 compute-dy-zid2xlarge-6
418 compute rplhps ec2-user R 8:00 1 compute-dy-zid2xlarge-6
419 compute rpllfpc ec2-user R 7:56 1 compute-dy-zid2xlarge-7
420 compute rpllfpc ec2-user R 7:56 1 compute-dy-zid2xlarge-7
421 compute rpllfpc ec2-user R 7:56 1 compute-dy-zid2xlarge-7
422 compute rpllfpc ec2-user R 7:56 1 compute-dy-zid2xlarge-7
423 compute rplhps ec2-user R 21:36 1 compute-dy-zid2xlarge-2
424 compute rplhps ec2-user R 20:48 1 compute-dy-zid2xlarge-1
425 compute rpllfpc ec2-user R 7:56 1 compute-dy-zid2xlarge-7
426 compute rplhps ec2-user R 11:25 1 compute-dy-zid2xlarge-2
427 compute rpllfpc ec2-user R 7:13 1 compute-dy-zid2xlarge-5
429 compute rplhps ec2-user R 4:21 1 compute-dy-zid2xlarge-4
430 compute rplhps ec2-user R 4:21 1 compute-dy-zid2xlarge-4
431 compute rpllfpc ec2-user R 4:21 1 compute-dy-zid2xlarge-4
432 compute rplhps ec2-user R 4:21 1 compute-dy-zid2xlarge-4
433 compute rpllfpc ec2-user R 4:21 1 compute-dy-zid2xlarge-4
434 compute rplhps ec2-user R 3:02 1 compute-dy-zid2xlarge-3
435 compute rpllfpc ec2-user R 3:02 1 compute-dy-zid2xlarge-3
436 compute rpllfpc ec2-user R 3:02 1 compute-dy-zid2xlarge-3
437 compute rplhps ec2-user R 3:02 1 compute-dy-zid2xlarge-3
438 compute rplhps ec2-user R 3:02 1 compute-dy-zid2xlarge-3
```

Cannot actually run 64 jobs as the Master Node uses 2 CPUs and the EC2 instance also uses 2 CPUs

Data Engineering Principles

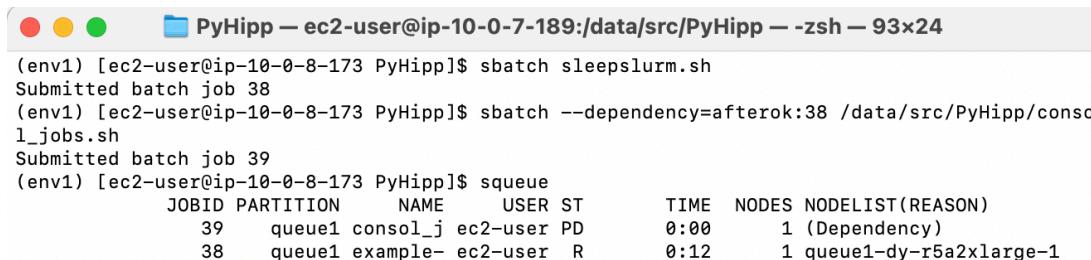
Review

- Aligning timestamps from different data
- Optimizing data processing pipelines by identifying dependencies
- Automatic head node deletion
- Using DPT to process hierarchically organized data
- Using PanGUI to view data and sets of data easily

Data Processing on AWS

Cluster Deletion

- Use slurm job dependencies to initiate snapshot:



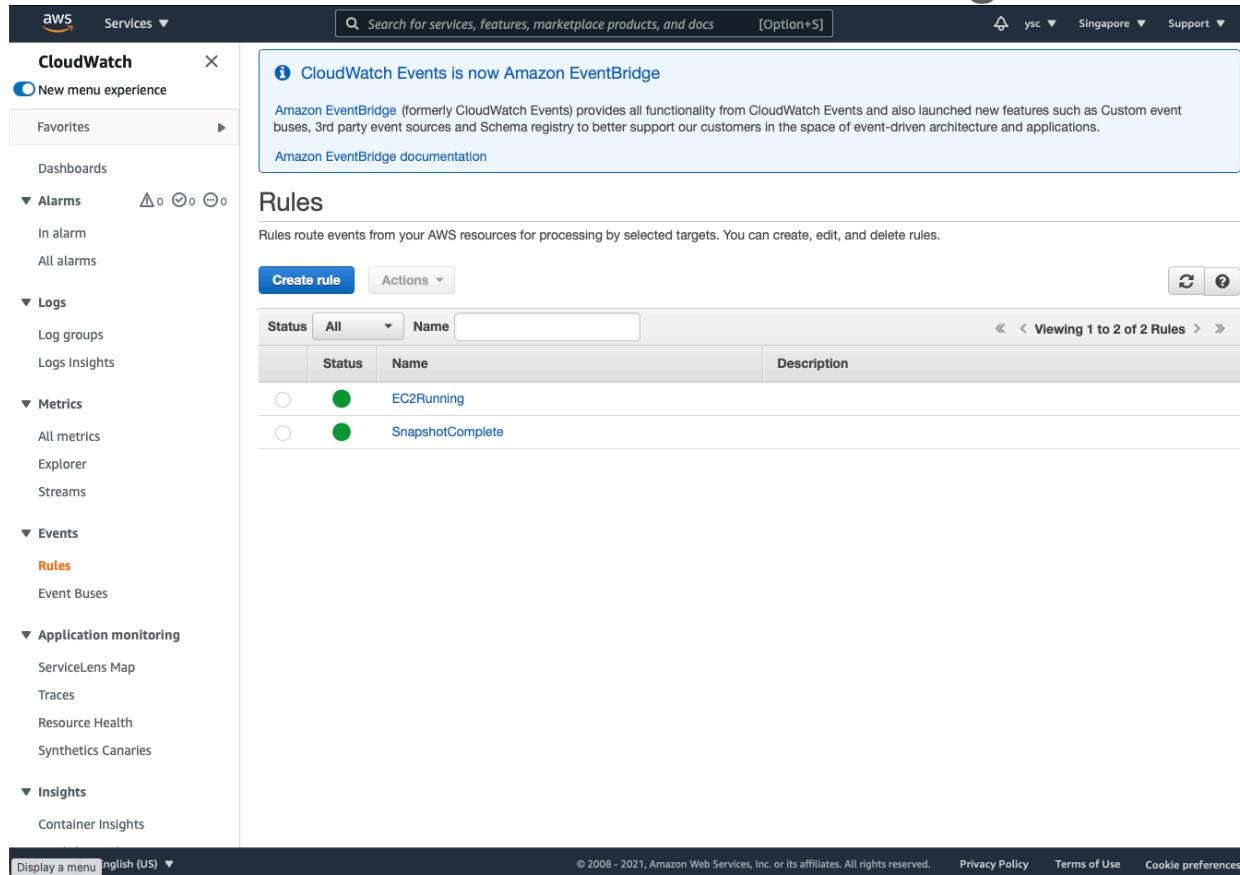
```
(env1) [ec2-user@ip-10-0-7-189 PyHipp]$ sbatch sleepslurm.sh
Submitted batch job 38
(env1) [ec2-user@ip-10-0-8-173 PyHipp]$ sbatch --dependency=afterok:38 /data/src/PyHipp/consol_jobs.sh
Submitted batch job 39
(env1) [ec2-user@ip-10-0-8-173 PyHipp]$ squeue
      JOBID PARTITION     NAME      USER 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

