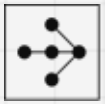


RAY
SUMMIT
2024

Accelerating Campus Computational Science with Ray

Jack Brassil, Princeton University
Associate Dean for Research
Senior Research Scholar, Dept. of Computer Science

PRINCETON
UNIVERSITY

Goal – Support mix of HPC and AI/ML workloads across campus platforms

- Diverse population of computational scientists needing Python horizontal and vertical scaling
- AI/ML research and training needs growing rapidly in Humanities and Social Sciences
- Some degree of cluster hardware divergence today (e.g., dedicated campus system for LLM research)
- Investigating RAY to provide a single framework to support on the widest range of target platforms

National Science Foundation Award OAC-2429485 (PI: Brassil)
Project Title: CC* Integration-Small: Unifying and Accelerating Campus Computational Science with Ray

Goal – Support mix of HPC and AI/ML workloads across campus platforms (**and beyond**)

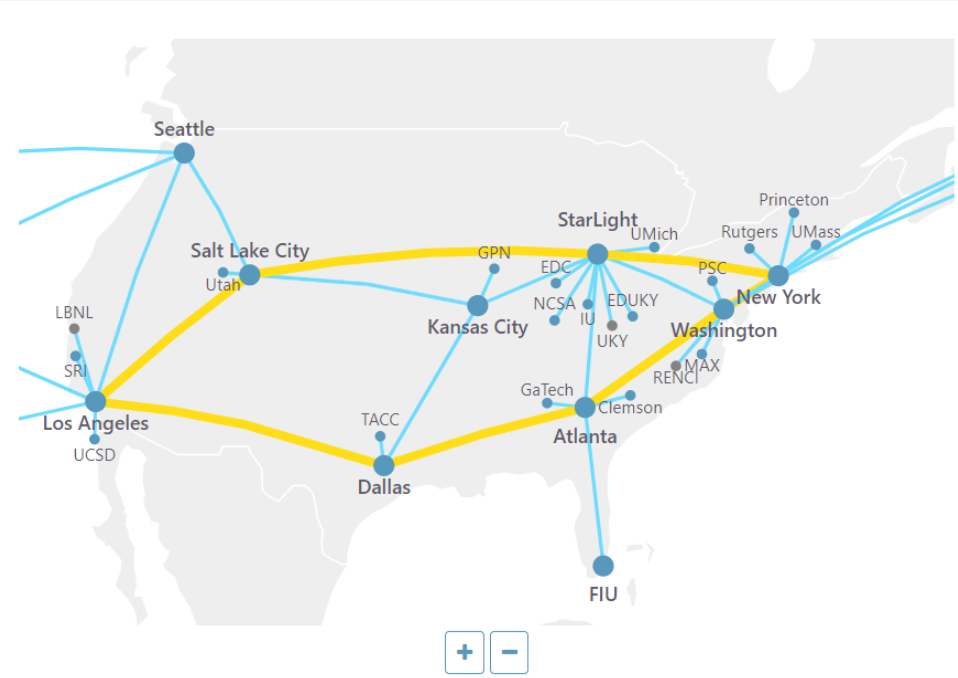
1. Public compute clouds
2. DoE Leadership Class Systems
3. NSF Shared SuperComputing Centers
4. NSF Computer & Networking Testbeds (FABRIC)
5. Regional AI Hub
6. Conventional campus HPC clusters
7. Dedicated AI GPU clusters (Princeton Language & Intelligence)
8. Desktops & departmental clusters
9. IoT (e.g., Jetson, Raspberry Pi)

Use case: Evolutionary Biology

Mpala Research Center, Nanyuki, Kenya

- RL-based camera trap applications trigger local environmental actuation (ruggedized Raspberry PI)
- OSU Imageomics backend: ML-based extraction of existing and new biological traits from images of organisms (cluster)
- Goal – simple software platforms for end-to-end research, OTA deployment

Use case: NSF FABRIC Research Infrastructure



TACC (TACC) Active

Resource	available / total
Cores	388/640
Disk (GB)	103703/107463
RAM (GB)	1598/2390
GPU	4/10
NVME	16/16
SmartNIC	5/6
SharedNIC	616/635
FPGA	1/1

- Testbed connecting major instruments, compute clouds, and universities
- Permits collaborative sharing of on- and off-campus computing systems
- whatisfabric.net

Ray over FABRIC

Step 1 - FABRIC Preliminaries. Build a global network topology

```
# Preliminaries (e.g., import FABlib libraries)

from ipaddress import ip_address,
IPv4Address, IPv6Address, IPv4Network, IPv6Network
from fabrictestbed_extensions.fablib.fablib
import FablibManager as fablib_manager
fablib = fablib_manager()

#Create FABRIC slice
slice_name = ’4node-2site’
node1_name = ’prin1’; node2_name = ’prin2’;
node3_name = ’ucsd3’; node4_name = ’ucsd4’;
net_name=’net1’
slice = fablib.new_slice(name=slice_name)

# Add network
net1 = slice.add_l2network(name=net_name,
subnet=IPv4Network("192.168.100.0/24")
```

Ray over FABRIC

Step 2 – Deploy VMs across FABRIC sites. Deploy contributed bare metal nodes across sites.

```
# Node1
node1 = slice.add_node(name=node1_name,
  site='PRIN', cores=4, ram=16, disk=32,
  image='default_ubuntu_22')
iface1 = node1.add_component(model='NIC_Basic',
  name='nic1').get_interfaces()[0]
iface1.set_mode('auto')
net1.add_interface(iface1)
.
.
.
# Node4
node4 = slice.add_node(name=node4_name,
site='UCSD', cores=4, ram=16, disk=32,
image='default_ubuntu_22')
iface4 = node4.add_component(model='NIC_Basic',
name='nic1').get_interfaces()[0]
iface4.set_mode('auto')
net1.add_interface(iface4)

# Create topology

slice.submit()
```

