Boost ML on Ray

Enabling Heterogeneous AI Accelerators Seamlessly

Tiejun Chen
VMware, OCTO

6/18/2023

Agenda

Empower AI everywhere

- Problem area on ML/AI
- ML on Ray
- How did we get to boost ML on Ray ?
- Demo
- What’s likely next ?

Problem area

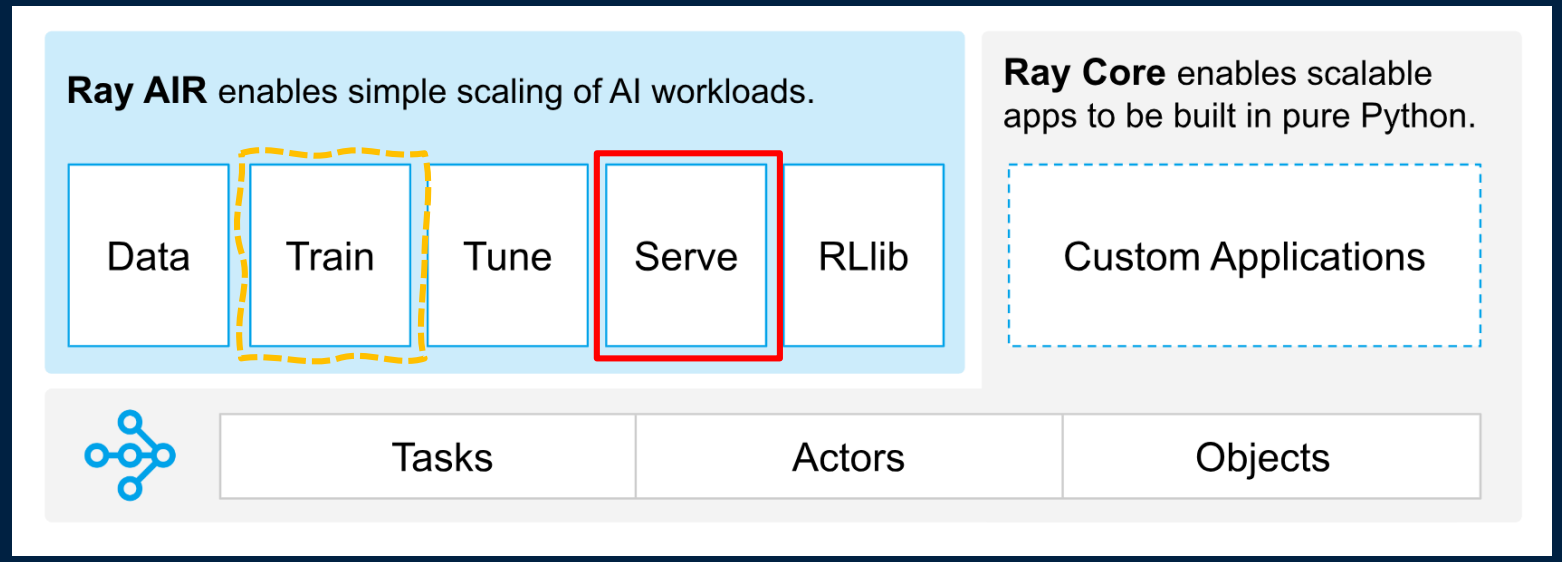
Towards modern AI application centric platform

- Heterogeneous AI HW accelerators
- Various upstream ML frameworks
- Hard to exploit the best performance
- No such a modern AI platform with cloud native principle

ML on Ray

Brief

❖ *Flexible distributed Python for machine learning*



https://github.com/ray-project/ray

- Ray Serve
  - Be framework-agnostic
  - Doesn't perform any model-specific optimizations

How did we get to boost ML on Ray ?

Project Yellowstone - Goal

- Build end-to-end ML service on Kubernetes from cloud to edge
  - ❑ Enable CRD based accelerators for ML serving
  - ❑ Boost ML by transparent backend acceleration

How did we get to boost ML on Ray ?