Data Processing on AWS

Cluster Deletion

Can we use CloudWatch/EventBridge instead?



The screenshot shows the AWS CloudWatch Events console. The left sidebar lists various CloudWatch services: Alarms, Logs, Metrics, Events, Application monitoring, and Insights. Under Events, the 'Rules' section is selected. A banner at the top states: "CloudWatch Events is now Amazon EventBridge". Below it, a note says: "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." A link to "Amazon EventBridge documentation" is provided.

Rules

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

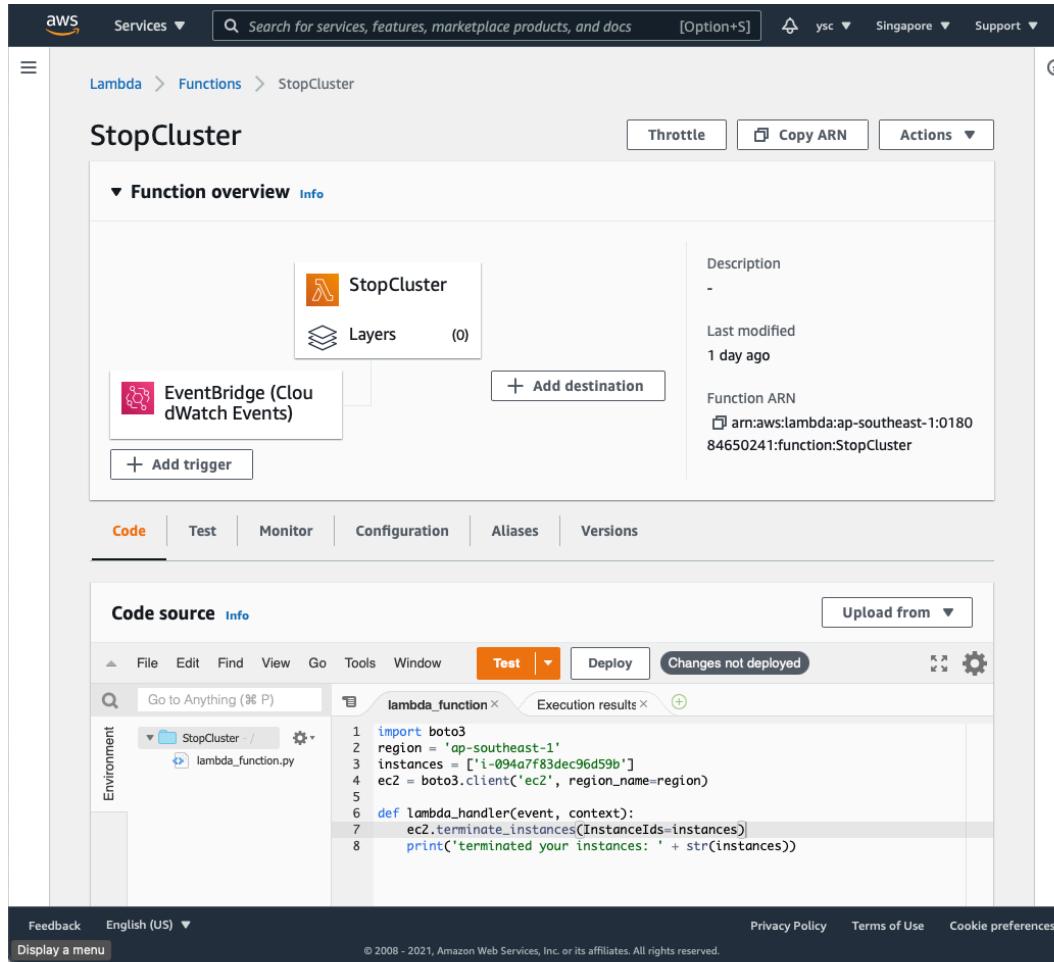
Status	All	Name	Description
<input type="radio"/>	<input checked="" type="radio"/>	EC2Running	
<input type="radio"/>	<input checked="" type="radio"/>	SnapshotComplete	

