Cloud Native Deployments of Bare-Metal High-Performance AI Workflows

ATLAS

George | Hao | Jing | Shawn | Shubham

MENTOR Chris | Mike | Ravi

Introduction:

- Current bare-metal machine in MIT (Satori):
  - 64 1TB memory Power 9 nodes.
  - Each node hosts four NVidia V100 32GB memory GPU cards.
  - 100Gb/s Infiniband network connects the cluster nodes.
- OpenShift/Kubernetes migration
- AI workflow between bare-metal and OpenShift
- Potential benefits in OpenShift

Vision and Goals:

- Survey existing MIT bare-metal workloads & containerize one of them.
- Monitor & compare OpenShift workflows and bare-metal workflows.
- Generate reports with pros/cons of migrating bare-metal workflow to OpenShift environment.

Users/Personas:

This system will target the following users:

- AI researchers looking to deploy high-performance workflows:

  bare-metal environment → cloud native environment.

- ML/AI engineers looking to deploy *processing-hungry* system to cloud.
- Users seeking data privacy in the cloud (provided by OpenShift).
- Users needing Singularity/virtualization systems to containerize workflows in HPC (high-performance computing) clusters.

Users/Personas:

- A quintessential example of a user could be the MIT-IBM Watson AI laboratory looking to scale their workflows into the cloud in a discrete fashion.
- Average users/hobbyists looking to deploy non-intensive computational processes to the cloud.

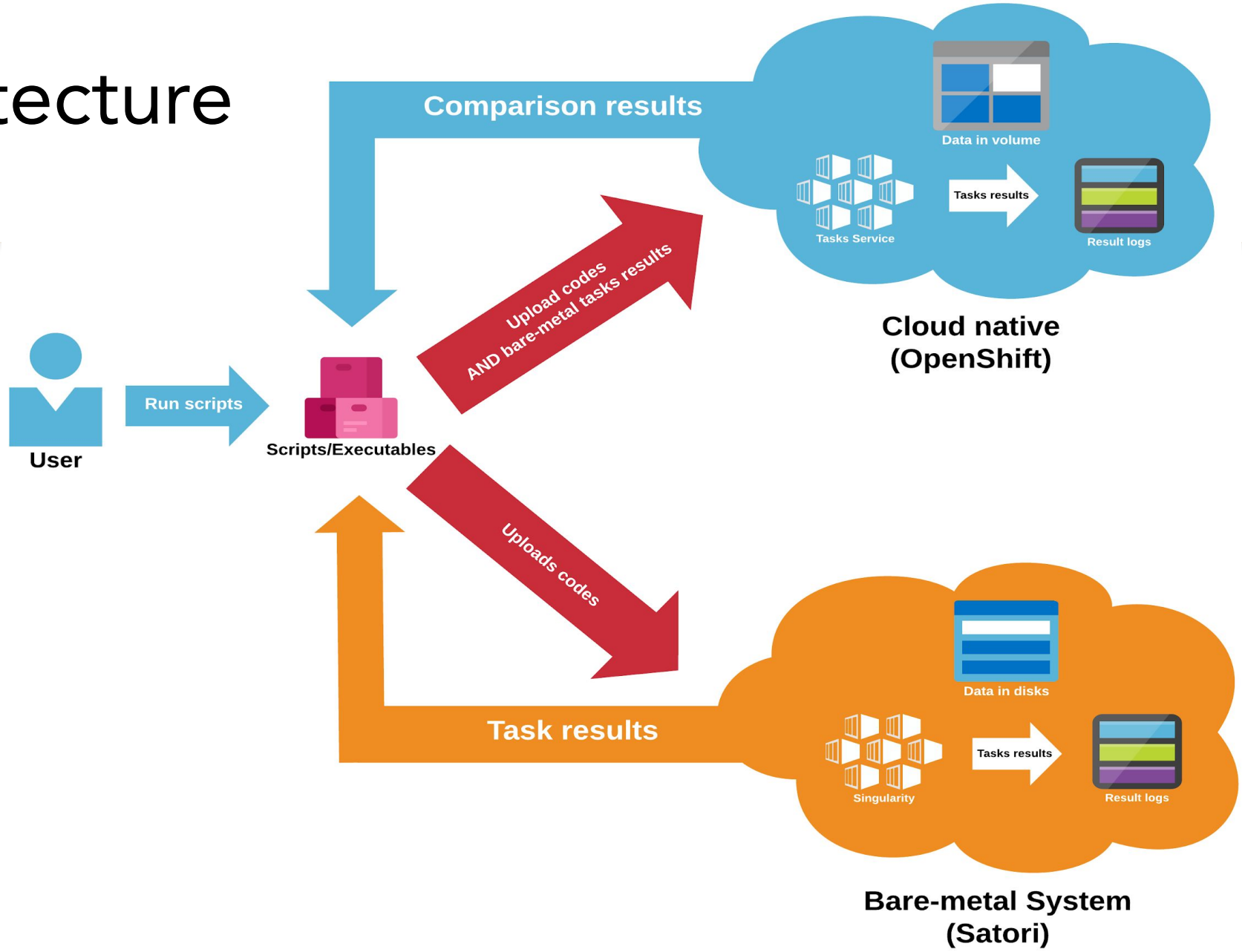
This system will NOT target the following users:

- Users with complex requirements who might require additional interface/systemic modification.

