EE3801 Data Engineering Principles

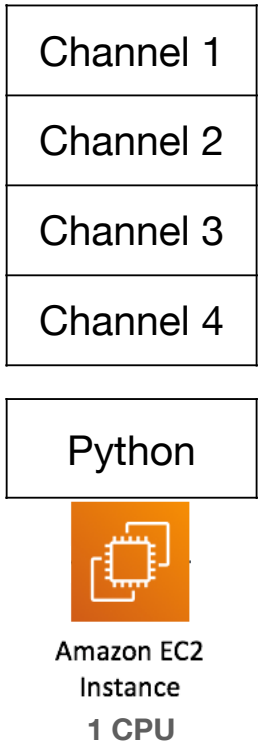
AWS ParallelCluster

AWS ParallelCluster

- Simplify creation and use of High Performance Computing (HPC) clusters on AWS
- Allows the use of up to 64 CPUs simultaneously
- High-performance computing (HPC)
- High-throughput computing (HTC)
- Many-task computing (MTC)

AWS ParallelCluster

Many-Task Computing



Serial Processing

AWS ParallelCluster

Many-Task Computing

Channel 1	Channel 2	Channel 3	Channel 4

Python 1	Python 2	Python 3	Python 4

Amazon EC2
Instance
1 CPU

Computer Multitasking

AWS ParallelCluster

Many-Task Computing

Channel 1	Channel 2	Channel 3	Channel 4

Python 1	Python 2	Python 3	Python 4

Amazon EC2
Instance
4 CPUs

Parallel Processing

AWS ParallelCluster

Many-Task Computing

Channel 1	Channel 2	Channel 3	Channel 4	…	Channel 100

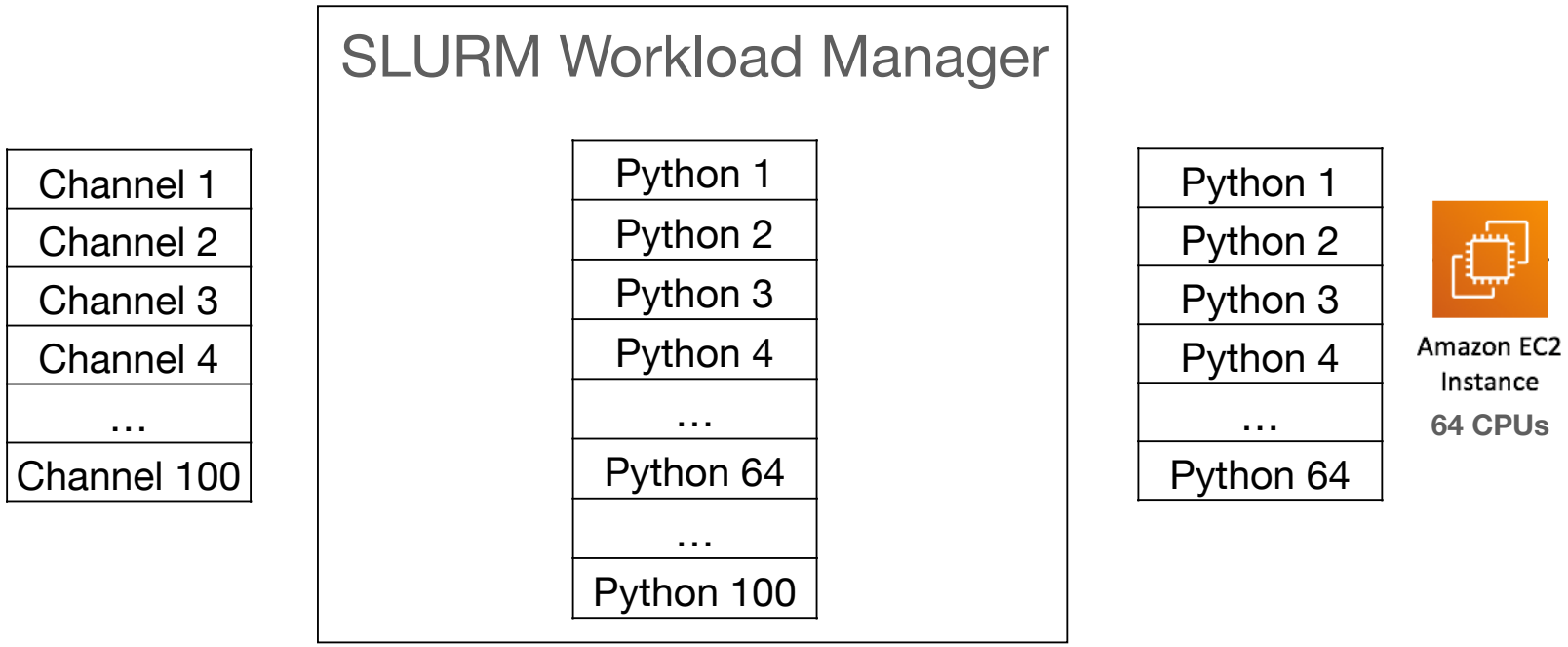
Python 1	Python 2	Python 3	Python 4	…	Python 100

Wastes time swapping items in and out of memory due to multi-tasking

Parallel Processing + Multitasking

AWS ParallelCluster

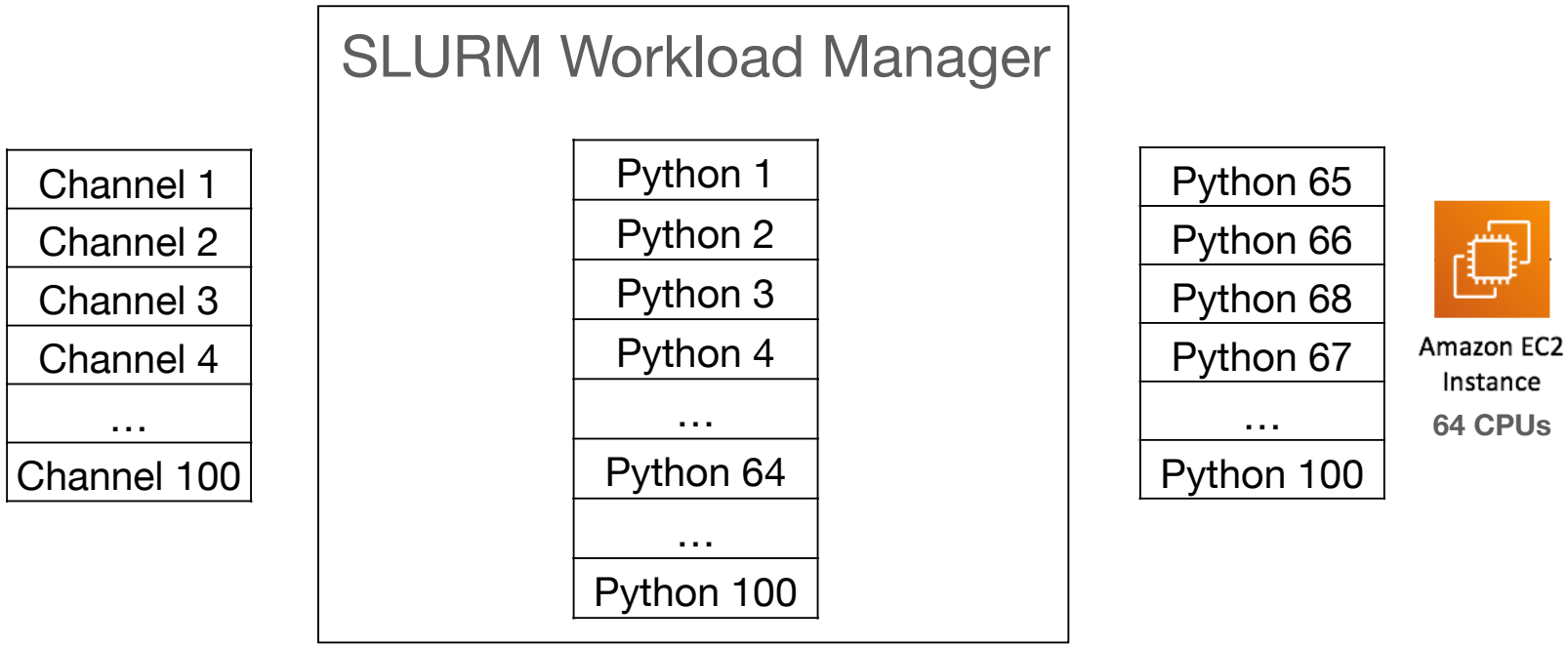
Many-Task Computing



Job Scheduling

AWS ParallelCluster

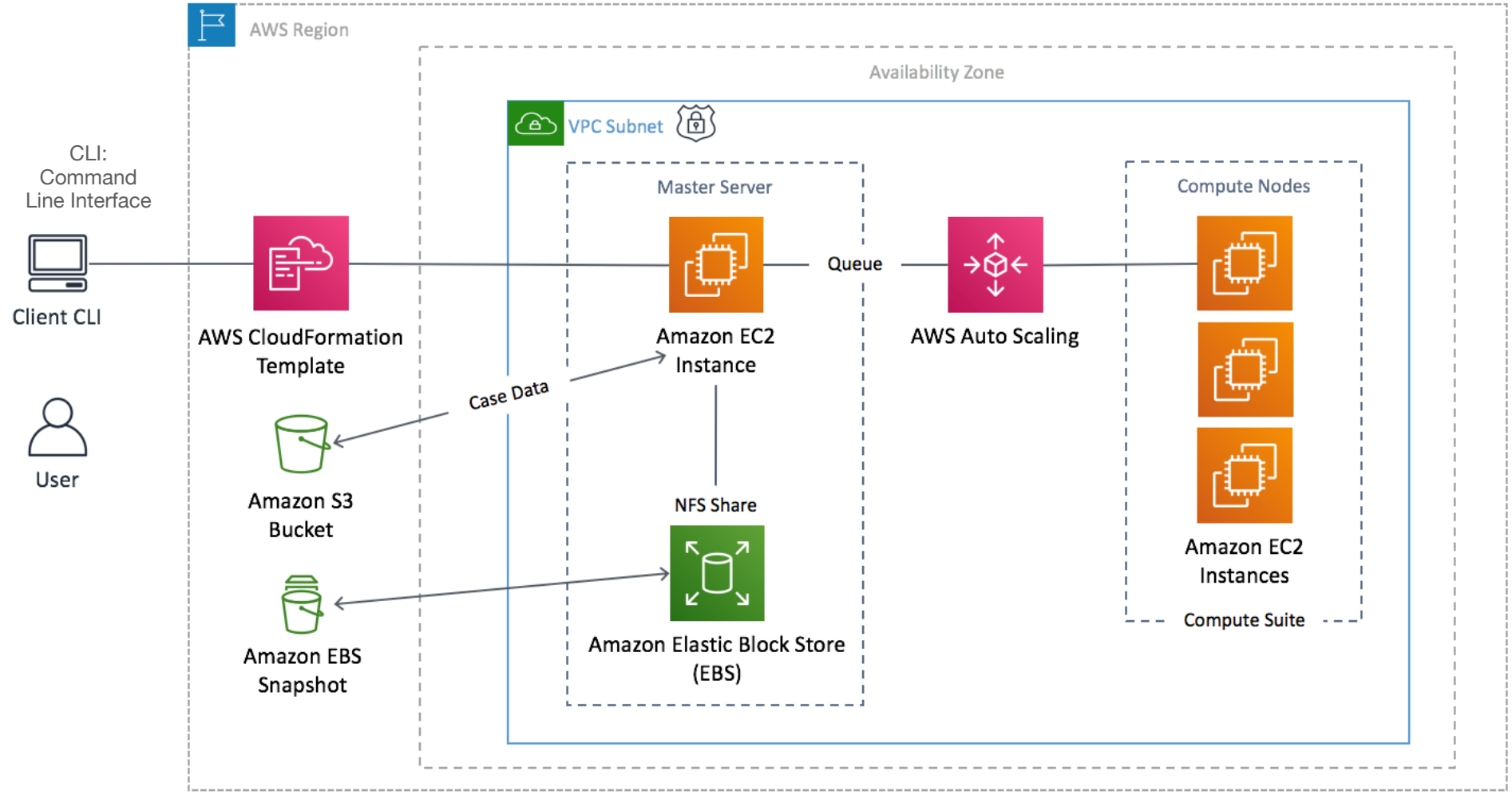
Many-Task Computing



Job Scheduling

AWS ParallelCluster

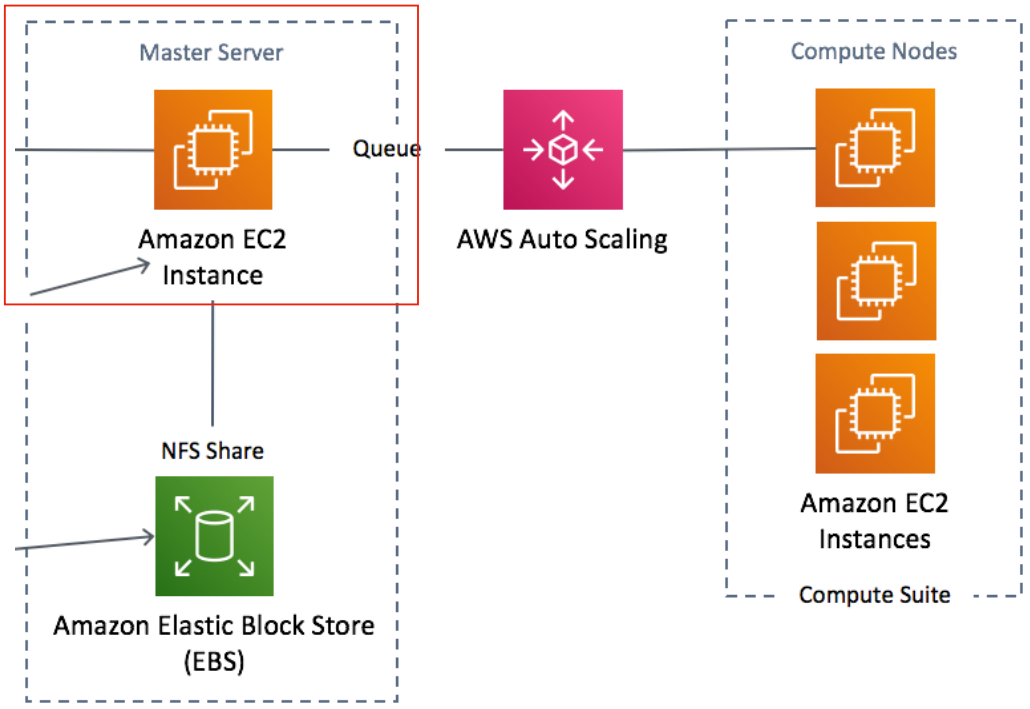
Architectural Diagram



AWS ParallelCluster

Master Server

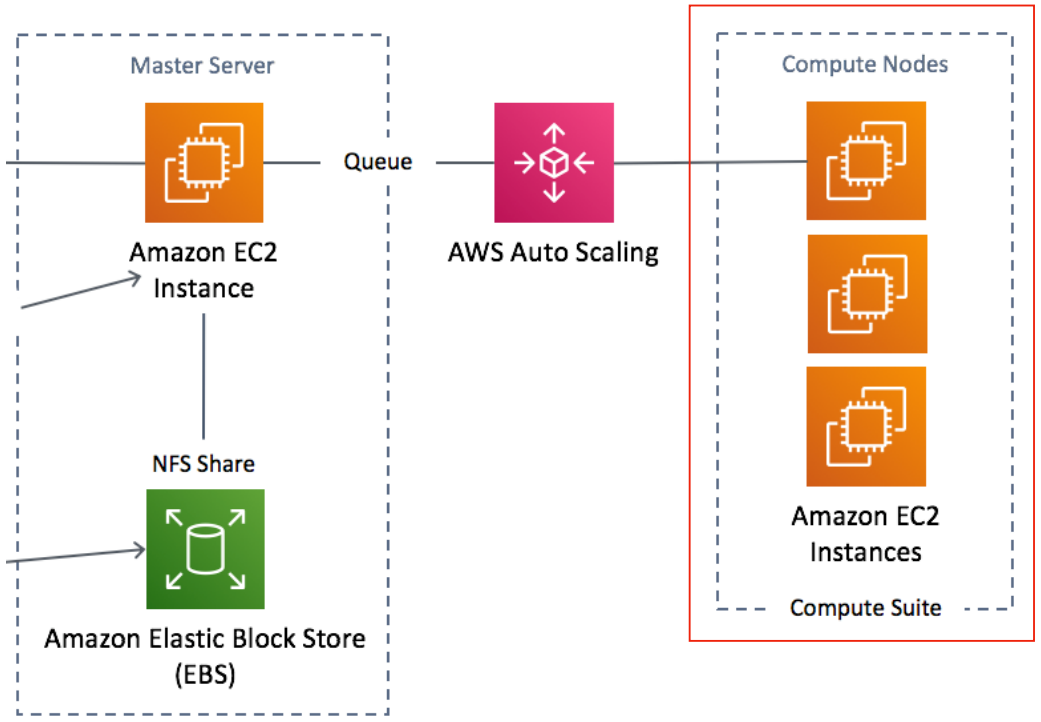
- Master or Head node
- Main login interface to the cluster
- Jobs are submitted and managed here
- Always running once the cluster is created
- Configured using the file ~/cluster-config.yaml