Viewing 1 to 2 of 2 Rules

Display a menu English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Data Processing on AWS

Cluster Deletion



Data Engineering Principles

Review

- Aligning timestamps from different data
- Optimizing data processing pipelines by identifying dependencies
- Automatic head node deletion
- Using DPT to process hierarchically organized data
- Using PanGUI to view data and sets of data easily

Parallel Data Processing

Data Processing Tools Levels

DataProcessingTools/DataProcessingTools/config.json

```
{"subjects": {"pattern": "([a-zA-Z]+)", "order": 0},  
  /data  
  "subject": {"pattern": "([a-zA-Z]+)", "order": 1},  
    picasso  
  "day": {"pattern": "([0-9]+)", "order": 2},  
    20181105  
  "session": {"pattern": "(session) ([a-zA-Z0-9]+)", "order": 3},  
    session01  
  "array": {"pattern": "(array) ([0-9]+)", "order": 4},  
    array01  
  "channel": {"pattern": "(channel) ([0-9]+)", "order": 5},  
    channel009  
  "cell": {"pattern": "(cell) ([0-9]+)", "order": 6}  
    cell01  
}
```

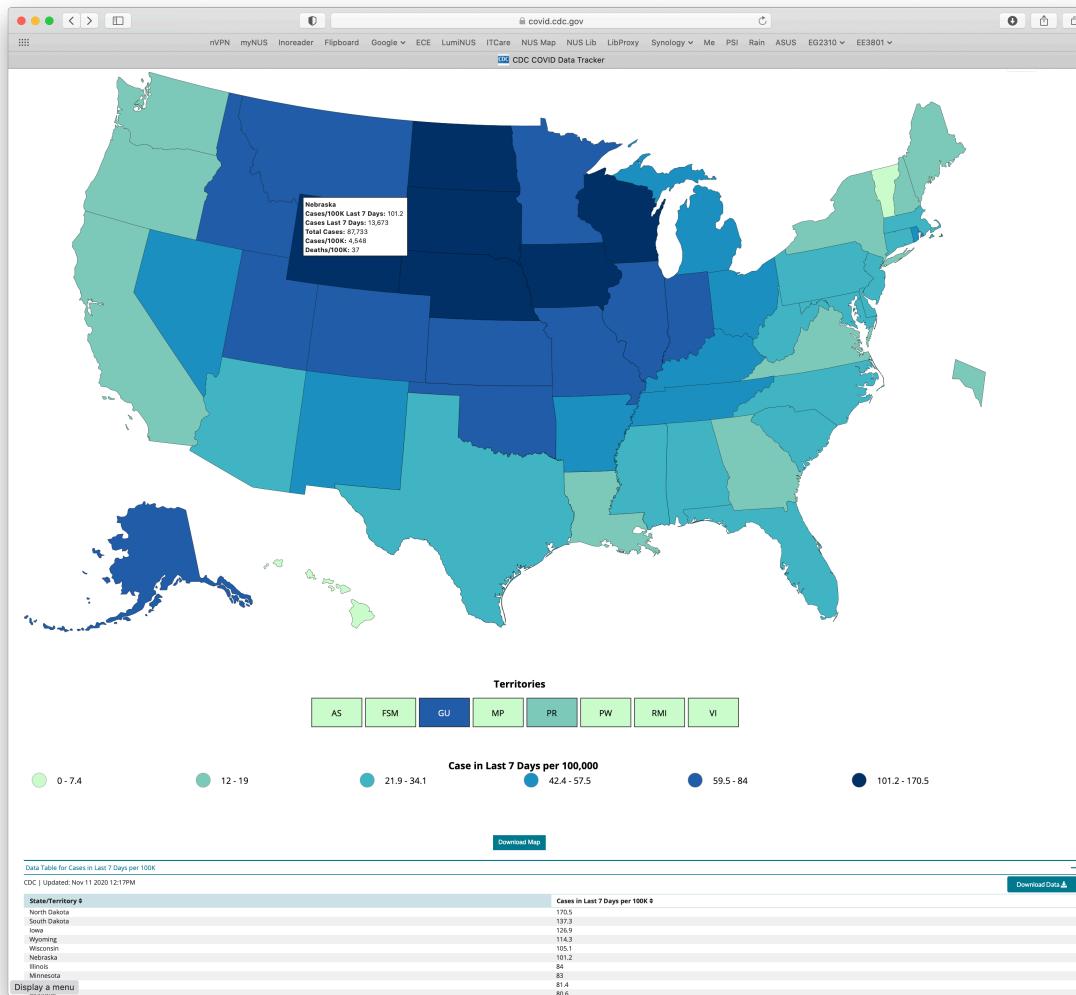
Data Acquisition & Processing

20181105			
	...		
	session01		
		...	
	array01		channel09
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl
		channel031	
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl
	array02		channel034
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl
		channel056	
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl
	array03		channel072
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl
		channel093	
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl
	array04		channel119
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl
		channel120	
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl

20181105			
	...		
	sessioneye		
		...	
		array01	channel09
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl
		channel031	
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl
		array02	
			channel034
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl
		channel056	
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl
		array03	
			channel072
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl
		channel093	
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl
		array04	
			channel119
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl
		channel120	
			rplraw_xxxx.hkl
			rplifp_xxxx.hkl

Data Acquisition & Processing

Hierarchically Organized Data



Data Acquisition & Processing

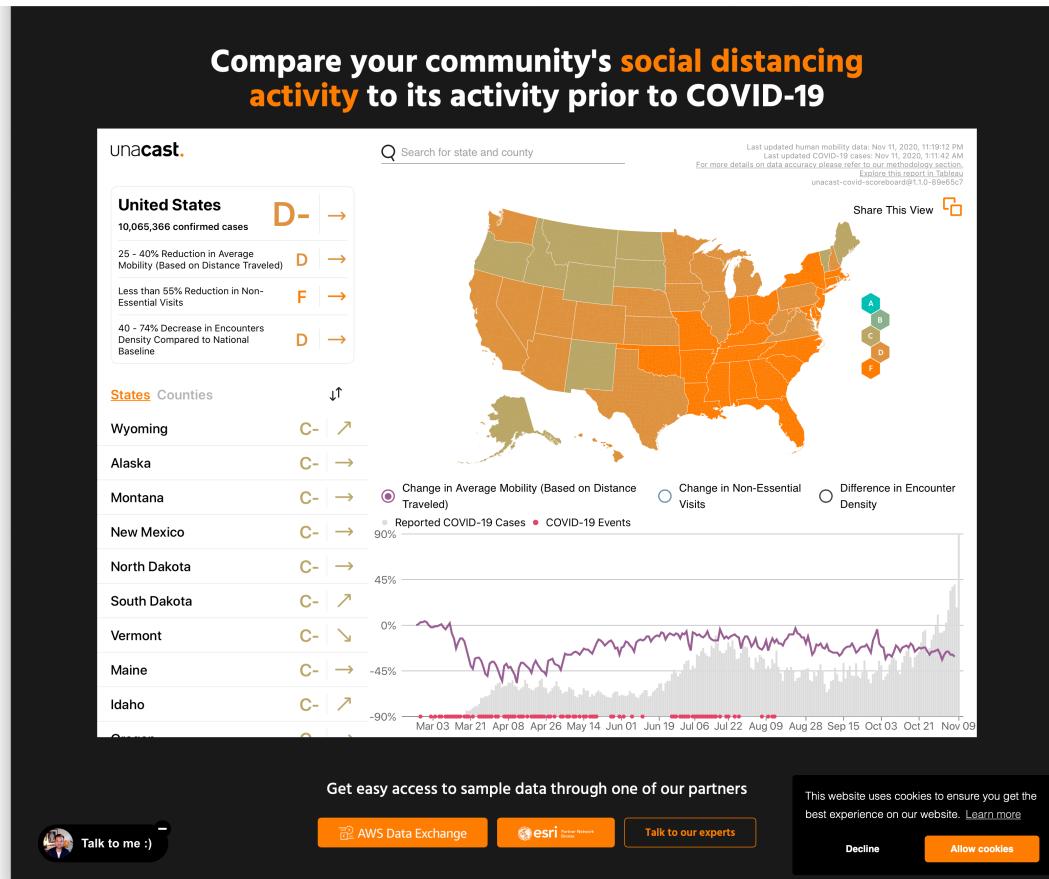
Hierarchically Organized Data

The screenshot shows a web browser window with the title "United States COVID-19 Cases and Deaths by County". The main content is a table titled "Cases By County" with the following columns: County, Total Cases, Percent of State's Cases, Cases per 100,000, and Total Deaths. The table lists 50 counties, starting with Los Angeles, CA at 325,876 cases and ending with Franklin, OH at 39,230 cases. A "View and Download COVID-19 Case Surveillance Public Use Data" button is located below the table. At the bottom of the page, there are sections for "HAVE QUESTIONS?", "CDC INFORMATION", "Privacy", and "CONNECT WITH CDC", along with links to various CDC resources and social media icons.