Project Yellowstone - Enable CRD based local accelerators for ML serving

- Node Feature Discovery
- Device plugins
- Node selector
- Kubernetes scheduler

How did we get to boost ML on Ray ?

Graph compliers

- What ?

ML frameworks

Graph complier

Lib/drivers

- Graph compilers
  - ❑ Apache TVM
  - ❑ Nvidia TensorRT
  - ❑ Intel OpenVINO
  - ❑ AMD ROCM
  - ❑ Xilinx vitis AI

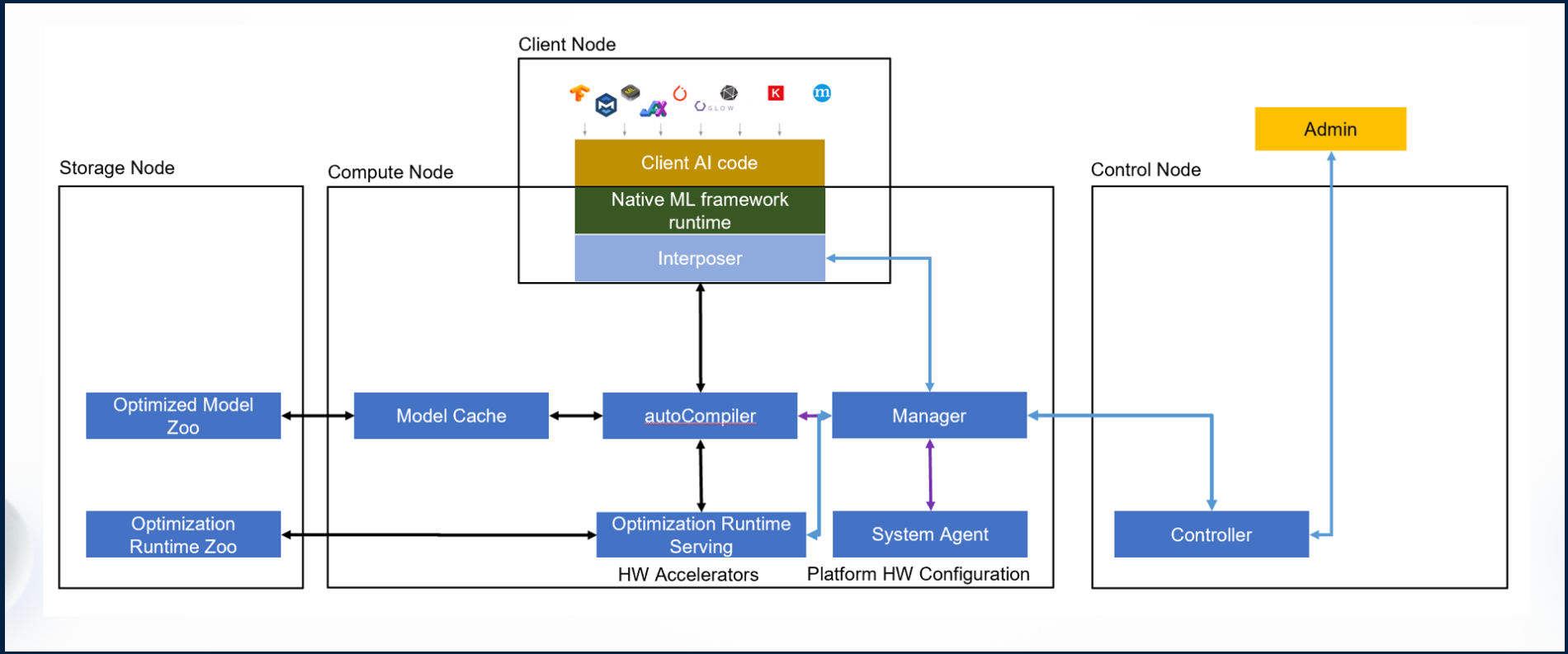
How did we get to boost ML on Ray ?

Project Yellowstone - Overview

- Target
  - Boost ML/AI by enabling ML upstream frameworks seamlessly with graph compilers
- Design
  - Build ML Boost Serving System
    - Backend
    - Automated
    - Unified server architecture
- How
  - Interpose ML framework API
  - Built-in graph compilers processing – Auto {detecting, compiling, scheduling, inferencing, etc}

How did we get to boost ML on Ray ?