AWS ParallelCluster

Compute Nodes

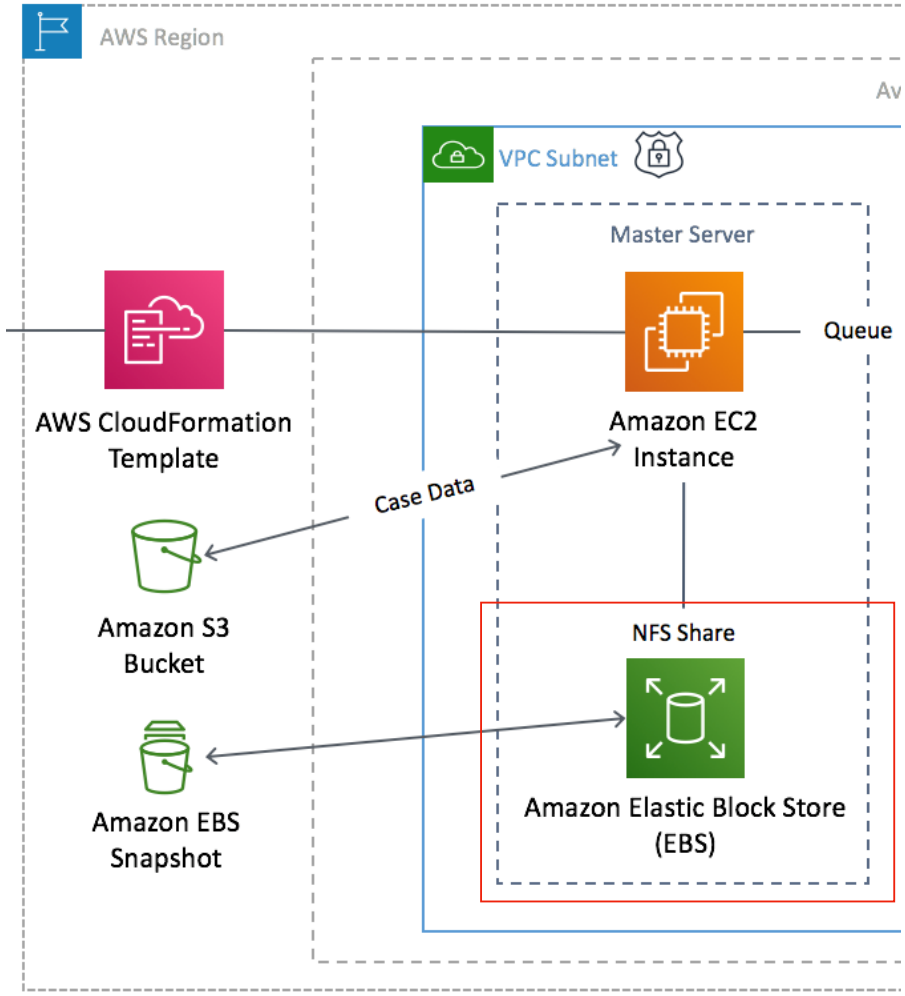
- Compute nodes
- Created as necessary when jobs are submitted
- Can be of different types (e.g. 4 CPUs or 8 CPUs)
- Copy their environments from the master node
- Released once jobs are completed



AWS ParallelCluster

EBS Storage

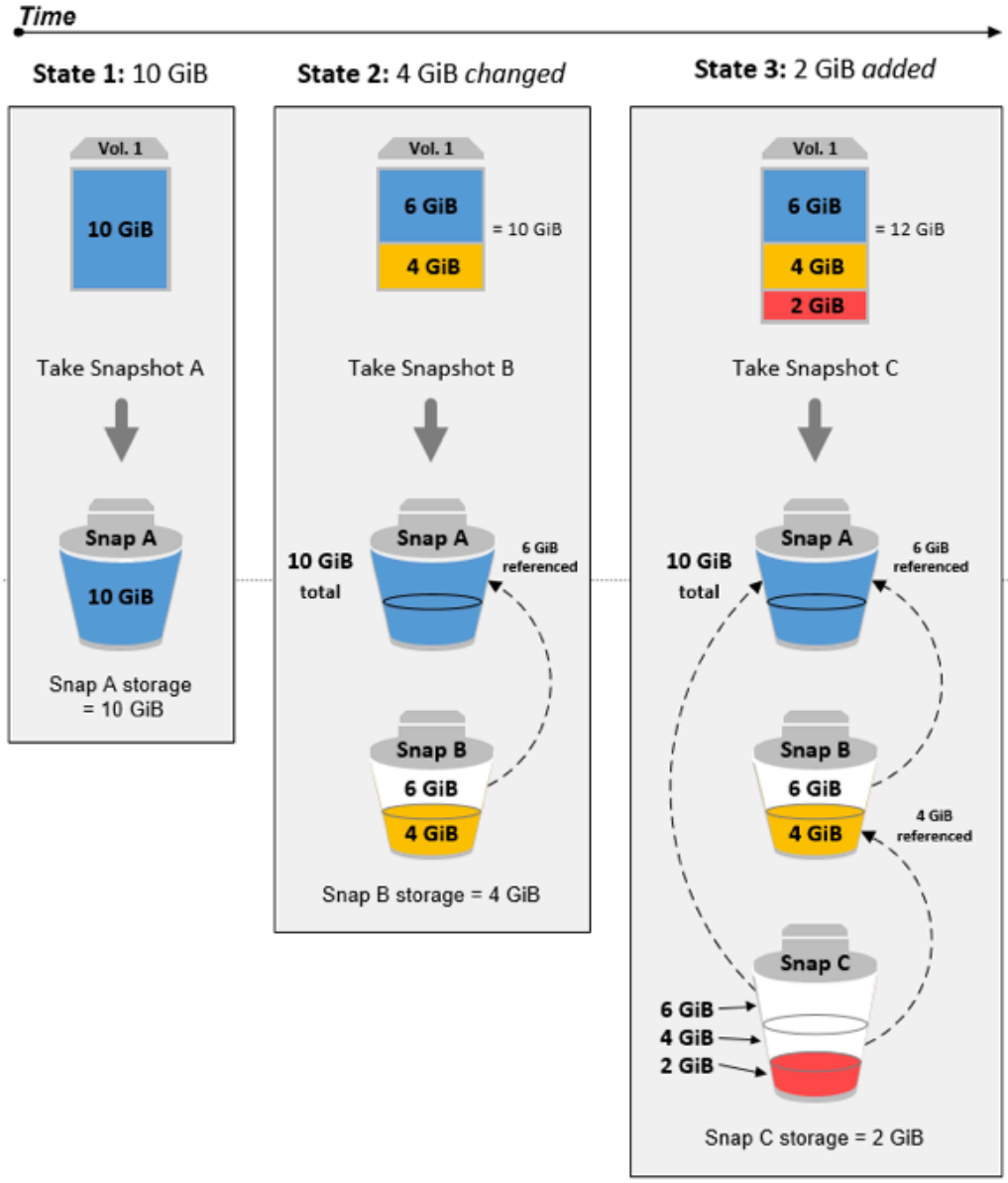
- File system for cluster
- Available from both master and compute nodes
- Needs to be saved to snapshot to preserve data
- Restored from snapshot when clusters are created
- Remember to save a snapshot before deleting a cluster



AWS ParallelCluster

EBS Snapshots

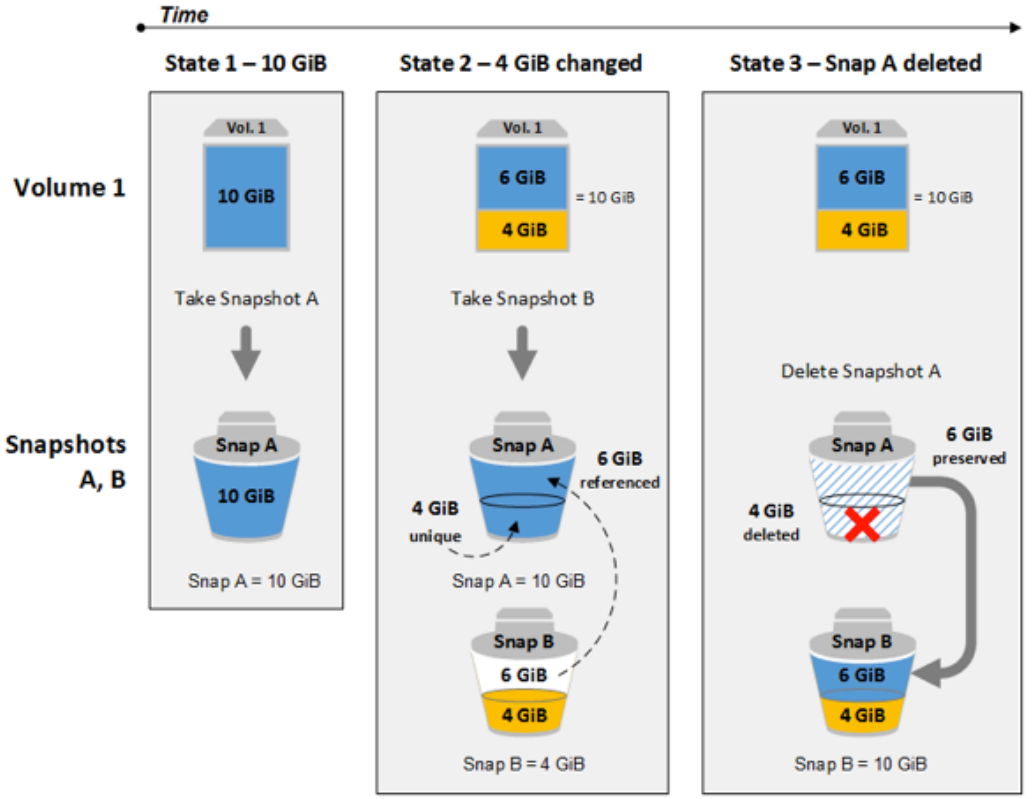
- Incremental backups
- Only blocks that have changed after your most recent snapshot are saved
- Unchanged blocks are referenced to previous snapshot
- New data will add to the storage required
- Total storage for 3 snapshots is 10 + 4 + 2 = 16 GiB



AWS ParallelCluster

EBS Snapshots

- Deleting snapshots
- Only data referenced exclusively by that snapshot is removed
- Data referenced by other snapshots will be retained
- 4 GiB deleted when deleting Snapshot A
- Total storage for Snapshot B is 10 GiB



AWS ParallelCluster

Config File

Master Node

Compute Node

EBS Storage

```
Region: ap-southeast-1
Image:
  Os: alinux2
HeadNode:
  InstanceType: t3a.nano
  Networking:
    SubnetId: subnet-xxxxxxxxxxxxxxxxx
  Ssh:
    KeyName: MyKeyPair
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: queue1
    ComputeResources:
    - Name: t3a-nano
      InstanceType: t3a.nano
      MinCount: 0
      MaxCount: 10
    Networking:
      SubnetIds:
      - subnet-xxxxxxxxxxxxxxxxx
SharedStorage:
  - MountDir: data
    Name: ee3801
    StorageType: Ebs
    EbsSettings:
      Size: 1000
      Encrypted: false
      SnapshotId: snap-xxxxxxxxxxxxxxxxx
```

AWS ParallelCluster

Instance Types

https://aws.amazon.com/ec2/instance-types/

https://ap-southeast-1.console.aws.amazon.com/ec2/v2/home?region=ap-southeast-1#InstanceTypes:

AWS ParallelCluster

Procedure

Computer	Edit config file
	`pcluster create-cluster --cluster-configuration ~/cluster-config.yaml --cluster-name MyCluster01`
	`pcluster ssh -i ~/MyKeyPair.pem --region ap-southeast-1 --cluster-name MyCluster01`
AWS	
Computer	

EE3801 Data Engineering Principles

Data Processing on AWS

Data Processing on AWS

Data Processing on AWS

- AWS setup
- Raw data organization
- Data pipeline
- Data objects and data processing tools

Data Processing on AWS

AWS ParallelCluster

- Saved snapshots are mounted in /data
- Contains:
  - lost+found - created by the OS
  - miniconda3 - conda environment (env1) created for you
  - picasso - data
  - RCP - compiled Unity VirtualMaze executable