County	Total Cases	Percent of State's Cases	Cases per 100,000	Total Deaths
Los Angeles, CA	325,876	32.9%	3,246	7,200
Cook, IL	227,425	44.5%	4,416	5,667
Miami-Dade, FL	195,273	22.9%	7,187	3,690
Maricopa, AZ	169,091	64.3%	3,770	3,728
Harris, TX	167,956	17%	3,563	2,871
Dallas, TX	103,184	10.5%	3,915	1,331
Broward, FL	91,704	10.8%	4,696	1,560
Clark, NV	90,125	80.4%	3,976	1,576
Queens, NY	80,040	14.9%	3,551	7,310
Kings, NY	79,073	14.7%	3,089	7,440
Riverside, CA	72,341	7.3%	2,928	1,352
San Bernardino, CA	70,347	7.1%	3,227	1,096
Bexar, TX	68,376	6.9%	3,413	1,433
Tarrant, TX	67,192	6.8%	3,196	882
El Paso, TX	65,651	6.7%	7,823	717
Orange, CA	62,830	6.3%	1,978	1,512
San Diego, CA	61,053	6.2%	1,829	915
Salt Lake, UT	58,495	42.5%	5,041	333
Bronx, NY	57,425	10.7%	4,049	5,010
Palm Beach, FL	56,145	6.6%	3,751	1,612
Milwaukee, WI	54,391	18.5%	5,751	630
Nassau, NY	52,173	9.7%	3,845	2,224
Suffolk, NY	51,332	9.6%	3,476	2,022
Hillsborough, FL	51,296	6%	3,485	847
Orange, FL	49,422	5.8%	3,547	598
Wayne, MI	47,363	19.3%	2,707	3,103
Philadelphia, PA	44,974	18.8%	2,839	1,912
Hennepin, MN	44,547	23.5%	3,519	1,025
Westchester, NY	42,696	8%	4,413	1,480
Shelby, TN	40,733	14.1%	4,346	593
Franklin, OH	39,230	15%	2,979	643