Scope and Features:

- Create documentation & scripts to allow users to containerize existing High Performance (AI) workflows
- Generate charting comparing performance metrics (elasticity, economics, etc.) between bare-metal & OpenShift environments.
- Generate display (of suggestions) for ‘under-utilized’ nodes in OpenShift (*running backfill workloads*?).
- Deploy researcher workflows/code with ease from bare-metal environment to OpenShift/Kubernetes.

Architecture



Acceptance Criteria:

- Interface: containerize & deploy a specific AI workflow.
- Generate comparisons (elasticity, performance, economics, etc.) between project being run in OpenShift *vs* Satori.
- Directing resources to under-utilized nodes (or displaying there are such instances).
- Extending to wider class of projects by circumventing problem of workflows being tied to a current system.

BigGAN - MIT HPC

- Generative Adversarial Network
- Novel/creative data
- Realistic: Inception Score (IS) of 166.5

Thus far...Satori

- BigGAN (138Gb training data)
- Submitting jobs to queue
- WMLCE Environment -- PowerAI

```
(wmlce-1.6.2) [sorescu@service0001 BigGAN-PyTorch]$ bsub < jobs/biggan_deep128_imagenet.lsf
Job <24334> is submitted to queue <normal>.
(wmlce-1.6.2) [sorescu@service0001 BigGAN-PyTorch]$ ls
animal_hash.py                datasets.py       layers        notebooks    SATORI.md
BigGANdeep.py                 eval_models.py    layers.py     profiler.py  scripts
BigGAN.py                     get_power         LICENSE       __pycache__  set.sh
```

Satori Logs

- Iterative training plots
- Tracking in Jupyter Notebooks

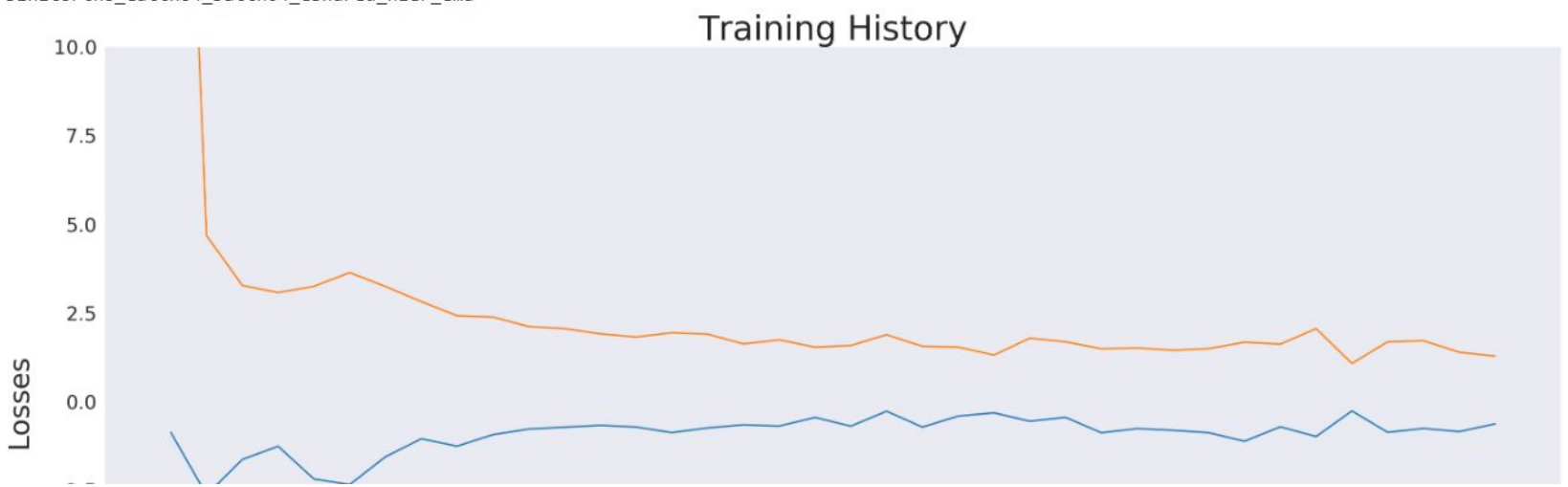
Plot Generator and Discriminator loss trajectories for a particular experiment

```
log_dirs =  [os.path.join('logs', f) for f in os.listdir('logs') if os.path.isdir(os.path.join('logs', f))]
for i, lg in enumerate(log_dirs):
    print(i, lg)

log_num = 0
logs = utils.load_logs(log_dirs[log_num])

utils.plot_loss_logs(logs['loss'], smoothing=0.000, figsize=(24, 12))
```

```
0 logs/BigGAN_ImageNet_128_biggan_deep_seed0_Gch128_Dch128_Gd2_Dd2_bs256_nDs2_Glr5.0e-05_Dlr2.0e-04_Gnlinplace_relu_Dnlinplace_relu_Ginitortho_
Dinitortho_Gattn64_Dattn64_cr_Gshared_hier_ema
1 logs/BigGAN_ImageNet_128_biggan_deep_seed0_Gch128_Dch128_Gd2_Dd2_bs256_nDs2_Glr5.0e-05_Dlr2.0e-04_Gnlinplace_relu_Dnlinplace_relu_Ginitortho_
Dinitortho_Gattn64_Dattn64_Gshared_hier_ema
```