AWS ParallelCluster

Config File

Master Node

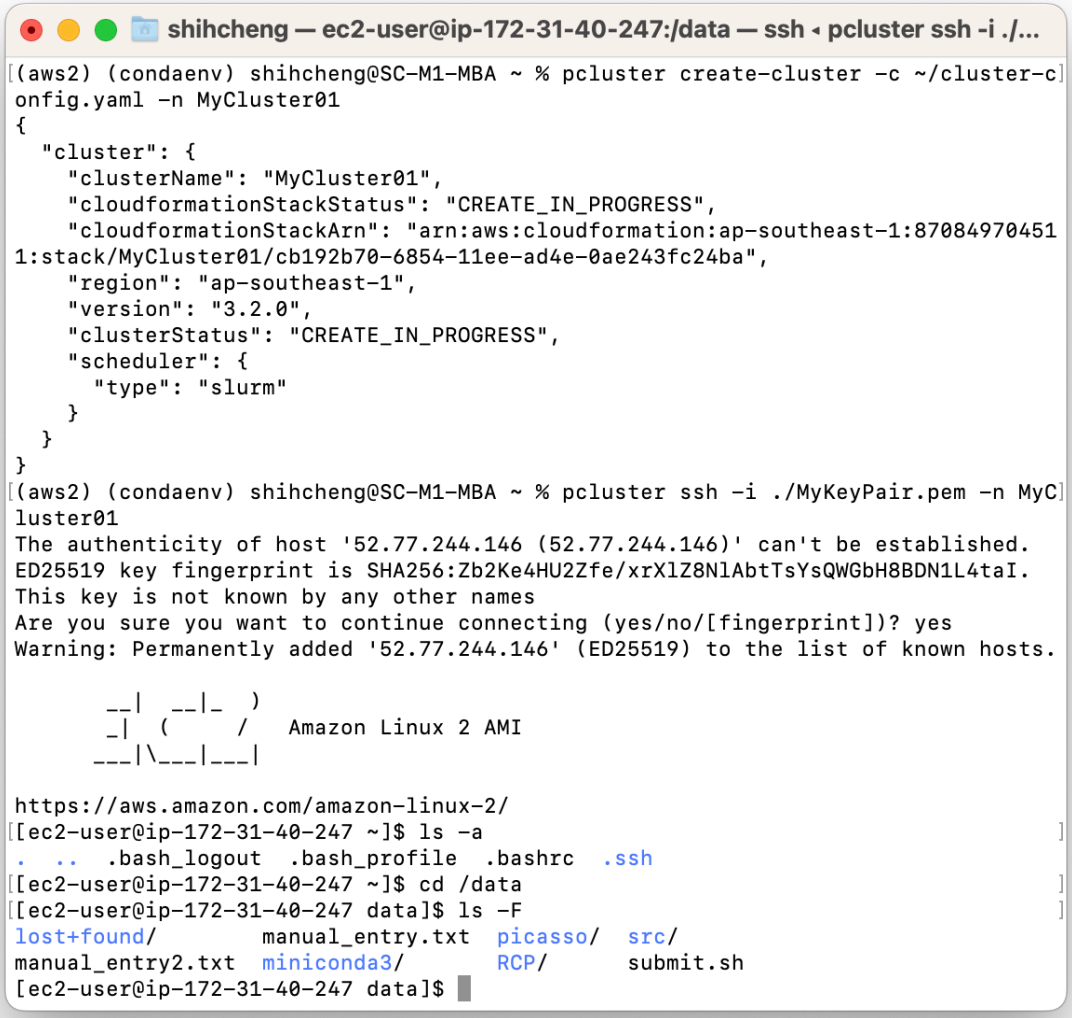
Compute Node

EBS Storage

```
Region: ap-southeast-1
Image:
  Os: alinux2
HeadNode:
  InstanceType: t3a.nano
  Networking:
    SubnetId: subnet-xxxxxxxxxxxxxxxxx
  Ssh:
    KeyName: MyKeyPair
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: queue1
    ComputeResources:
    - Name: t3a-nano
      InstanceType: t3a.nano
      MinCount: 0
      MaxCount: 10
    Networking:
      SubnetIds:
      - subnet-xxxxxxxxxxxxxxxxx
SharedStorage:
  - MountDir: data
    Name: ee3801
    StorageType: Ebs
    EbsSettings:
      Size: 1000
      Encrypted: false
      SnapshotId: snap-xxxxxxxxxxxxxxxxx
```

Data Processing on AWS

AWS ParallelCluster



Data Processing on AWS

AWS ParallelCluster One-Time Set Up

- After creating and logging into a cluster for the first time:
  - Save AWS credentials to ~/.aws
  - cp -r ~/.aws /data/aws
  - /data/miniconda3/bin/conda init ← Modifies ~/.bashrc
  - source ~/.bashrc
  - conda activate env1
  - Clone GitHub code to /data/src
- Save /data to snapshot

AWS ParallelCluster

Procedure after One-Time Set Up

Computer	Edit config file
Computer	`pcluster create-cluster --cluster-configuration ~/cluster-config.yaml --cluster-name MyCluster01`
Computer	`pcluster ssh -i ~/MyKeyPair.pem --region ap-southeast-1 --cluster-name MyCluster01`
AWS	`cp -r /data/aws ~/.aws`
AWS	`/data/miniconda3/bin/conda init` ← Modifies ~/.bashrc
AWS	`source ~/.bashrc`
AWS	`conda activate env1` ← Needs to be done on subsequent logins as well
AWS	`nano slurm-script.sh`
AWS	`sbatch slurm-script.sh`
AWS	`squeue`
AWS	`exit`
Computer	`update_snapshot.sh data 2`
Computer	`pcluster delete-cluster --region ap-southeast-1 --cluster-name MyCluster01`

AWS ParallelCluster

Slurm Script

```
#!/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 "example-job"   # job name

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

# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE
python -u -c "import PyHippocampus as pyh; pyh.RPLLFP(saveLevel = 1)"
```

AWS ParallelCluster

Procedure after One-Time Set Up

Computer	Edit config file
Computer	`pcluster create-cluster --cluster-configuration ~/cluster-config.yaml --cluster-name MyCluster01`
Computer	`pcluster ssh -i ~/MyKeyPair.pem --region ap-southeast-1 --cluster-name MyCluster01`
AWS	`cp -r /data/aws ~/.aws`
AWS	`/data/miniconda3/bin/conda init` ← Modifies ~/.bashrc
AWS	`source ~/.bashrc`
AWS	`conda activate env1` ← Needs to be done on subsequent logins as well
AWS	`nano slurm-script.sh`
AWS	`sbatch slurm-script.sh` ← Submits job
AWS	`squeue` ← Checks on jobs
AWS	`exit`
Computer	`update_snapshot.sh data 2`
Computer	`pcluster delete-cluster --region ap-southeast-1 --cluster-name MyCluster01`

Data Processing on AWS

Data Processing on AWS

- AWS setup
- Raw data organization
- Data pipeline
- Data objects and data processing tools

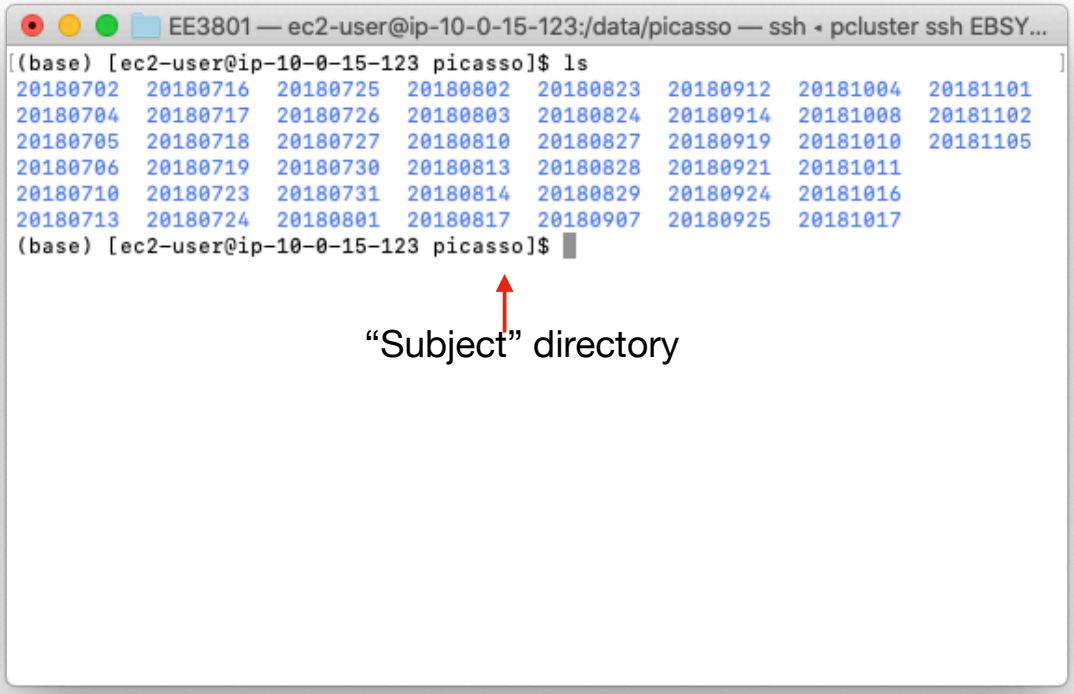
Data Processing on AWS

AWS ParallelCluster

- Saved snapshots are mounted in /data
- Contains:
  - lost+found - created by the OS
  - miniconda3 - conda environment (env1) created for you
  - picasso - data ←
  - RCP - compiled Unity VirtualMaze executable

Data Processing on AWS

Raw Data Organization



Stored on AWS snapshot in /data/picasso

Data Processing on AWS

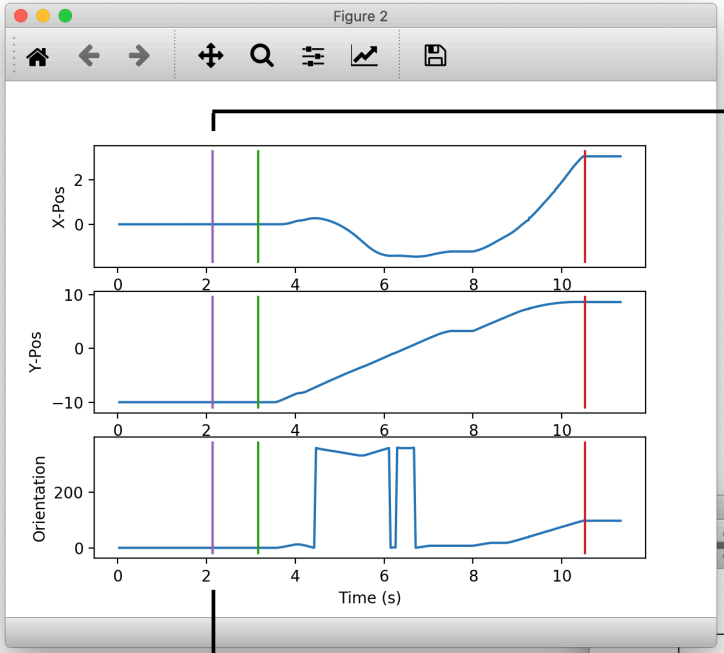
Data Processing Tools Levels

DataProcessingTools/DataProcessingTools/config.json

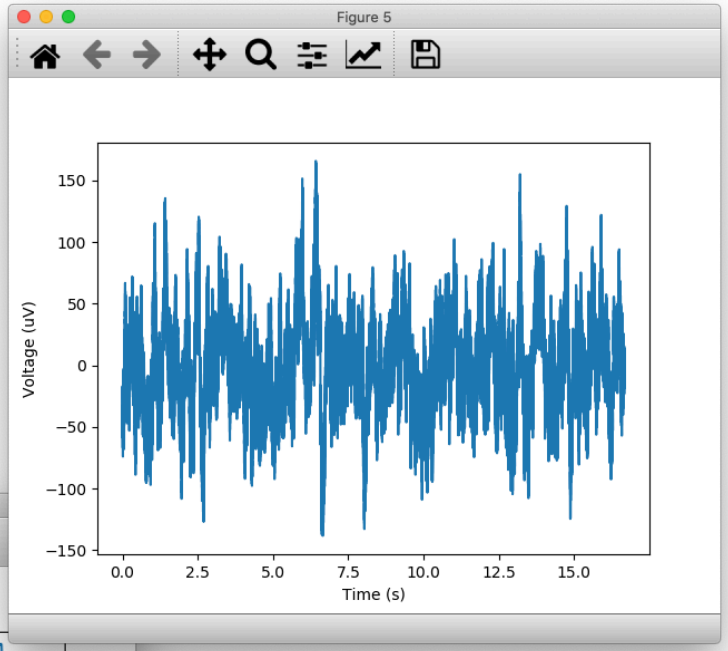
```
{"subjects": {"pattern": "([a-zA-Z]+)","order":0},
      /data
   "subject": {"pattern": "([a-zA-Z]+)", "order": 1},
      picasso
   "day": {"pattern": "([0-9]+)", "order": 2},
      20181105
   "session": {"pattern": "(session)([a-z0-9]+)","order": 3},
      session01
   "array": {"pattern": "(array)([0-9]+)", "order": 4},
      array01
   "channel": {"pattern": "(channel)([0-9]+)", "order": 5},
      channel009
   "cell": {"pattern": "(cell)([0-9]+)", "order": 6}
      cell01
}
```

Data Processing on AWS

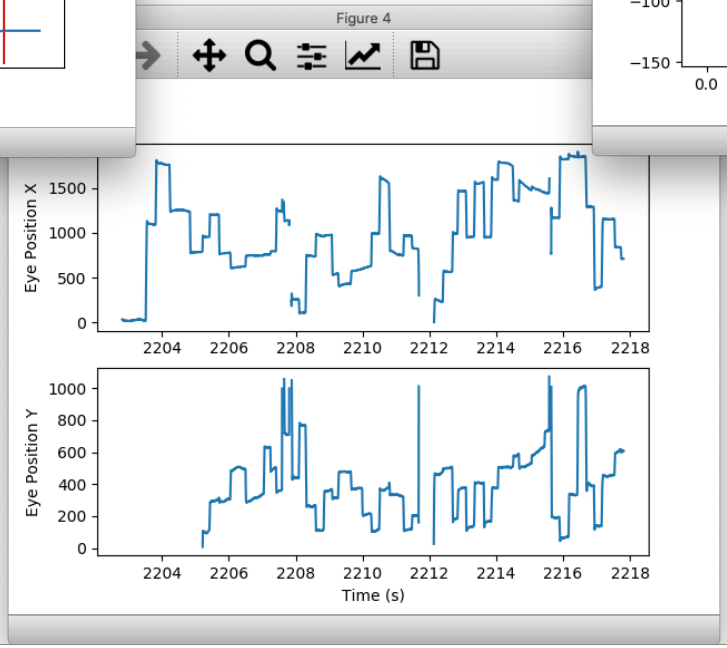
Data Timelines



Parallel Port



Eyelink



Unity

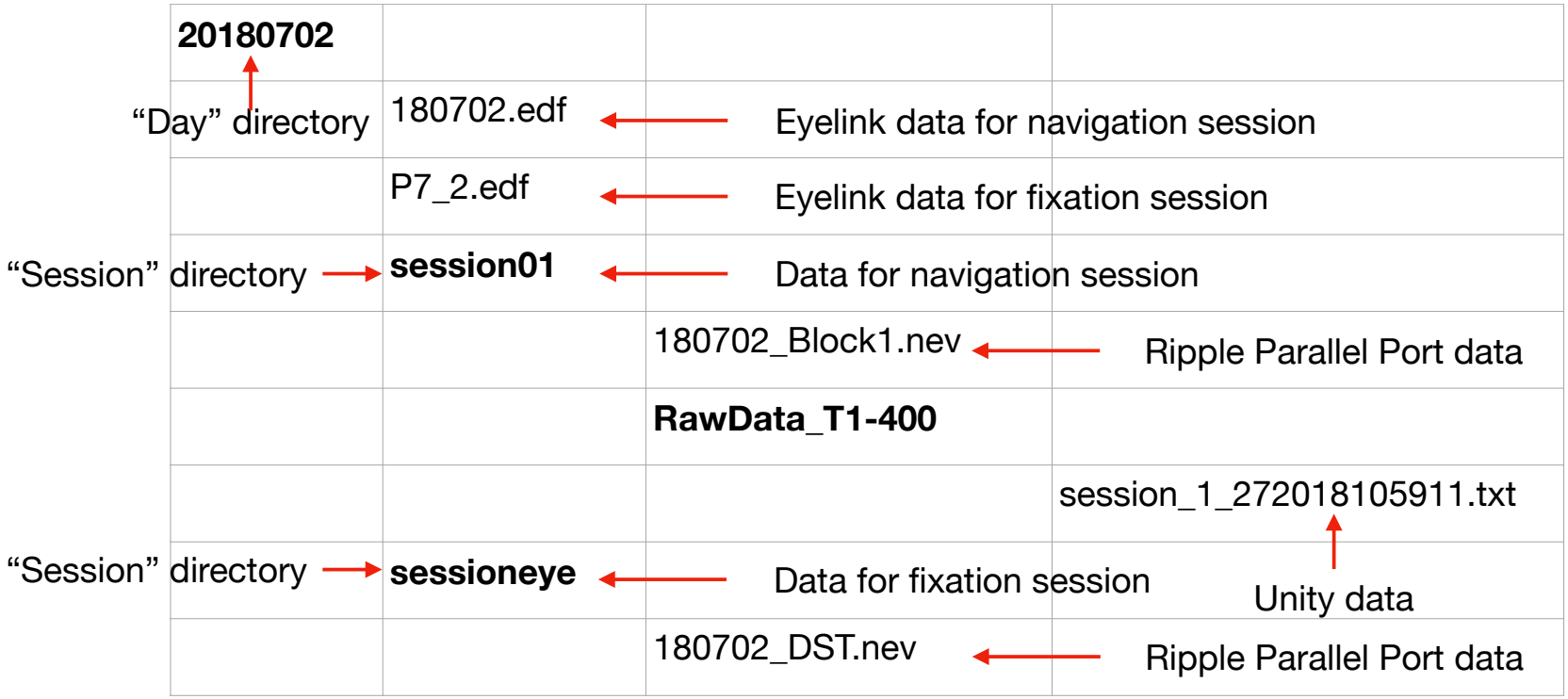
Ripple

Ethernet

Data Processing on AWS

Raw Data Organization

20180702 – 20181017



Stored on AWS snapshot in /data/picasso

Data Processing on AWS

Raw Data Organization (with Neural Data)