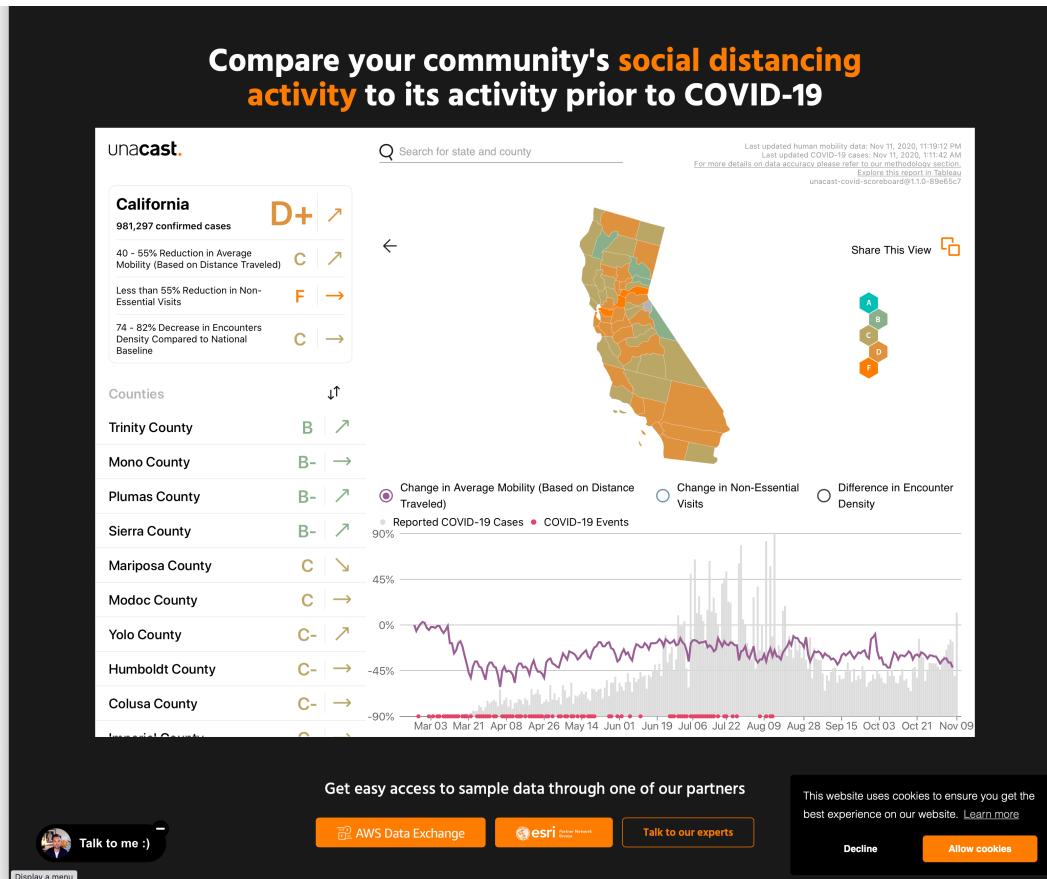
Data Acquisition & Processing

Hierarchically Organized Data



Data Acquisition & Processing

Hierarchically Organized Data



<https://www.unacast.com/covid19/social-distancing-scoreboard>

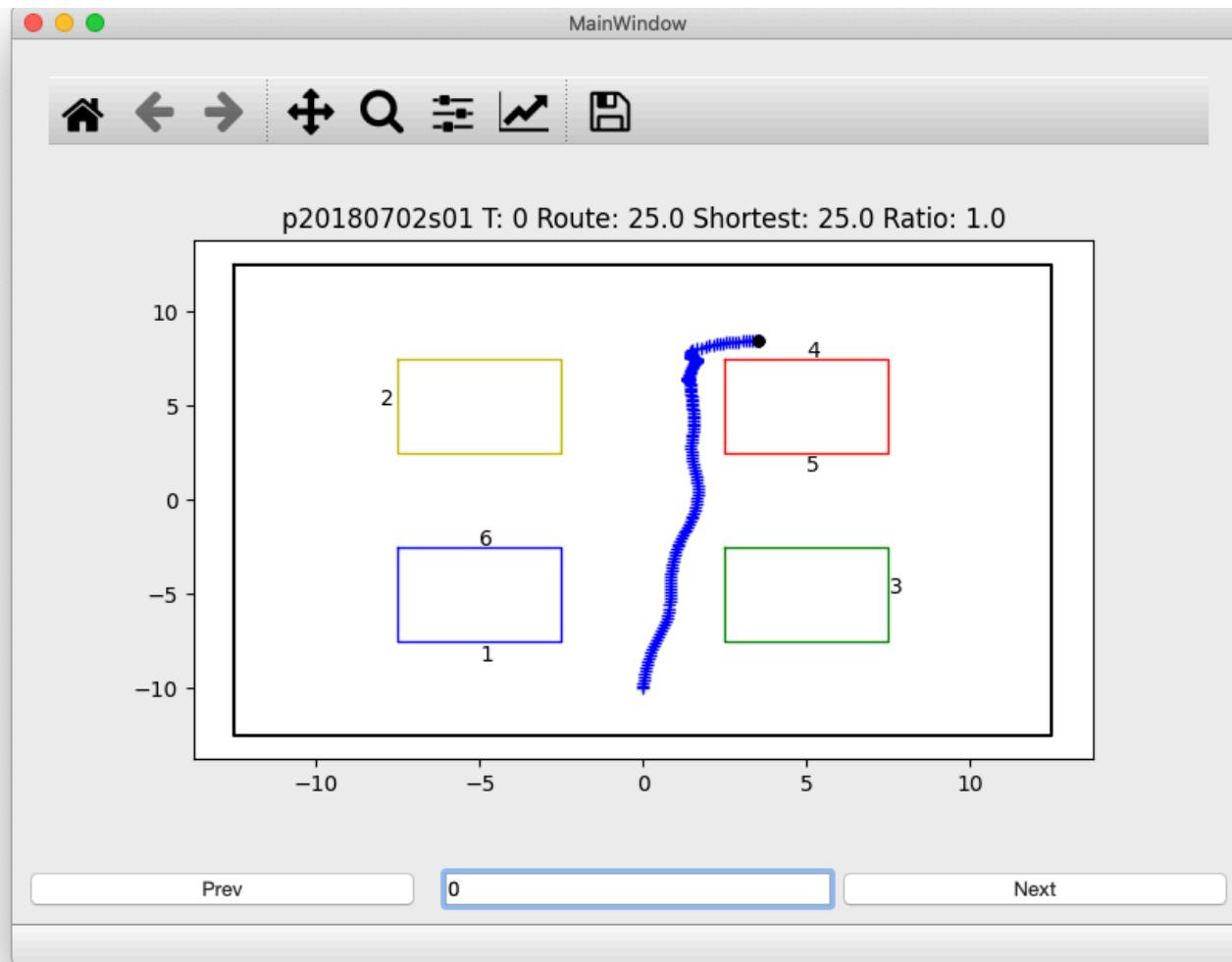
Data Engineering Principles

Review

- Aligning timestamps from different data
- Optimizing data processing pipelines by identifying dependencies
- Using DPT to process hierarchically organized data
- Using PanGUI to view data and sets of data easily