Project Yellowstone - Architect



How did we get to boost ML on Ray ?

Project Yellowstone – Seamless interposer

- Runtime interposer
  - ❑ Target to key APIs
  - ❑ Mapping between ML Frameworks APIs and Backend APIs
- Example
  - ❑ Tensorflow - Python
    - tensorflow.keras.models.load_model = booster_load_model
    - tensorflow.keras.models.Model.predict = booster_predict
  - ❑ Tensorflow Serving - C++
    - predict --> session->Run()
    - Hijack the process at runtime to call booster_predict

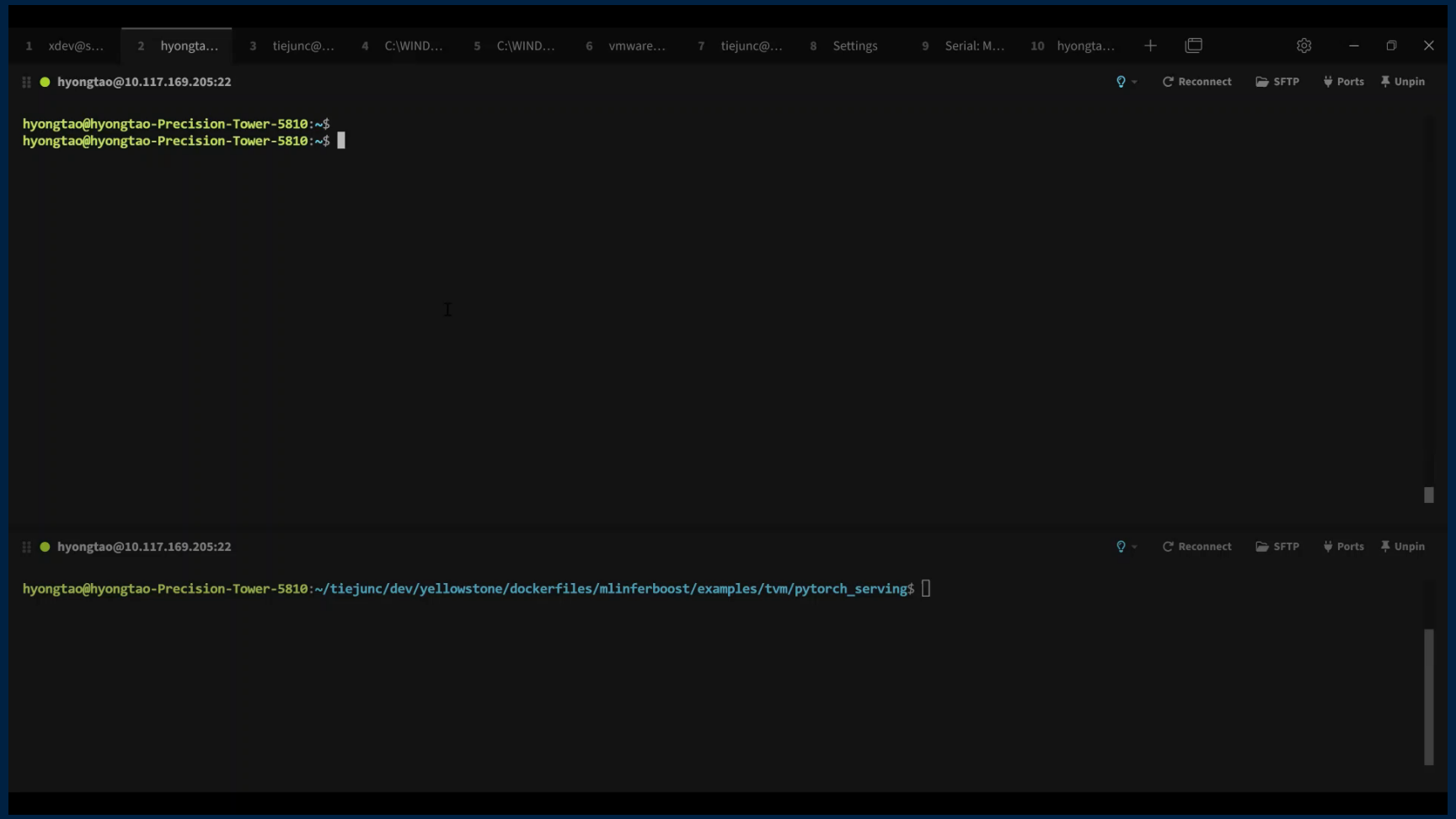
How did we get to boost ML on Ray ?