20181101 – 20181105

20181105			
	181105.edf		
	P11_5.edf		
	session01		
		181105_Block1.nev	
		181105_Block1.ns5	← Ripple Neural data
		RawData_T1-400	
			session_1_5112018105323.txt
	sessioneye		
		181105_DST.nev	
		181105_DST.ns5	← Ripple Neural data

Stored on AWS snapshot in /data/picasso

Data Processing on AWS

Data Processing on AWS

- AWS setup
- Raw data organization
- Data pipeline
- Data objects and data processing tools

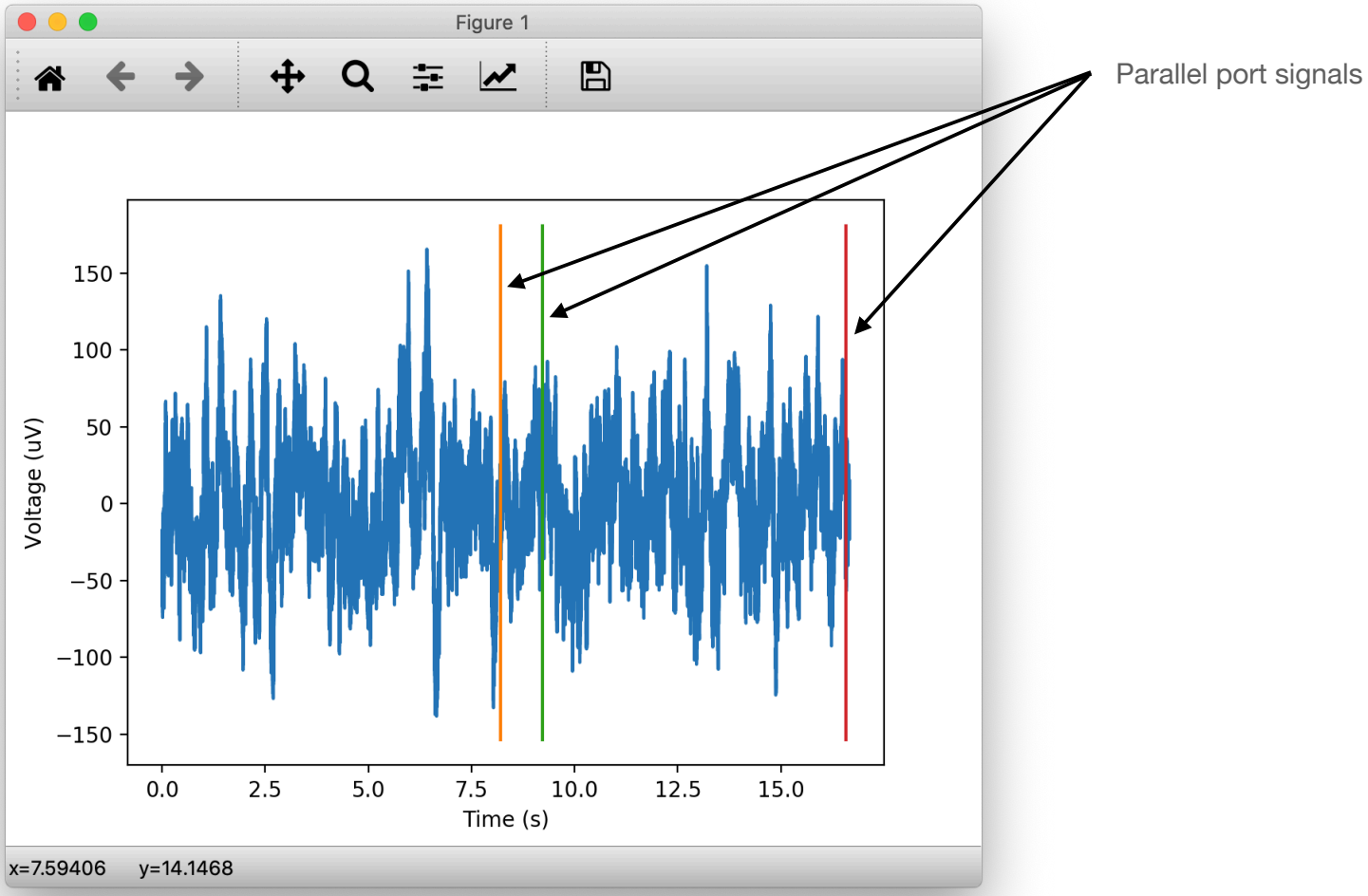
Data Processing on AWS

Data Pipeline

Data Type	Input	Function	Output
Ripple Parallel Port Data	180702_Block1.nev	RPLParallel	rplparallel_xxxx.hkl
Unity Data	session_1_272018105911.txt rplparallel_xxx.hkl	Unity	unity_xxxx.hkl
Eyelink Data	180702.edf, P7_2.edf rplparallel_xxxx.hkl	EDFSplit	eyelink_xxxx.hkl
Aligned Data	rplparallel_xxxx.hkl unity_xxxx.hkl eyelink_xxxx.hkl	aligning_objects	unity_xxxx.hkl eyelink_xxxx.hkl
Raycast Data	unity_xxxx.hkl eyelink_xxxx.hkl	raycast	bindata.hdf slist.txt …
Ripple Neural Data	181105_Block1.ns5	RPLSplit	rplraw_xxxx.hkl
Low-Pass Neural Data	rplraw_xxxx.hkl	RPLLFP	rpllfp_xxxx.hkl
High-Pass Neural Data	rplraw_xxxx.hkl	RPLHighPass	rplhighpass_xxxx.hkl
Spiketrain Data	rplhighpass_xxxx.hkl (from both navigation and fixation sessions)	mountain_batch	spiketrain_xxx.hkl

Data Processing on AWS

Ripple Parallel Port Data



Data Processing on AWS

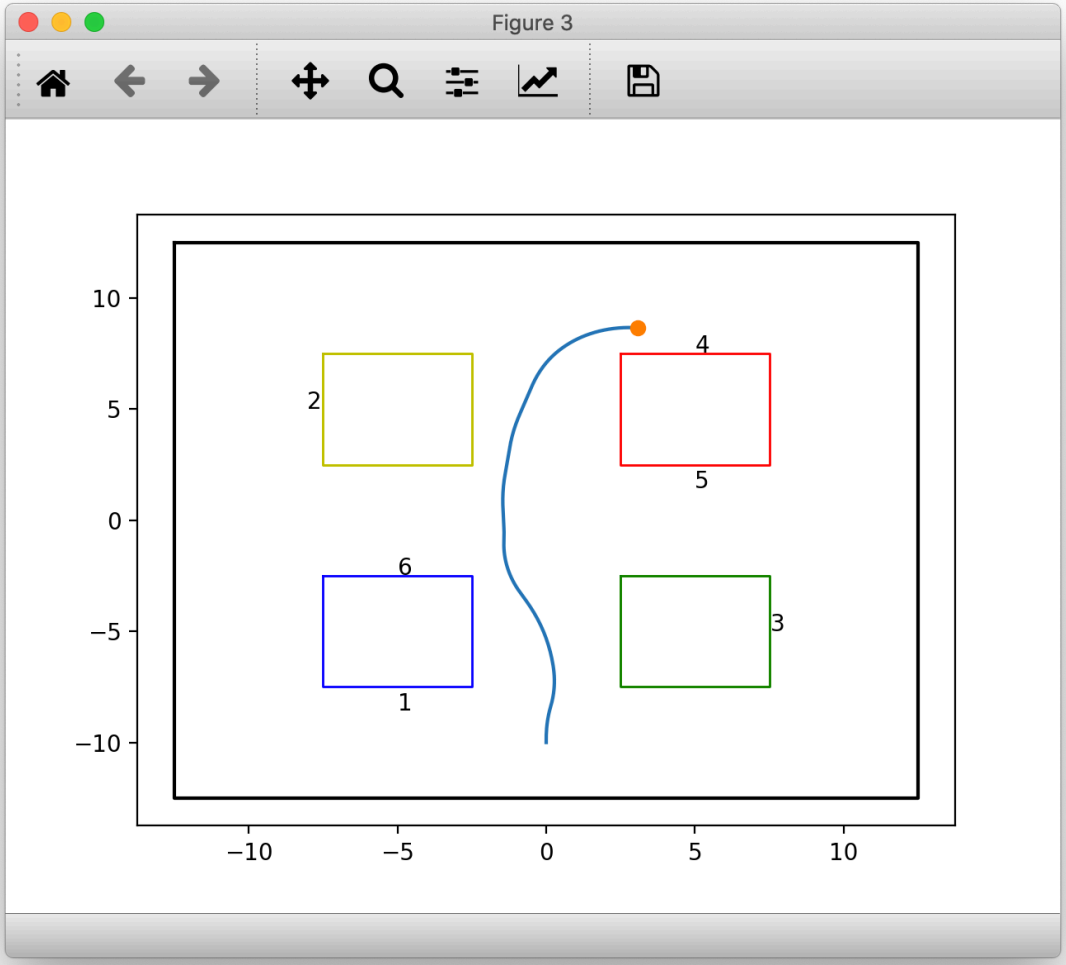
Ripple Parallel Port Data

Data Type	Input	Function	Output
Ripple Parallel Port Data	180702_Block1.nev	RPLParallel	rplparallel_xxxx.hkl

20180702			
	180702.edf		
	P7_2.edf		
	session01		
		180702_Block1.nev	RPLParallel (from rplparallel.py) Called from the Session Directory
		rplparallel_72a7.hkl	
		RawData_T1-400	
			session_1_272018105911.txt
	sessioneye		
		180702_DST.nev	RPLParallel (from rplparallel.py) Called from the Session Directory
		rplparallel_72a7.hkl	

Data Processing on AWS

Unity Data



Data Processing on AWS

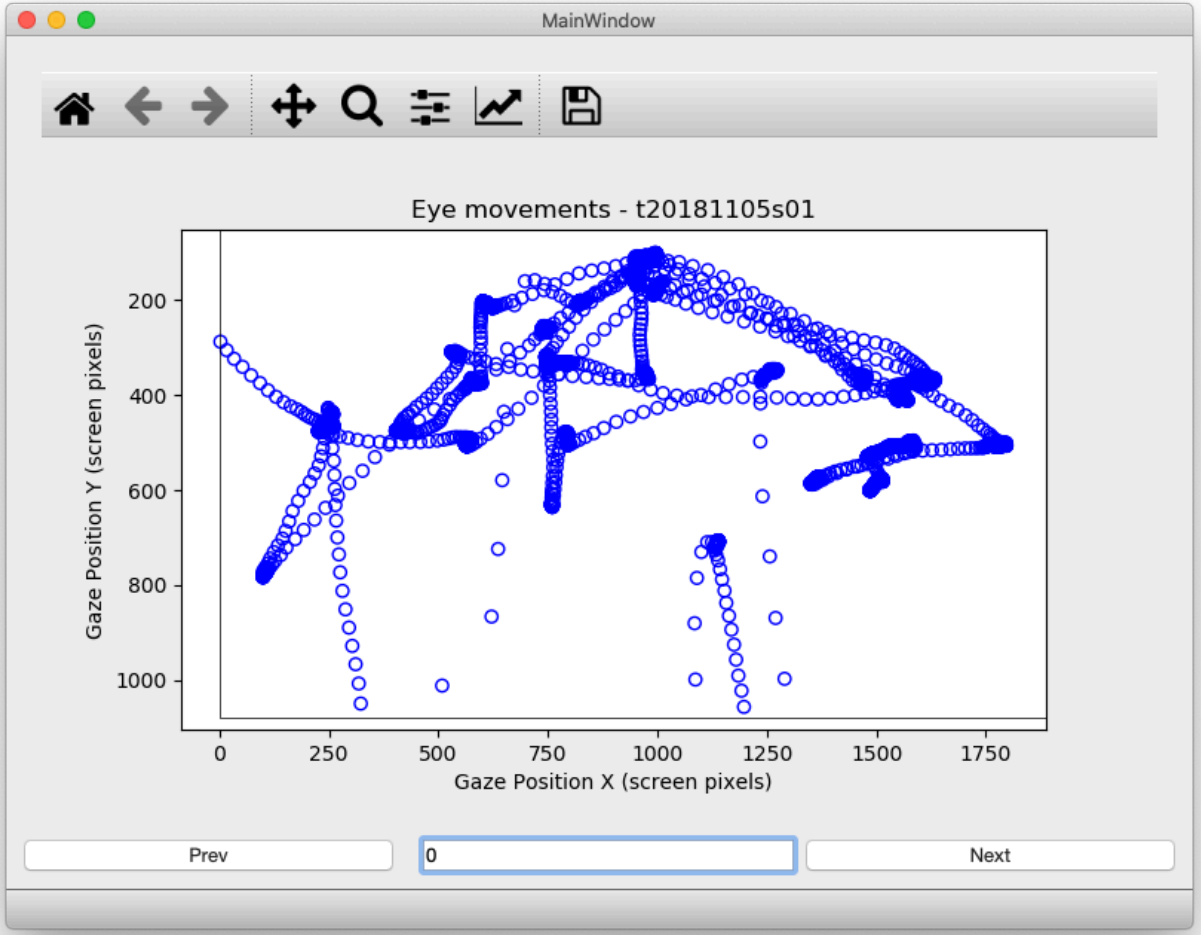
Unity Data

Data Type	Input	Function	Output
Unity Data	session_1_272018105911.txt rplparallel_xxx.hkl	Unity	unity_xxxx.hkl