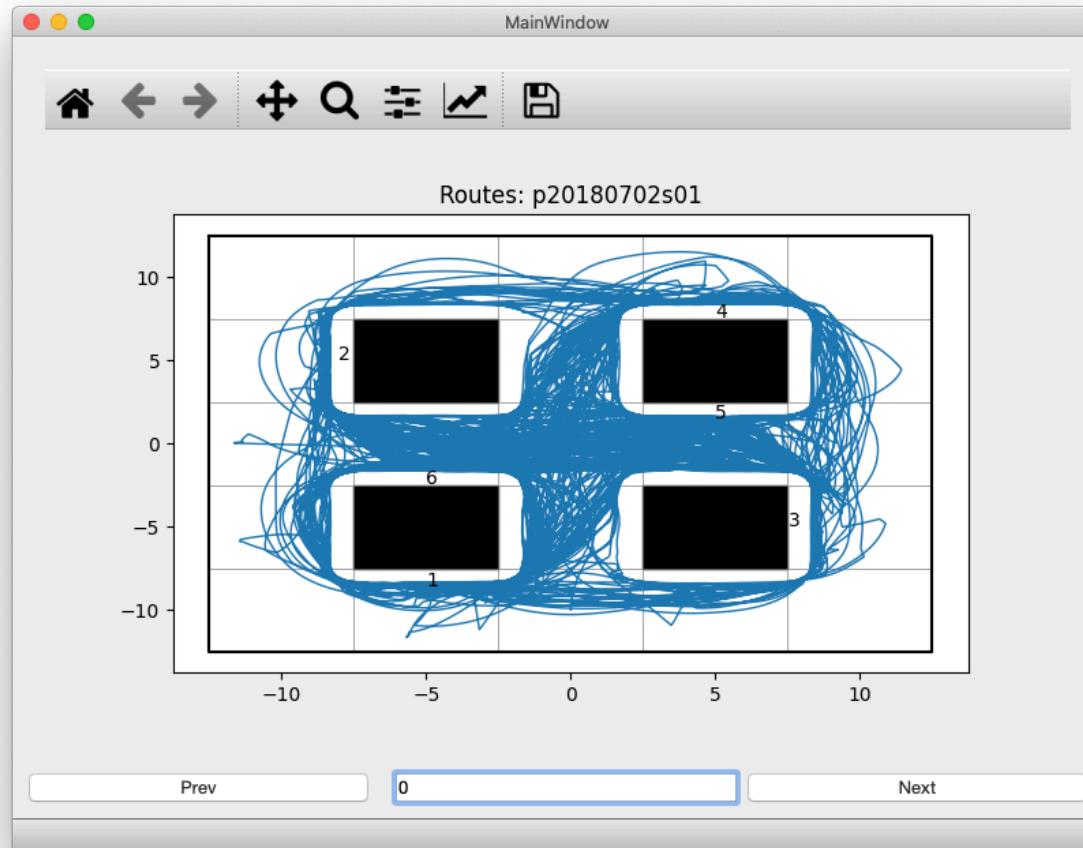
High-Throughput Visualization

PanGUI



High-Throughput Visualization

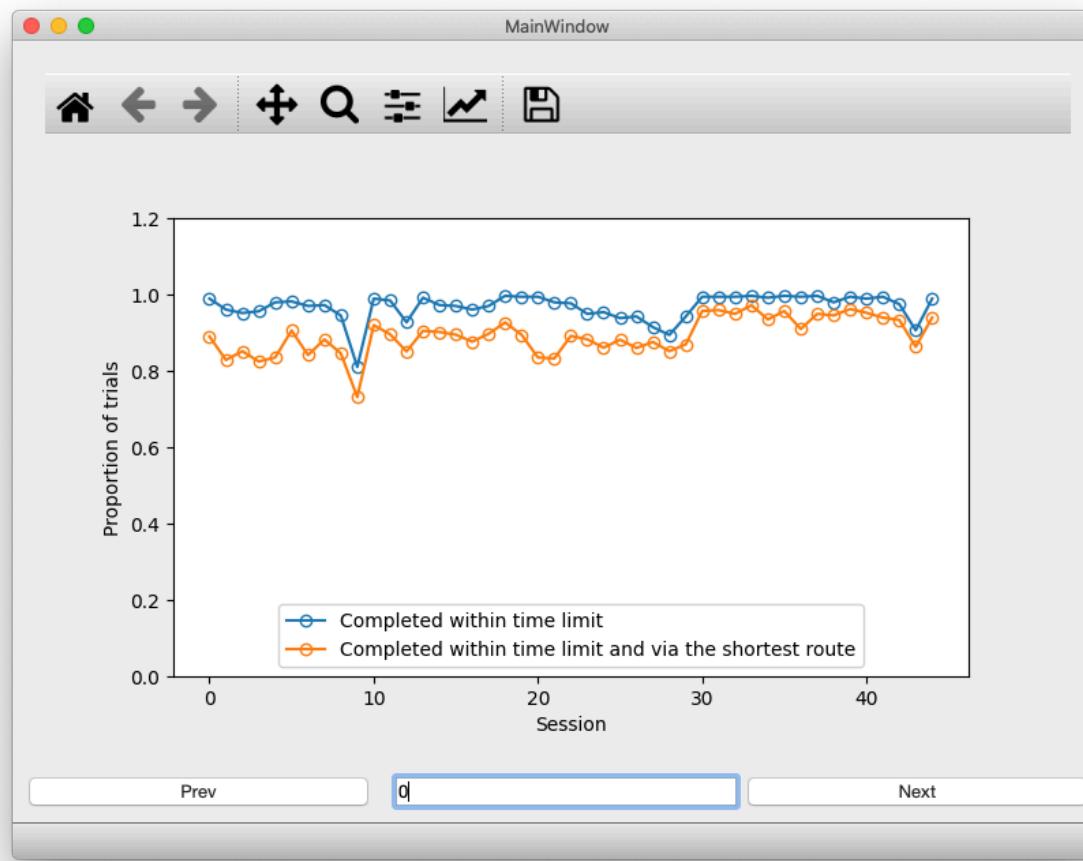
PanGUI



403 Trials

High-Throughput Visualization

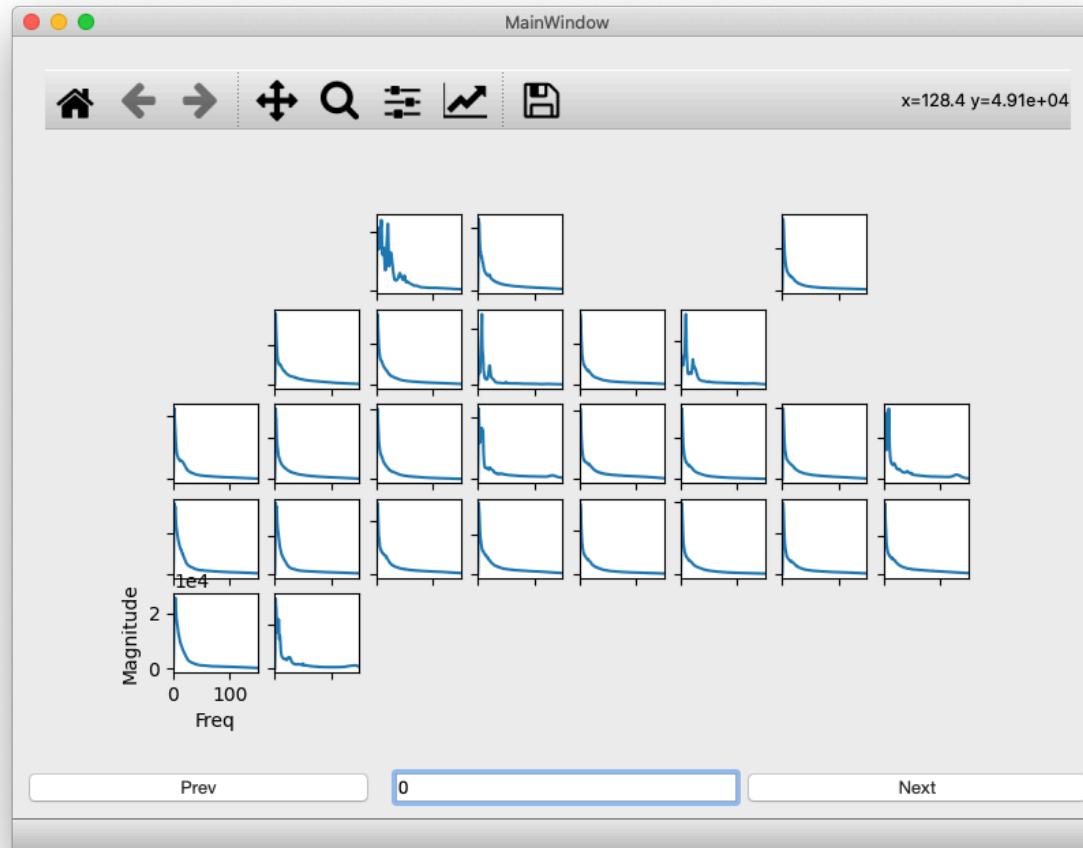
PanGUI



18,305 Trials

High-Throughput Visualization

PanGUI



32 channels

Quiz Preparations

Quiz on Nov 16 from 3 to 4 pm

- Before the quiz
 - Modify cluster-config.yaml on your EC2 instance to use only one compute node (keep the instance type the same as the one you used for the labs)

```
Scheduling:  
  Scheduler: slurm  
  SlurmQueues:  
    - Name: queue1  
      ComputeResources:  
        - Name: r5-2xlarge  
          InstanceType: r5.2xlarge  
          MinCount: 1  
          MaxCount: 1
```

- In order to reduce time spent copying files during the quiz, you will be given a Snapshot ID on Canvas Announcements 30 mins before the quiz to copy and subsequently use during cluster creation
- Create the cluster from your EC2 instance before the quiz (change instance type for the compute node if necessary)
- Do not perform any processing before the start of the quiz
- Link to quiz instructions will be sent via Canvas Announcements at 3 pm

Quiz Preparations

- Past year quizzes and solutions have been posted on Canvas
- Scripts from Lab 8 have been added to the upstream repository

Data Engineering Principles