Satori Logs

- Metadata logs
- Track performance: bare-metal *vs* cloud

```
datetime: 2020-02-10 10:16:34.361624
config: {'dataset': 'ImageNet', 'resolution': 128, 'augment': False, 'num_workers': 2, 'pi
'use_multiepoch_sampler': False, 'world_size': 32, 'rank': 0, 'dist_url': 'file:///nobacku
'dist_backend': 'nccl', 'multiprocessing_distributed': True, 'gpu': 0, 'dataset_type': 'Im
'D_param': 'SN', 'G_ch': 128, 'D_ch': 128, 'G_depth': 2, 'D_depth': 2, 'D_wide': True, 'G_
'hier': True, 'cross_replica': False, 'mybn': False, 'G_nl': 'inplace_relu', 'D_nl': 'inpl
'bn', 'seed': 0, 'G_init': 'ortho', 'D_init': 'ortho', 'skip_init': False, 'G_lr': 5e-05,
'D_B2': 0.999, 'batch_size': 64, 'G_batch_size': 0, 'num_G_accumulations': 1, 'num_D_steps
'num_epochs': 500, 'parallel': False, 'G_fp16': False, 'D_fp16': False, 'D_mixed_precision
False, 'num_standing_accumulations': 16, 'use_torch_FID': False, 'pretrained': 'imagenet',
2, 'num_best_copies': 5, 'which_best': 'IS', 'no_fid': False, 'test_every': 2000, 'num_inc
'data_root': 'data', 'weights_root': 'weights', 'logs_root': 'logs', 'samples_root': 'samp
'name_suffix': '', 'experiment_name': '', 'config_from_name': False, 'ema': True, 'ema_dec
1e-06, 'BN_eps': 1e-05, 'SN_eps': 1e-06, 'num_G_SVs': 1, 'num_D_SVs': 1, 'num_G_SV_itrs':
'toggle_grads': True, 'which_train_fn': 'GAN', 'load_weights': '', 'resume': False, 'logst
False, 'sv_log_interval': 10, 'debug': False, 'n_classes': 1000, 'G_activation': ReLU(inpl
True}
```