20180702			
	180702.edf		
	P7_2.edf		
	session01		
		180702_Block1.nev	
		rplparallel_72a7.hkl	
		RawData_T1-400	
			session_1_272018105911.txt
		unity_5878.hkl	Unity (from unity.py) Called from the Session Directory
	sessioneye		
		180702_DST.nev	
		rplparallel_72a7.hkl	

Data Processing on AWS

Eyelink Data



Data Processing on AWS

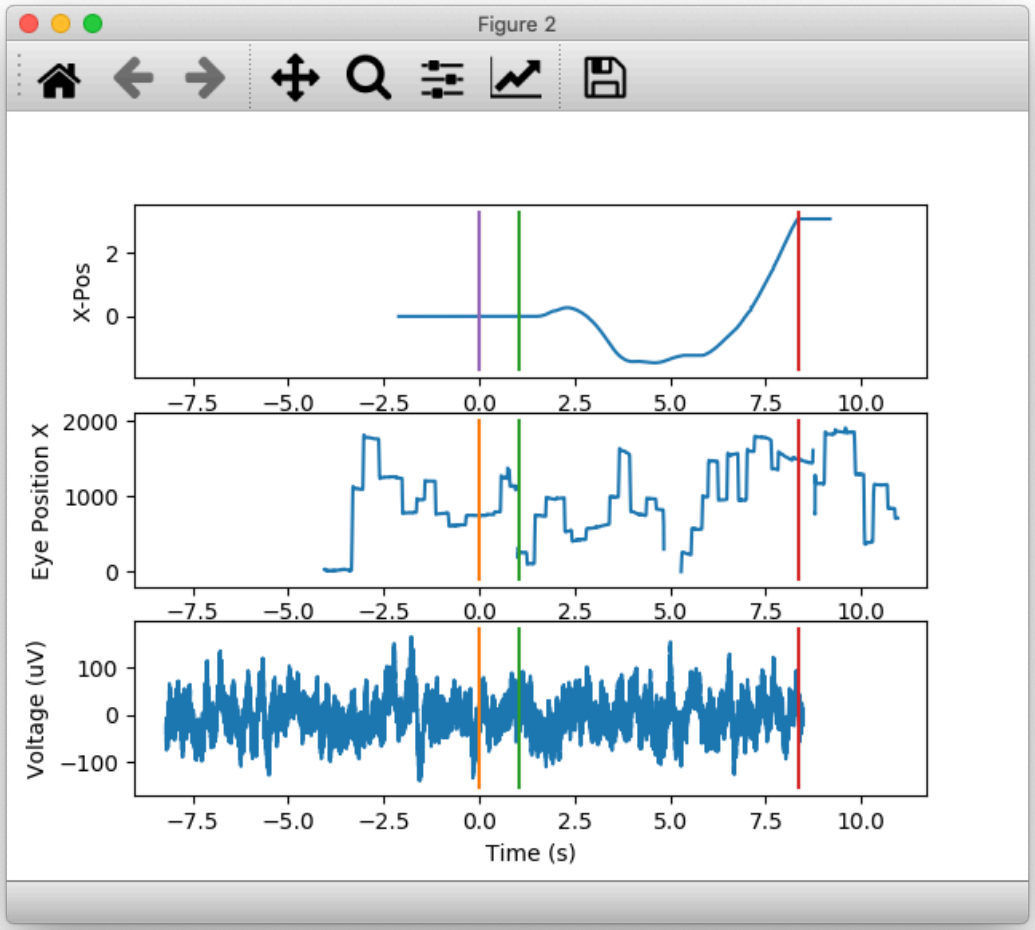
Eyelink Data

Data Type	Input	Function	Output
Eyelink Data	180702.edf, P7_2.edf rplparallel_xxxx.hkl	EDFSplit	eyelink_xxxx.hkl

20180702		EDFSplit (from eyelink.py) Called from the Day Directory	
	180702.edf		
	P7_2.edf		
	session01		
		180702_Block1.nev	
		eyelink_c5f2.hkl	
		rplparallel_72a7.hkl	
		RawData_T1-400	
			session_1_272018105911.txt
		unity_5878.hkl	
	sessioneye		
		180702_DST.nev	
		eyelink_c5f2.hkl	
		rplparallel_72a7.hkl	

Data Processing on AWS

Aligned Data



Data Processing on AWS

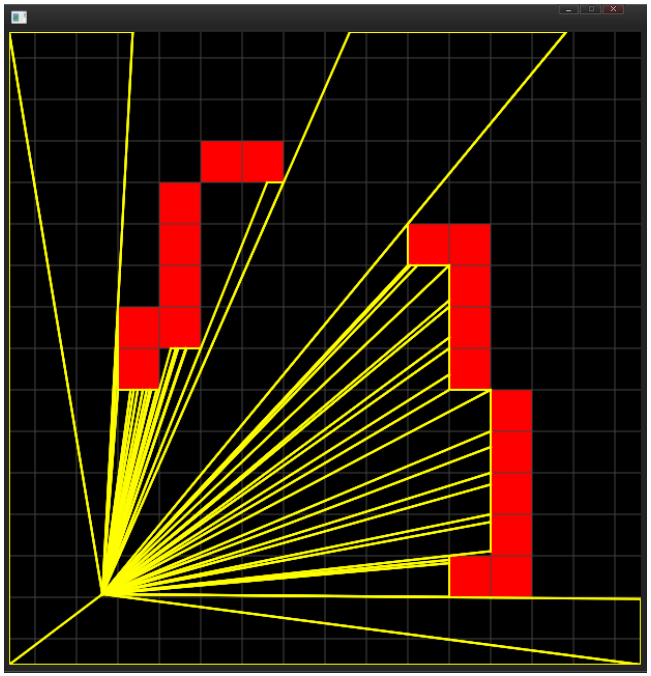
Aligned Data

Data Type	Input	Function	Output
Aligned Data	rplparallel_xxxx.hkl unity_xxxx.hkl eyelink_xxxx.hkl	aligning_objects	unity_xxxx.hkl eyelink_xxxx.hkl

20180702			
	180702.edf		
	P7_2.edf		
	session01		
		180702_Block1.nev	
		eyelink_c5f2.hkl	aligning_objects (from aligning_objects.py) Called from the Session Directory
		rplparallel_72a7.hkl	
		RawData_T1-400	
			session_1_272018105911.txt
		unity_5878.hkl	
	sessioneye		
		180702_DST.nev	
		eyelink_c5f2.hkl	

Data Processing on AWS

Raycast Data



Data Processing on AWS

Raycast Data

Data Type	Input	Function	Output
Raycast Data	unity_xxxx.hkl eyelink_xxxx.hkl	raycast	bindata.hdf slist.txt …

20180702			
	180702.edf		
	P7_2.edf		
	session01		
		180702_Block1.nev	
		bindata.hdf	
		eyelink_c5f2.hkl	
		logs.txt	raycast (from raycast.py)
		missingData.csv	Called from the Session Directory
		rplparallel_72a7.hkl	
		RawData_T1-400	
			session_1_272018105911.txt
		slist.txt	
		unity_5878.hkl	
		unityfile_eyelink.csv	
		VirtualMazeBatchLog.txt	
	sessioneye		

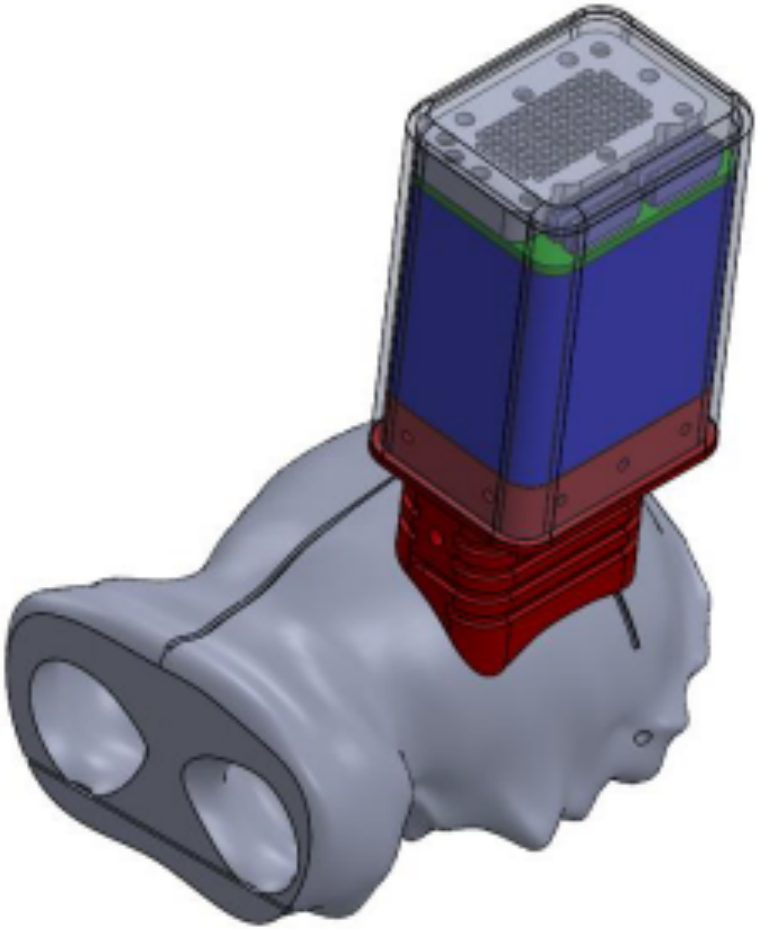
Data Processing on AWS

Data Pipeline

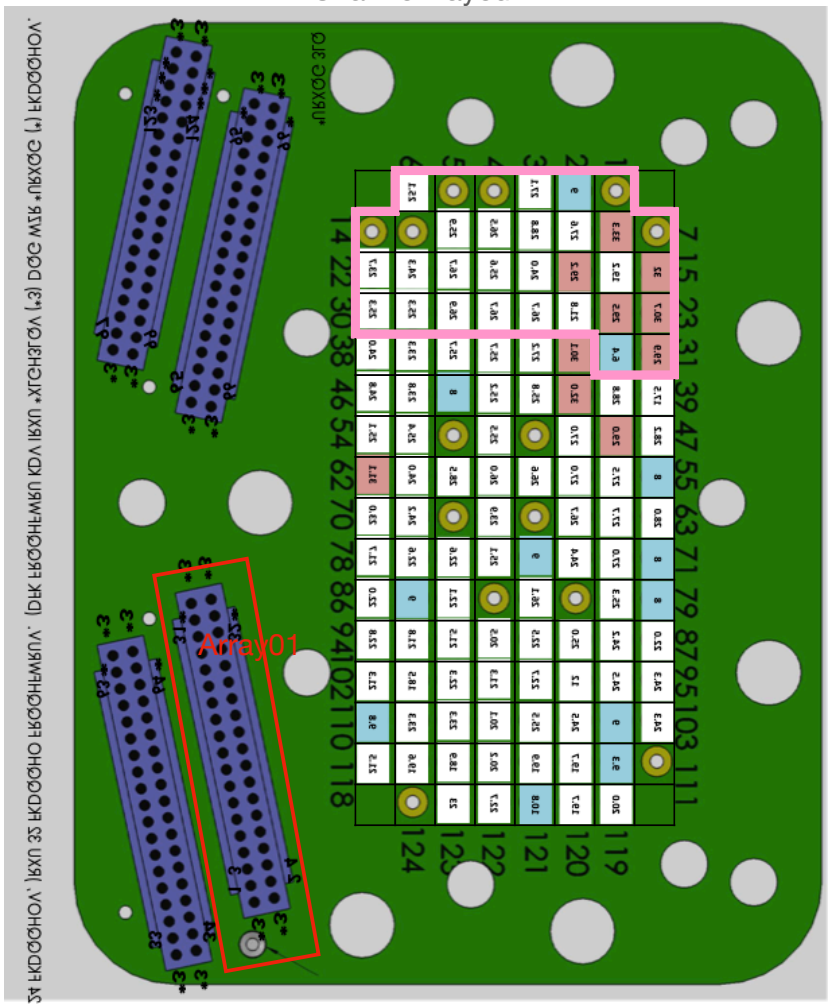
Data Type	Input	Function	Output
Ripple Parallel Port Data	180702_Block1.nev	RPLParallel	rplparallel_xxxx.hkl
Unity Data	session_1_272018105911.txt rplparallel_xxx.hkl	Unity	unity_xxxx.khl
Eyelink Data	180702.edf, P7_2.edf rplparallel_xxxx.hkl	EDFSplit	eyelink_xxxx.hkl
Aligned Data	rplparallel_xxxx.hkl unity_xxxx.hkl eyelink_xxxx.hkl	aligning_objects	unity_xxxx.hkl eyelink_xxxx.hkl
Raycast Data	unity_xxxx.hkl eyelink_xxxx.hkl	raycast	bindata.hdf slist.txt …
Ripple Neural Data	181105_Block1.ns5	RPLSplit	rplraw_xxxx.hkl
Low-Pass Neural Data	rplraw_xxxx.hkl	RPLLFP	rpllfp_xxxx.hkl
High-Pass Neural Data	rplraw_xxxx.hkl	RPLHighPass	rplhighpass_xxxx.hkl
Spiketrain Data	rplhighpass_xxxx.hkl (from both navigation and fixation sessions)	mountain_batch	spiketrain_xxx.hkl