Project Yellowstone – Demo

- TorchServe on GPU with backend TVM acceleration
- Tensorflow Serving on GPU with backend TVM acceleration
- Pytorch on GPU with backend TVM acceleration on Ray

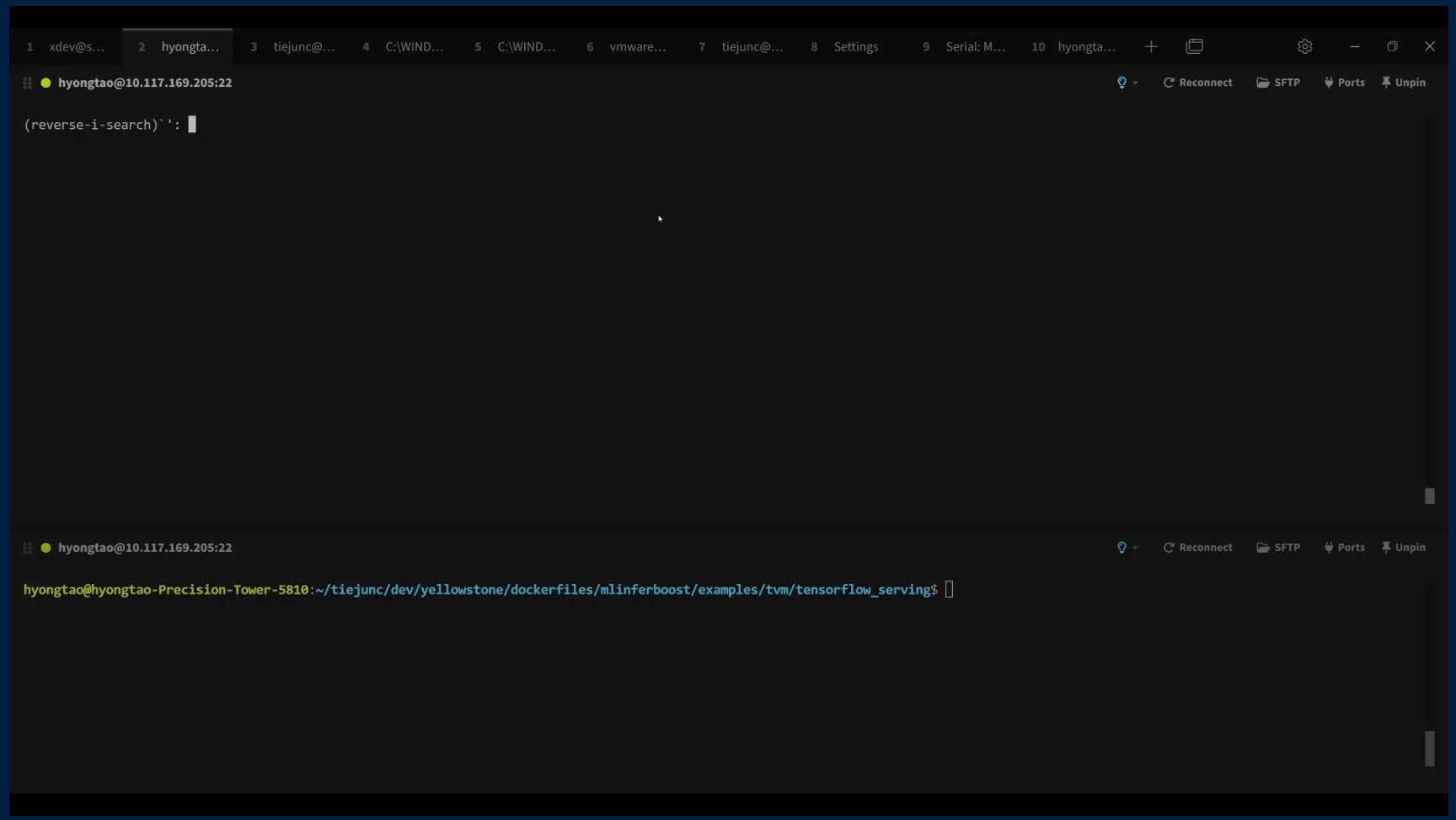
Demo I

TorchServe on GPU with backend TVM acceleration



Demo II

Tensorflow Serving on GPU with backend TVM acceleration



Demo III

Pytorch on GPU with backend TVM acceleration on Ray

1 tiejunc... 2 root@7e... 3 root@7e... 4 tkgmlm... 5 ecl@bas... 6 ecl@ml-... 7 hyongta... 8 hyongta... 9 crx@crx... 10 tiejunc...

tiejunc@10.221.33.199:22 Reconnect SFTP Ports Unpin

```
root@7e4a35d5b0cf:/mlinferboost# python3 compiler.py -f pytorch -b tvm
ML Framework: pytorch is accelerated by the backend tvm.
Waiting for Pytorch ...
 * Serving Flask app 'compiler'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a prod
uction WSGI server instead.
 * Running on http://127.0.0.1:9000
Press CTRL+C to quit
[ INFO ] target:  llvm -mcpu=core-avx2
[ INFO ] target_host:  llvm -mcpu=core-avx2
[ INFO ] Auto Detect Devices -> Done
[ INFO ] We'll hook /usr/local/lib/python3.8/dist-packages/torch/__init__.py.
[ INFO ] Hook pytorch -> Done
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
The interposer is running...
```

tiejunc@10.221.33.199:22 Reconnect SFTP Ports Unpin

```
root@7e4a35d5b0cf:/# cat infer.ray.py
from ray import serve

from io import BytesIO
from PIL import Image
import requests

import torch
from torchvision import transforms
from torchvision.models import resnet18

@serve.deployment(route_prefix="/image_predict")
class ImageModel:
    def __init__(self):