Ray over FABRIC

Step 3 – Deploy executables & tools

```
# Upload configuration files

result1 = node1.upload_file
 ('config_script.sh', 'config_script.sh')
        .
        .
        .
result4 = node4.upload_file
 ('config_script.sh', 'config_script.sh')

# Additional script arguments

script_args="net-tools wireshark"

# Run configurations

stdout, stderr = node1.execute(f'chmod +x
 config_script.sh && ./config_script.sh
 {script_args} >> config1.log')
        .
        .
        .
stdout, stderr = node4.execute(f'chmod +x
 config_script.sh && ./config_script.sh
 {script_args} >> config4.log')
```

Ray over FABRIC

Step 4 – Run Ray benchmarks

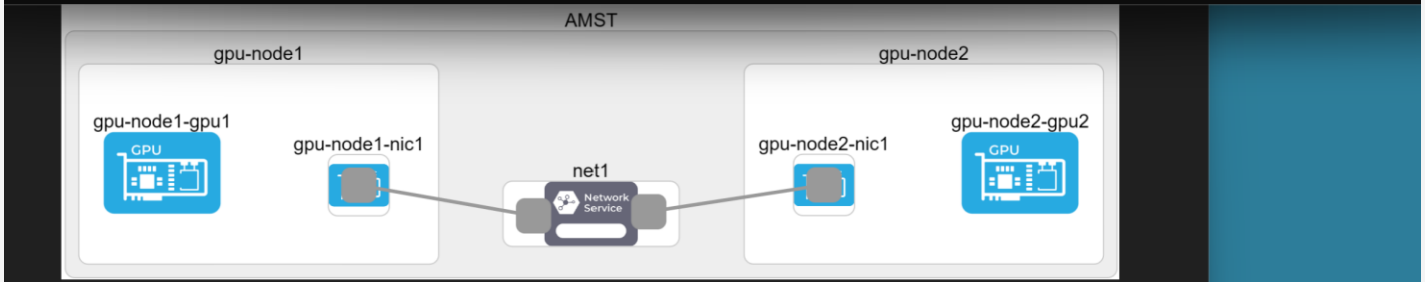
```
------------config_script.sh------------
#!/bin/bash
args=$@
sudo apt install -y $args

pip install -U "ray[default]"

if (worker_node):
  ray start
  --address='192.168.100.3:6379'
  --node-ip-address=192.168.100.3
ray microbenchmark
else:
  ray start --head
  --node-ip-address=192.168.100.3
  --metrics-export-port=8080
  --dashboard-host=192.168.100.3
```

ubuntu@gpu-node2: ~

```
Training saved a checkpoint for iteration 10 at: (local)/tmp/ray_results/ptl-mnist-example/TorchTrainer_d2fa8_00000_0_2024-10
-02_14-37-17/checkpoint_000009
(RayTrainWorker pid=55856) Checkpoint successfully created at: Checkpoint(filesystem=local, path=/tmp/ray_results/ptl-mnist-e
xample/TorchTrainer_d2fa8_00000_0_2024-10-02_14-37-17/checkpoint_000009)
(RayTrainWorker pid=55856) `Trainer.fit` stopped: `max_epochs=10` reached.
Epoch 9: 100%|██████████| 430/430 [00:05<00:00, 76.99it/s, v_num=0]
(RayTrainWorker pid=55856) LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Testing: |          | 0/? [00:00<?, ?it/s]
Testing DataLoader 0:   9%|▉         | 7/79 [00:00<00:00, 87.22it/s]
Testing DataLoader 0:  24%|██▍       | 19/79 [00:00<00:00, 105.50it/s]
Testing DataLoader 0:  37%|███▋      | 29/79 [00:00<00:00, 105.85it/s]
Testing DataLoader 0:  51%|█████     | 40/79 [00:00<00:00, 103.87it/s]
Testing DataLoader 0:  63%|██████▎   | 50/79 [00:00<00:00, 104.34it/s]
Testing DataLoader 0:  76%|███████▌  | 60/79 [00:00<00:00, 103.91it/s]
Testing DataLoader 0:  90%|████████▉ | 71/79 [00:00<00:00, 103.36it/s]
Testing DataLoader 0: 100%|██████████| 79/79 [00:00<00:00, 108.79it/s]
(RayTrainWorker pid=55856)
(RayTrainWorker pid=55856)
(RayTrainWorker pid=55856)
(RayTrainWorker pid=55856)
(RayTrainWorker pid=55856)
```

Test metric	DataLoader 0
test_accuracy	0.9775000214576721

```
Training completed after 10 iterations at 2024-10-02 14:38:32. Total running time: 1min 14s
2024-10-02 14:38:32,121 INFO tune.py:1009 -- Wrote the latest version of all result files and experiment state to '/tmp/ray_r
```



Example: Using Ray on FABRIC testbed

MNIST training run on 2 A30 GPU nodes at FABRIC Amsterdam site

Characterizing RAY performance across multiple, diverse academic platforms.

Understanding how RAY can streamline teaching distributed systems to a growing cohort of students spanning multiple disciplines.

Seeking to build and strengthen a RAY community across university Research Computing organizations!

Thanks!