Data Processing on AWS

Ripple Neural Data

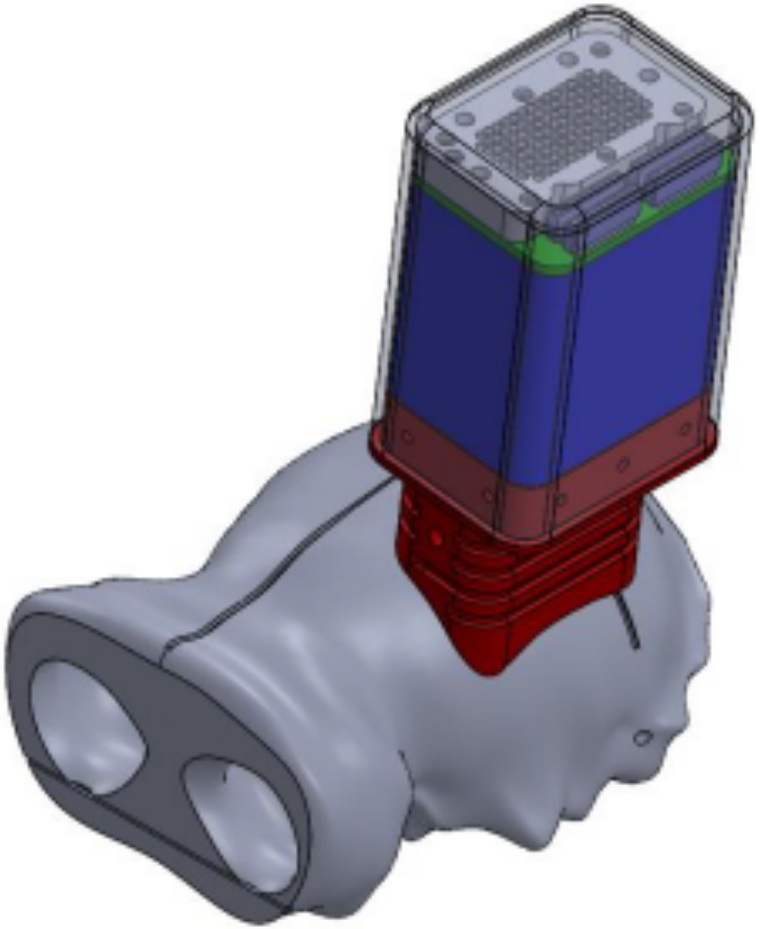


Channel Layout

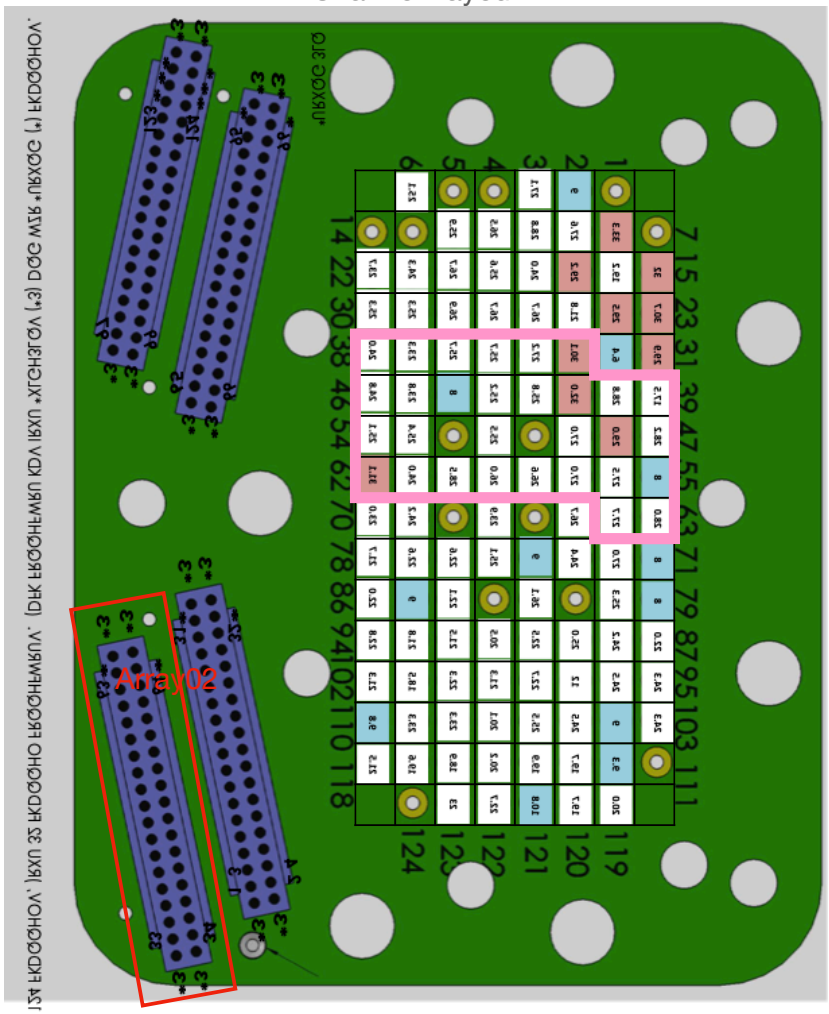


Data Processing on AWS

Ripple Neural Data

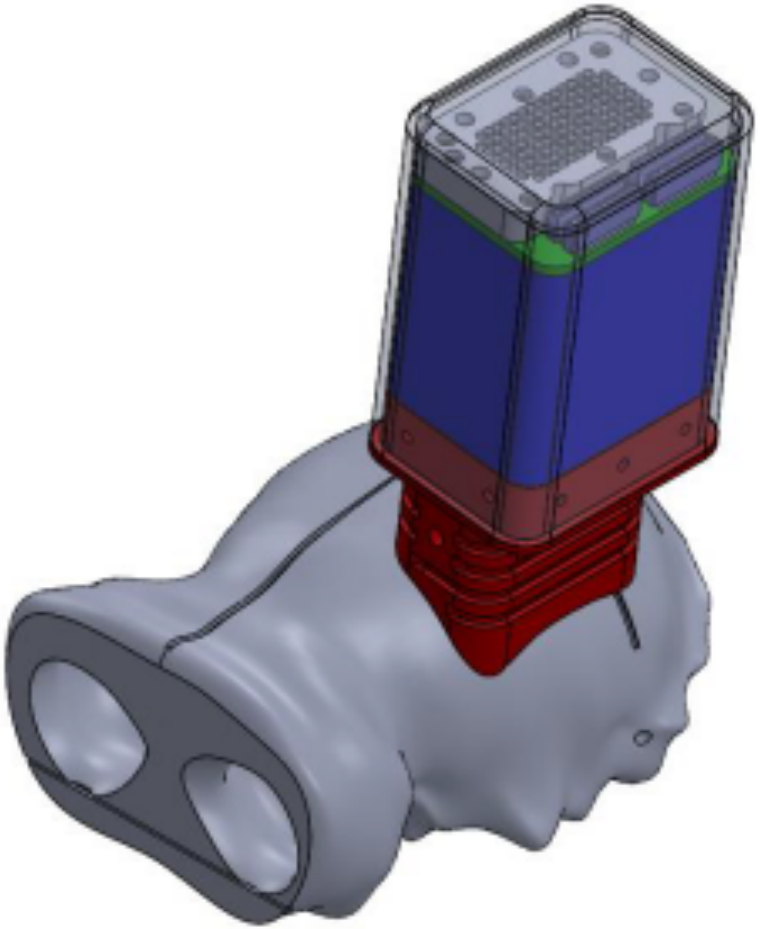


Channel Layout

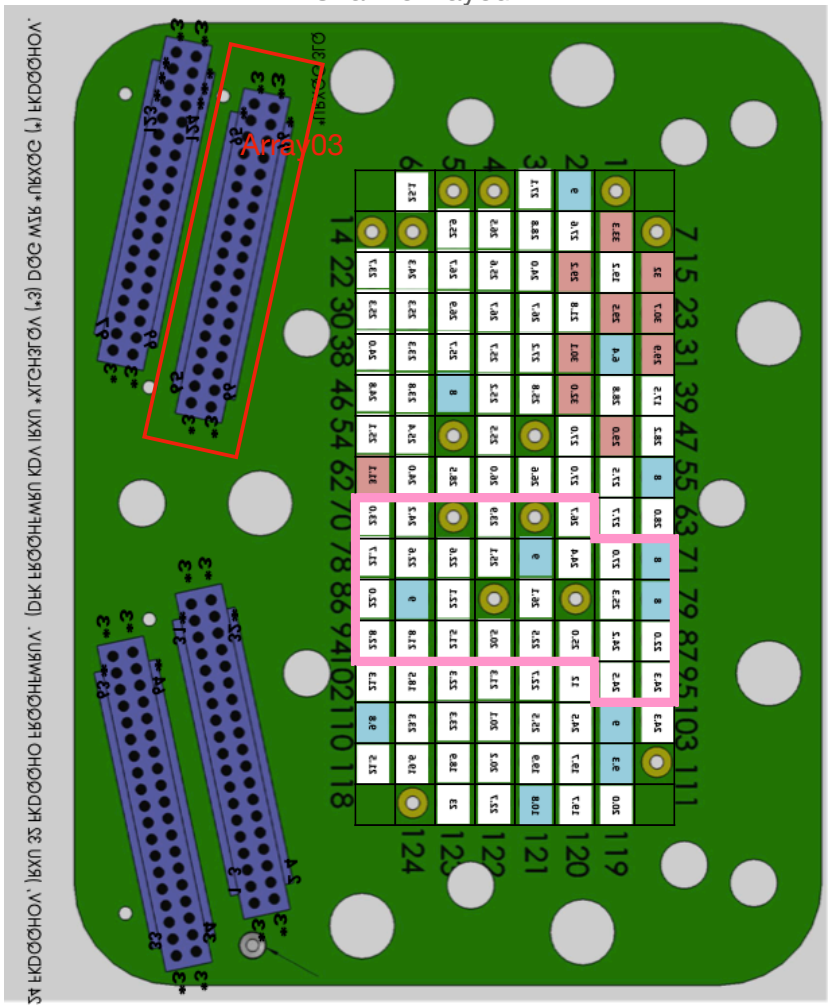


Data Processing on AWS

Ripple Neural Data

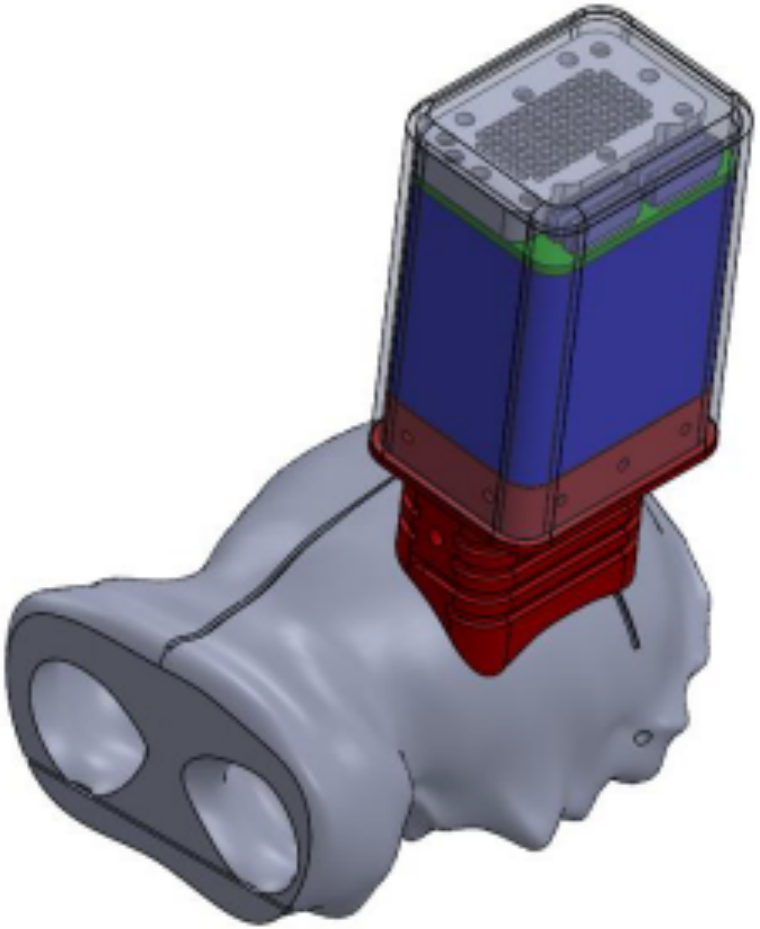


Channel Layout

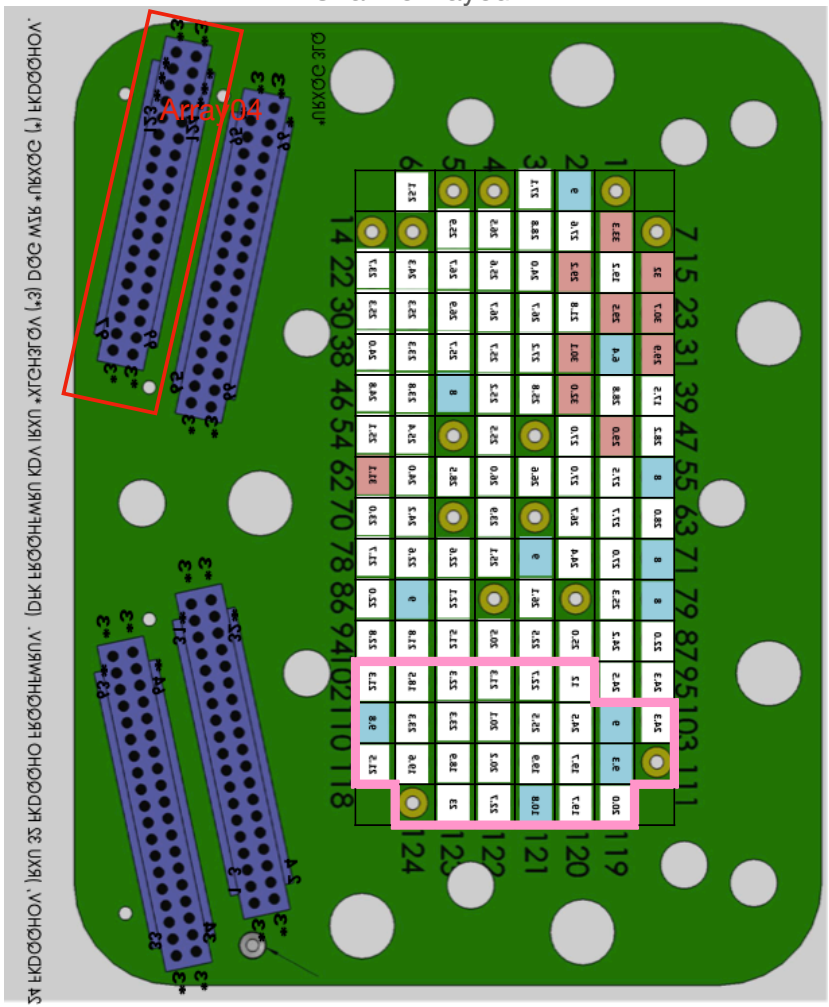


Data Processing on AWS

Ripple Neural Data

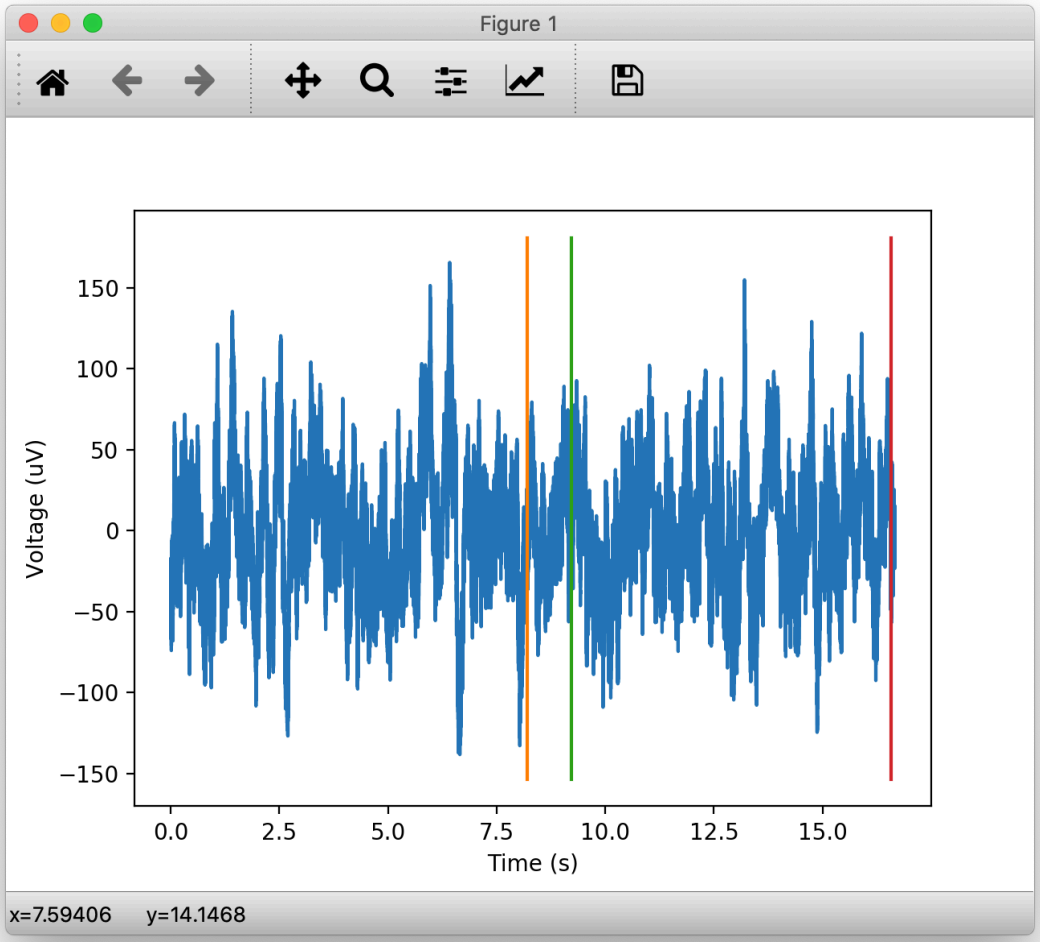


Channel Layout



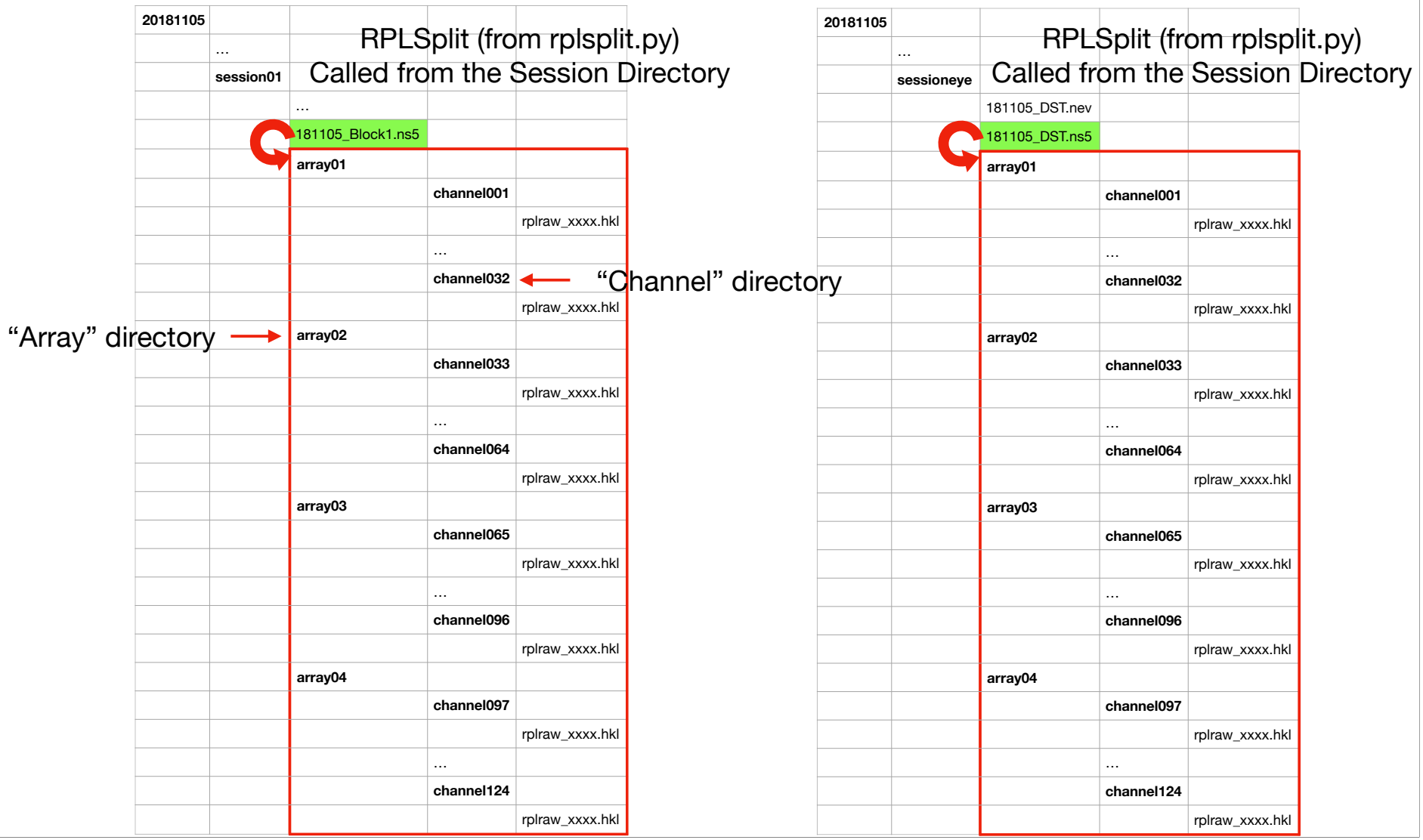
Data Processing on AWS

Ripple Neural Data



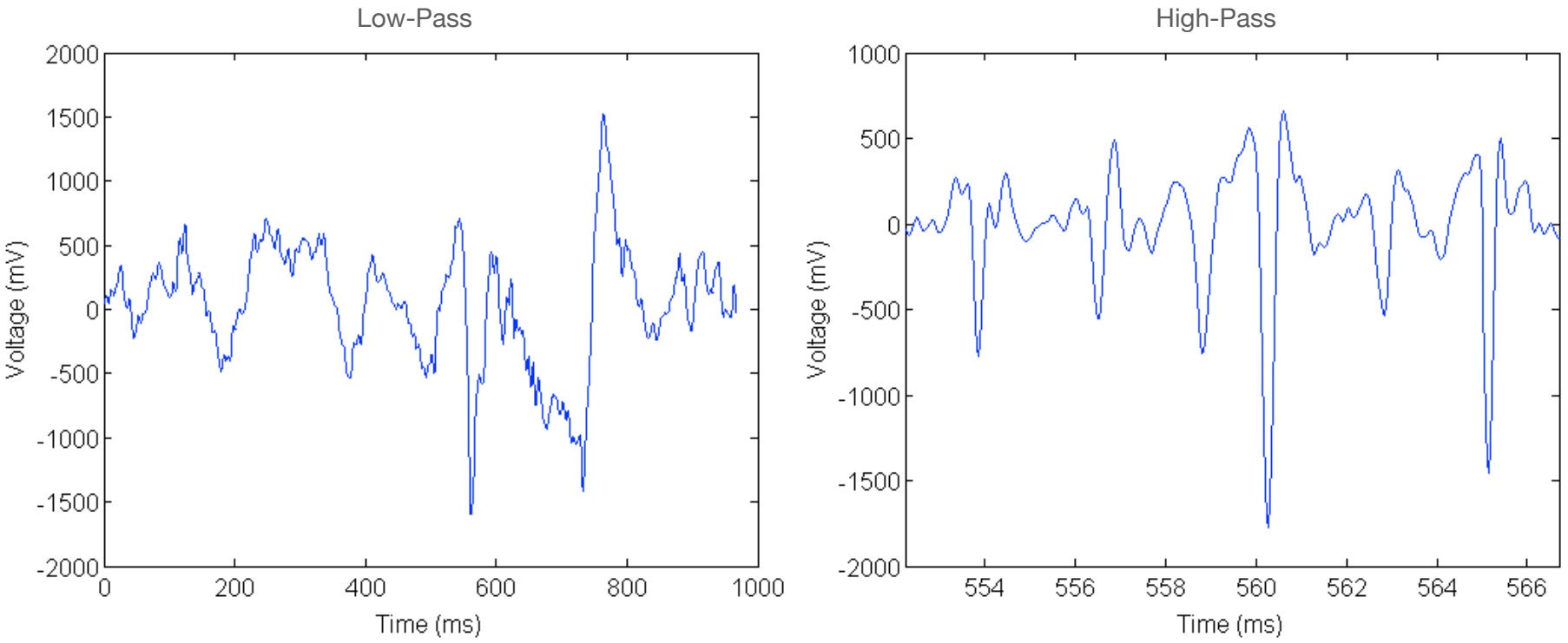
Data Processing on AWS

Ripple Neural Data



Data Processing on AWS

Ripple Low-Pass & High-Pass Data



Data Processing on AWS

Ripple Low-Pass & High-Pass Data

Data Type	Input	Function	Output
Low-Pass Neural Data	rplraw_xxxx.hkl	RPLLFP	rpllfp_xxxx.hkl
High-Pass Neural Data	rplraw_xxxx.hkl	RPLHighPass	rplhighpass_xxxx.hkl

	array01		
		channel001	
RPLLFP (from rpllfp.py) Called from the Channel Directory			rplraw_xxxx.hkl
			rpllfp_xxxx.hkl
		…	
	array02		
		channel033	
RPLLFP (from rpllfp.py) Called from the Channel Directory			rplraw_xxxx.hkl
			rpllfp_xxxx.hkl
		…	
	…		

	array01		
		channel001	
RPLHighPass (from rplhighpass.py) Called from the Channel Directory			rplraw_xxxx.hkl
			rpllfp_xxxx.hkl
			rplhighpass_xxxx.hkl
		…	
	array02		