        self.model = resnet18(pretrained=True).eval()
        self.preprocessor = transforms.Compose([
            transforms.Resize(224),
            transforms.CenterCrop(224),
            transforms.ToTensor(),
            transforms.Lambda(lambda t: t[:3, ...]),  # remove alpha channel
            transforms.Normalize(
                mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
        ])

    async def __call__(self, starlette_request):
        image_payload_bytes = await starlette_request.body()
        pil_image = Image.open(BytesIO(image_payload_bytes))
        print("[1/3] Parsed image data: {}".format(pil_image))

        pil_images = [pil_image]  # Our current batch size is one
        input_tensor = torch.cat(
            [self.preprocessor(i).unsqueeze(0) for i in pil_images])
        print("[2/3] Images transformed, tensor shape {}".format(
            input_tensor.shape))

        with torch.no_grad():
            output_tensor = self.model(input_tensor)
        print("[3/3] Inference done!")
        return {"class_index": int(torch.argmax(output_tensor[0]))}
```

What's "here" now ?

Project Yellowstone

- ML frameworks
  - Tensorflow/Pytorch/ONNX, Tensorflow Serving, TorchServe, KServe, etc
  - Ray
- Backend acceleration technologies
  - Apache TVM
  - Intel OpenVINO
  - Nvidia TensorRT
  - Xilinx vitis AI
- AI HW accelerators
  - {Nvidia, AMD, Intel} GPU, Xilinx FPGA
  - CPU

What’s likely next ?

Project Yellowstone++

- Moving only ML Inference to ML {Training, Inference}
- Moving towards multi-AI cloud

Empower AI everywhere

Thank you !

Q & A

Tiejun Chen <tiejunc@vmware.com>