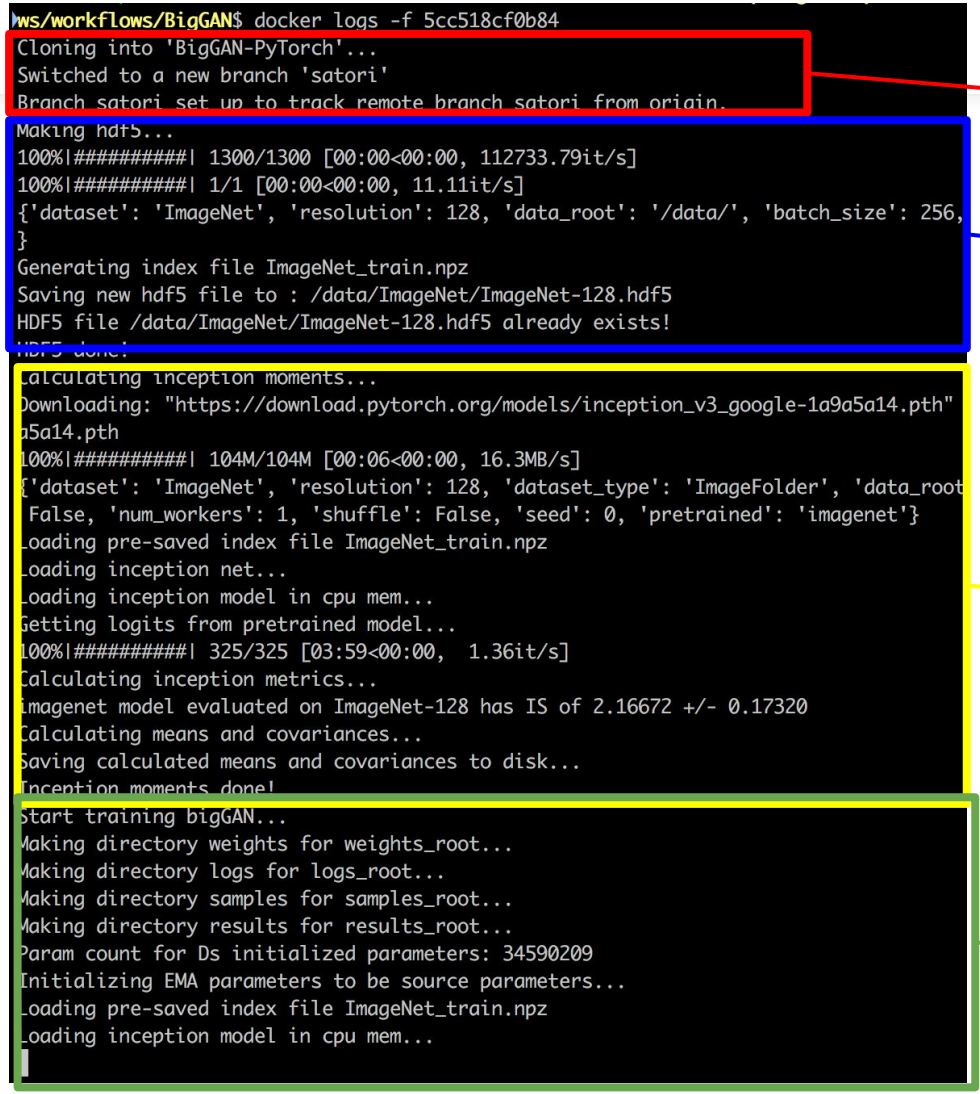
Containerize Big-GAN workflow

1. Write bash scripts to automate the entire workflow (data preprocess -> pre-fetch eval metric scores -> model training)
2. Build a docker image that runs the workflow in a container
3. Run & monitor the container (redirect std out to log file)

```
GAN$ docker build -t big-gan .
Sending build context to Docker daemon   7.68kB
Step 1/9 : FROM centos:centos7
 ---> 5e35e350aded
Step 2/9 : RUN yum -y update; yum clean all
 ---> Using cache
 ---> 52a0c208bf19
Step 3/9 : RUN yum -y install epel-release gcc openssl-devel bzip2-de
vel git python37 python3-pip vim; yum clean all
 ---> Using cache
 ---> b09d7e2135cd
Step 4/9 : ADD ./requirements.txt /python/requirements.txt
 ---> Using cache
 ---> e6f316b93acc
Step 5/9 : RUN pip3 install --upgrade pip && pip install -r /python/r
equirements.txt
 ---> Using cache
 ---> 220ce5b216b7
Step 6/9 : ADD ./ /BigGAN
 ---> Using cache
 ---> 686eafbfc818
Step 7/9 : WORKDIR /BigGAN
 ---> Using cache
 ---> ced3018731d2
Step 8/9 : CMD cd /BigGAN
 ---> Using cache
 ---> 182f9585b109
Step 9/9 : CMD /bin/sh run.sh
 ---> Using cache
 ---> 798971a0cd44
Successfully built 798971a0cd44
Successfully tagged big-gan:latest
```

```
Shawn@dhcp-wifi-8021x-155-41-73-214:~/Documents/BU/courses/2020_spring/CComp$ docker images | head -n 2
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
big-gan             latest              6cd9e755c717        5 minutes ago       2.89GB
```

Containerize Big-GAN workflow - stdout logs



Cloning code from github repo

Data preprocess & compress

Pre-fetch eval metric

Model training

Demo

1. git clone https://github.com/BU-CLOUD-S20/Cloud-native-deployments-of-bare-metal-high-performance-AI-workflows.git
2. cd Cloud-native-deployments-of-bare-metal-high-performance-AI-workflows/workflows/BigGAN/
3. docker build -t big-gan .
4. docker run -d --name cs528_demo -v /Users/Shawn/Documents/BU/courses/2020_spring/CComp/Cloud-native-deployments-of-bare-metal-high-performance-AI-workflows/data/:/data/ big-gan:latest
5. docker exec -it 5cc518cf0b84 /bin/bash

Optional Features:

- Automatic deploy: interface or containers to automatically execute experimental codes in two different environments.
- Generalized: orientation towards high-performance AI workflows, but have the capability to deploy wide range of projects.
- Operate with ease across multiple deployments (MIT HPC, MIT-IBM Watson lab, etc.).

Thank you!

Q&A

Whoa, how did you read this?