		channel033	
RPLHighPass (from rplhighpass.py) Called from the Channel Directory			rplraw_xxxx.hkl
			rpllfp_xxxx.hkl
			rplhighpass_xxxx.hkl
	…		

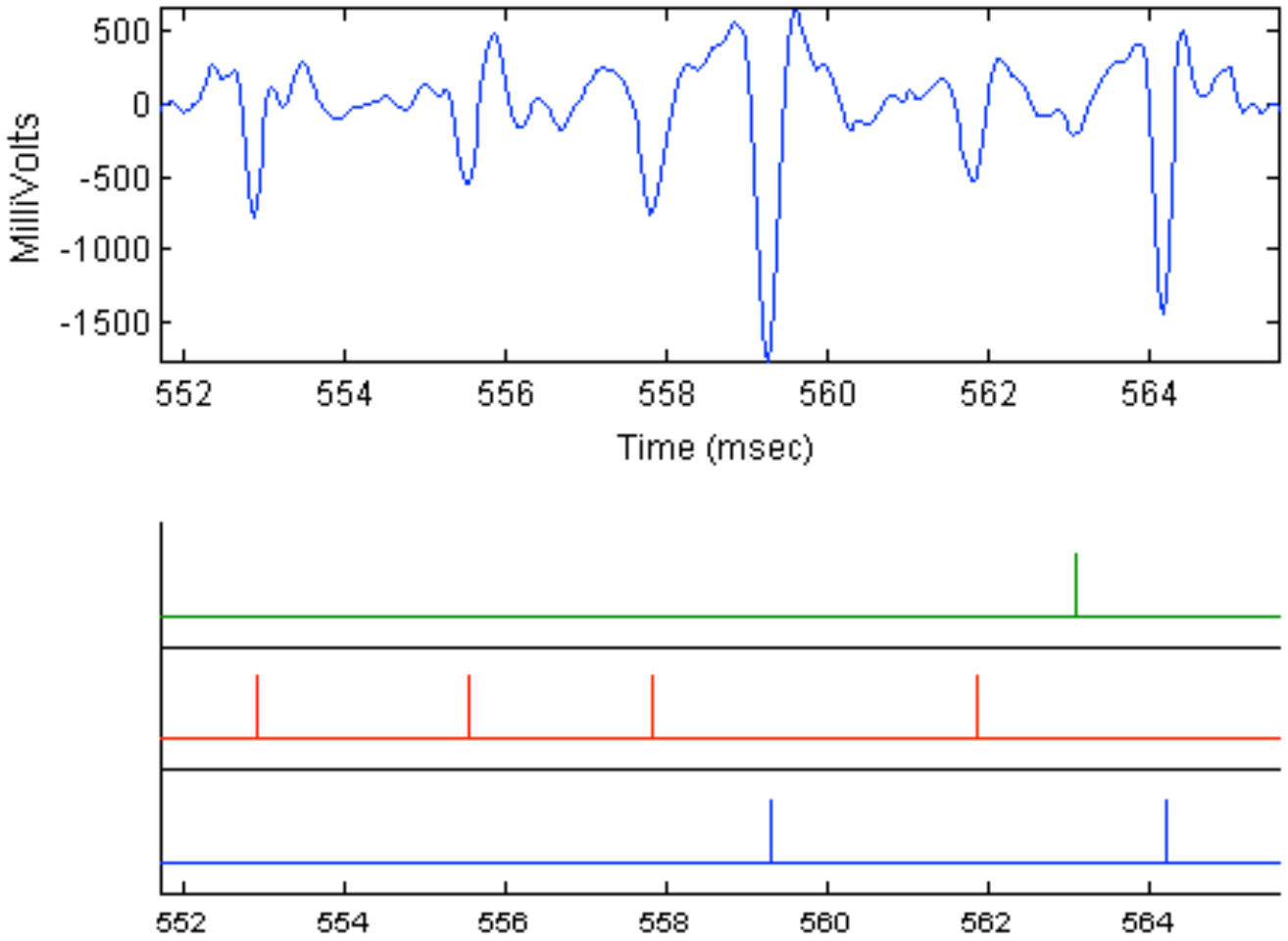
Data Processing on AWS

Data Pipeline

Data Type	Input	Function	Output
Ripple Parallel Port Data	180702_Block1.nev	RPLParallel	rplparallel_xxxx.hkl
Unity Data	session_1_272018105911.txt rplparallel_xxx.hkl	Unity	unity_xxxx.khl
Eyelink Data	180702.edf, P7_2.edf rplparallel_xxxx.hkl	EDFSplit	eyelink_xxxx.hkl
Aligned Data	rplparallel_xxxx.hkl unity_xxxx.hkl eyelink_xxxx.hkl	aligning_objects	unity_xxxx.hkl eyelink_xxxx.hkl
Raycast Data	unity_xxxx.hkl eyelink_xxxx.hkl	raycast	bindata.hdf slist.txt …
Ripple Neural Data	181105_Block1.ns5	RPLSplit	rplraw_xxxx.hkl
Low-Pass Neural Data	rplraw_xxxx.hkl	RPLLFP	rpllfp_xxxx.hkl
High-Pass Neural Data	rplraw_xxxx.hkl	RPLHighPass	rplhighpass_xxxx.hkl
Spiketrain Data	rplhighpass_xxxx.hkl (from both navigation and fixation sessions)	mountain_batch	spiketrain_xxx.hkl

Data Processing on AWS

Spike Sorted Data



Data Processing on AWS

Spike Sorted Data

session01				
	array01			
		channel001		
			…	
			rplhighpass_xxxx.hkl	
			cell01	
				spiketrain_xxxx.hkl
				spiketrain.csv
sessioneye			mountain_batch (from mountain_batch.py) Called from the Channel Directory in session01 directory	
	array01			
		channel001		
			…	
			rplhighpass_xxxx.hkl	
			cell01	
				spiketrain_xxxx.hkl
				spiketrain.csv