Wrap Up

- Cleaning up
 - pcluster list
 - Delete clusters by doing: pcluster delete ClusterName
 - Delete EC2 instance using AWS Dashboard
 - Delete EBS snapshots using AWS Dashboard
 - Delete volumes
 - Delete NAT Gateways in VPC

Data Engineering Principles

Wrap Up

The screenshot shows the AWS VPC dashboard for the Asia Pacific region. The left sidebar includes sections for Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only Internet gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, Peering connections), Security (Network ACLs, Security groups), DNS firewall (Rule groups, Domain lists), and Network Firewall (Firewalls, Firewall policies). The main content area displays 'Resources by Region' with a grid of 16 cards. The 'NAT Gateways' card is highlighted with a red box. Other visible cards include VPCs (1), Subnets (3), Route Tables (1), Internet Gateways (1), Egress-only Internet Gateways (0), DHCP option sets (1), Endpoints (0), Instance Connect Endpoints (0), Launch EC2 Instances (Note: Your instances will launch in the Asia Pacific region), Service Health (View complete service health details), Settings (Zones, Console Experiments), Additional Information (VPC Documentation, All VPC Resources, Forums, Report an Issue), AWS Network Manager (Get started with Network Manager), and Site-to-Site VPN Connections (Create VPN Connection).

Questions?