Data Processing on AWS

Data Processing on AWS

- AWS setup
- Raw data organization
- Data pipeline
- Data objects and data processing tools

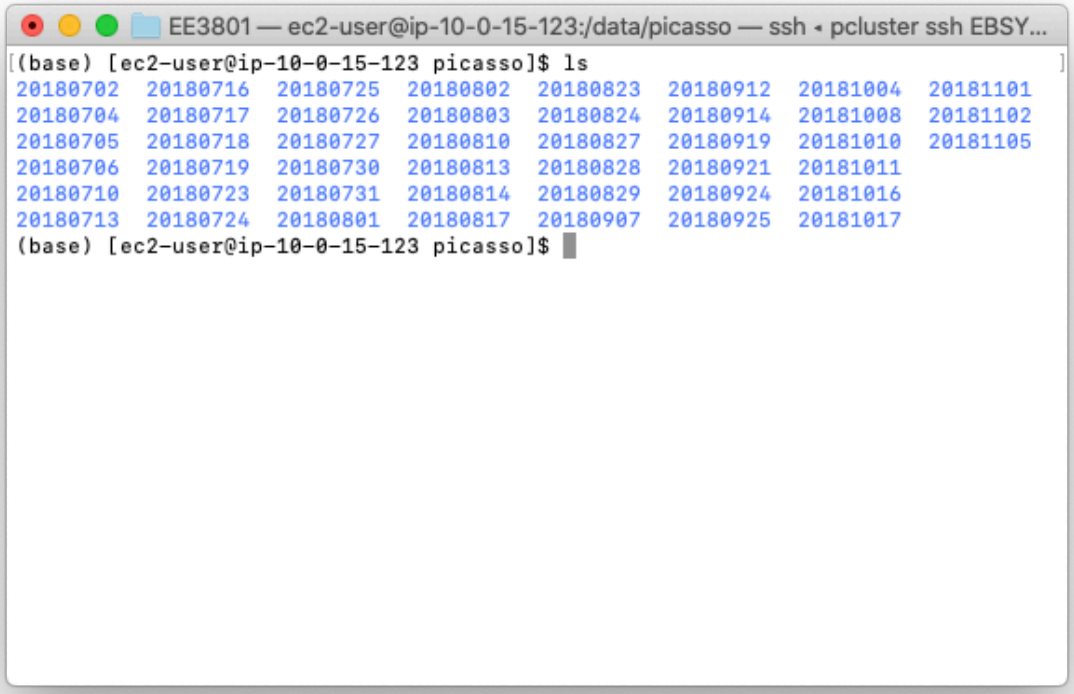
Data Processing on AWS

Creating Data Objects

Data Type	Directory	Function
Ripple Parallel Port Data	session01 sessioneye	pyh.RPLParallel(saveLevel=1)
Unity Data	session01	pyh.Unity(saveLevel=1)
Eyelink Data	20181105	pyh.EDFSplit()
Aligned Data	session01	pyh.aligning_objects()
Raycast Data	session01	pyh.raycast(1)
Ripple Neural Data	session01 sessioneye	pyh.RPLSplit(channel=[9])
Low-Pass Neural Data	session01/array01/ channel009 sessioneye/array01/ channel009	pyh.RPLLFP(saveLevel=1)
High-Pass Neural Data	session01/array01/ channel009 sessioneye/array01/ channel009	pyh.RPLHighPass(saveLevel=1)
Spiketrain Data	session01/array01/ channel009	mountain_batch.mountain_batch()

Data Processing on AWS

Raw Data Organization



But there are lots of directories …

Data Processing on AWS

Data Processing Tools

- DPT contains functions to traverse a directory hierarchy to perform data processing
- Uses object-oriented programming (OOP) primarily to re-use code
- Reduces code needed to implement new data processing
- Each data analysis is implemented as an object
- Object has a directory level in which the object should be created

```
class RPLParallel(DPT.DPObject):

    filename = 'rplparallel.hkl'
    argsList = []
    level = 'session'
```

Object should be created in the Session directory

Data Processing on AWS

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-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 Processing on AWS

Data Processing Tools

```
In[ ]: pwd
Out[ ]: /Users/syen/Documents/picasso/20181105/
In[ ]: DPT.objects.processDirs(dirs=None, objtype=pyh.RPLParallel, saveLevel=1)
```

20181105			
	181105.edf		
	P11_5.edf		
	session01		
		181105_Block1.nev	
		rplparallel_72a7.hkl	
		RawData_T1-400	
			session_1_5112018105911.txt
	sessioneye		
		181105_DST.nev	
		rplparallel_72a7.hkl	

```
class RPLParallel(DPT.DPObject):

    filename = 'rplparallel.hkl'
    argsList = []
    level = 'session'
```

Data Processing on AWS

Data Processing Tools

```
In[ ]: DPT.objects.processDirs(dirs=None, objtype=pyh.RPLSplit,
channel=[9, 31, 34, 56, 72, 93, 119, 120])
```

```
class RPLSplit(DPT.DPObject):

                    filename = 'rplsplit.hkl'
                    argsList = [('channel', [*range(1, 125)]),
                                ('SkipHPC', True),
                                ('SkipLFP', True),
                                ('SkipHighPass', True),
                                ('SkipSort', True),
                                ('SkipParallel', True)]
                    level = 'session'
```

Object should be created in the Session directory

Data Processing on AWS

Data Processing Tools

20181105				
	…			
	session01			
		…		
		181105_Block1.ns5		

20181105				
	…			
	sessioneye			
		…		
		181105_DST.ns5		

Data Processing on AWS

Data Processing Tools

20181105				
	…			
	session01			
		…		
		181105_Block1.ns5		
		array01		
			channel009	
				rplraw_xxxx.hkl
			channel031	
				rplraw_xxxx.hkl
		array02		
			channel034	
				rplraw_xxxx.hkl
			channel056	
				rplraw_xxxx.hkl
		array03		
			channel072	
				rplraw_xxxx.hkl
			channel093	
				rplraw_xxxx.hkl
		array04		
			channel119	
				rplraw_xxxx.hkl
			channel120	
				rplraw_xxxx.hkl

20181105				
	…			
	sessioneye			
		…		
		181105_DST.ns5		

Data Processing on AWS

Data Processing Tools

20181105				
	…			
	session01			
		…		
		181105_Block1.ns5		
		array01		
			channel009	
				rplraw_xxxx.hkl
			channel031	
				rplraw_xxxx.hkl
		array02		
			channel034	
				rplraw_xxxx.hkl
			channel056	
				rplraw_xxxx.hkl
		array03		
			channel072	
				rplraw_xxxx.hkl
			channel093	
				rplraw_xxxx.hkl
		array04		
			channel119	
				rplraw_xxxx.hkl
			channel120	
				rplraw_xxxx.hkl

20181105				
	…			
	sessioneye			
		…		
		181105_DST.ns5		

Data Processing on AWS

Data Processing Tools

20181105				
	…			
	session01			
		…		
		181105_Block1.ns5		
		array01		
			channel009	
				rplraw_xxxx.hkl
			channel031	
				rplraw_xxxx.hkl
		array02		
			channel034	
				rplraw_xxxx.hkl
			channel056	
				rplraw_xxxx.hkl
		array03		
			channel072	
				rplraw_xxxx.hkl
			channel093	
				rplraw_xxxx.hkl
		array04		
			channel119	
				rplraw_xxxx.hkl
			channel120	
				rplraw_xxxx.hkl

20181105				
	…			
	sessioneye			
		…		
		181105_DST.ns5		
		array01		
			channel009	
				rplraw_xxxx.hkl
			channel031	
				rplraw_xxxx.hkl
		array02		
			channel034	
				rplraw_xxxx.hkl
			channel056	
				rplraw_xxxx.hkl
		array03		
			channel072	
				rplraw_xxxx.hkl
			channel093	
				rplraw_xxxx.hkl
		array04		
			channel119	
				rplraw_xxxx.hkl
			channel120	
				rplraw_xxxx.hkl

Data Processing on AWS

Data Processing Tools

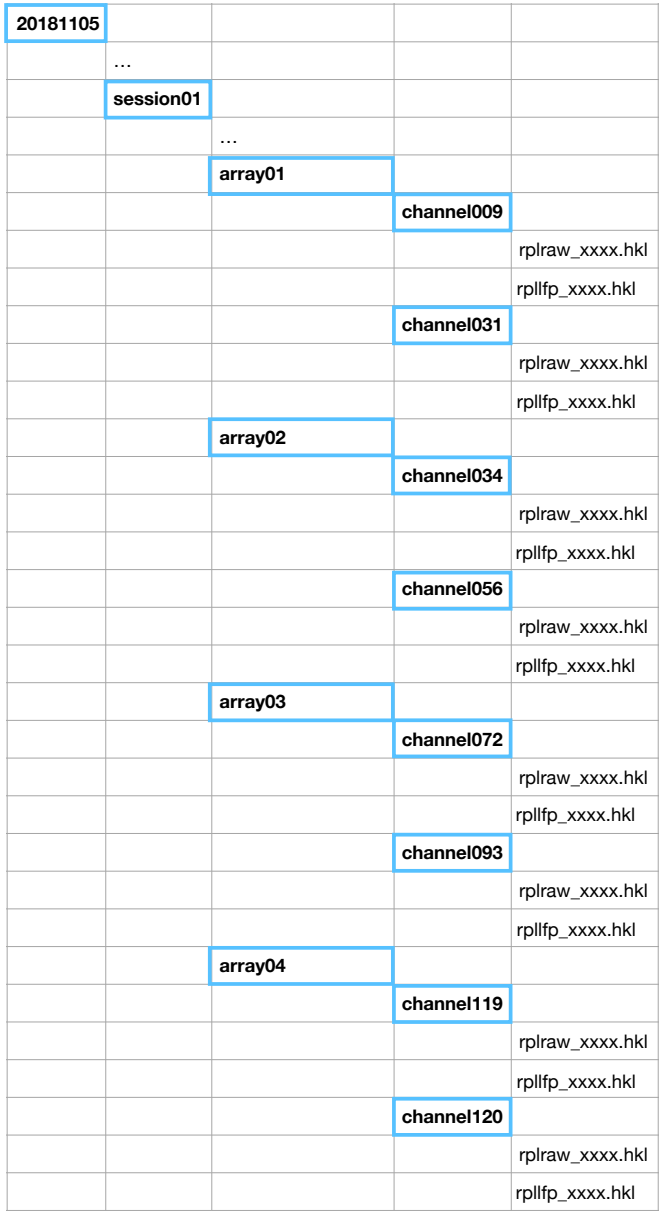
In[ ]: DPT.objects.processDirs(dirs=None, objtype=pyh.RPLLFP, saveLevel=1)

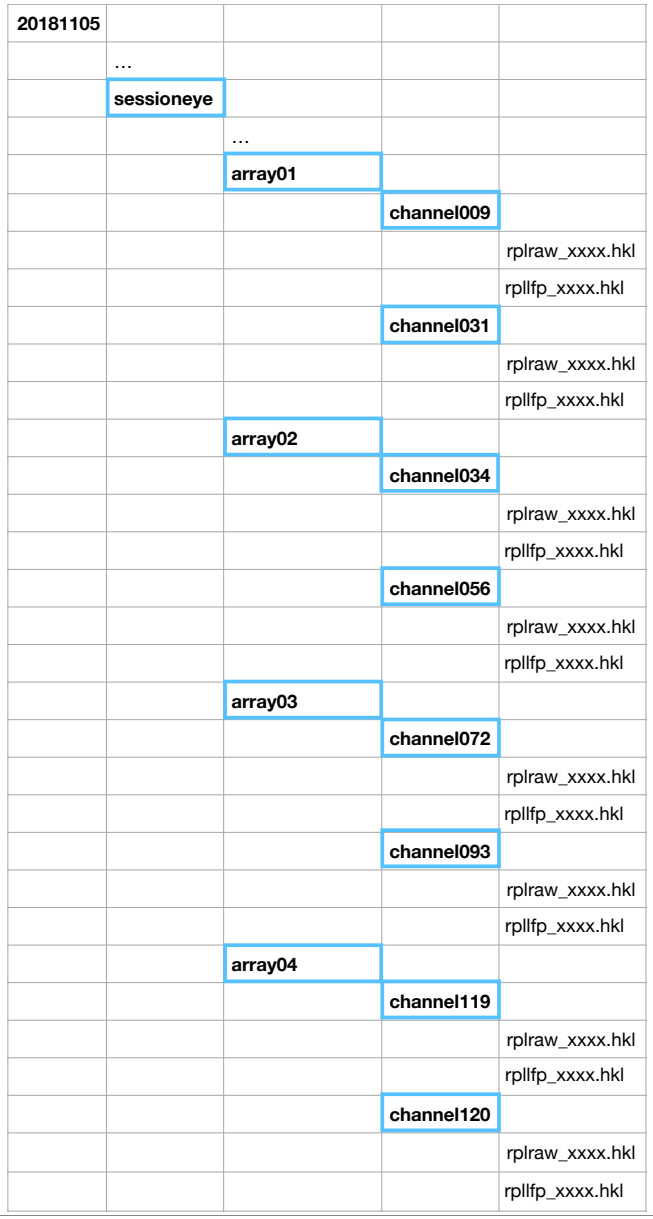
```
class RPLLFP(DPT.DPObject):

    filename = "rpllfp.hkl"
    argsList = [('ResampleRate', 1000),
                ('LFPOrder', 8),
                ('LowPassFrequency', [1, 150])]
    level = 'channel'
```

Object should be created in the Channel directory

Data Processing on AWS





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"
```

Data Processing on AWS

Data Processing Tools

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();');

session01				
	array01			
		channel001		
			…	
			rplhighpass_xxxx.hkl	
sessioneye			mountain_batch (from mountain_batch.py)	
	array01		Called from the Channel Directory in	
		channel001	session01 directory	
			…	
			rplhighpass_xxxx.hkl	

Data Processing on AWS

Data Processing Tools

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();');

session01				
	array01			
		channel001		
			…	
			rplhighpass_xxxx.hkl	
sessioneye			mountain_batch (from mountain_batch.py)	
	array01		Called from the Channel Directory in	
		channel001	session01 directory	
			…	
			rplhighpass_xxxx.hkl	

Data Processing on AWS

Data Processing Tools

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();');

session01				
	array01			
		channel001		
			…	
			rplhighpass_xxxx.hkl	
			cell01	
				spiketrain_xxxx.hkl
				spiketrain.csv
sessioneye			mountain_batch (from mountain_batch.py)	
	array01		Called from the Channel Directory in	
		channel001	session01 directory	
			…	
			rplhighpass_xxxx.hkl	
			cell01	
				spiketrain_xxxx.hkl
				spiketrain.csv

AWS ParallelCluster

AWS Credits

- Every student given US$400 of AWS budget
- Monitor your AWS charges carefully
- Your account will be disabled once you exceed US$400
- Include your AWS charges and budget balance in every lab report

AWS ParallelCluster

Lab Instructions

- Lab 5 Instructions:
  - https://ee3801.github.io/Lab5/instruction.html
- Submit to Canvas (Lab 5->Lab 5A & Lab 5->Lab 5B)
- Submit in PDF format
- Name the files Lab5A_YourName.pdf and Lab5B_YourName.pdf
- Part A due on Monday (Oct 16) 2 pm
- Part B due on Wednesday (Oct 18) 9 pm

Questions?