



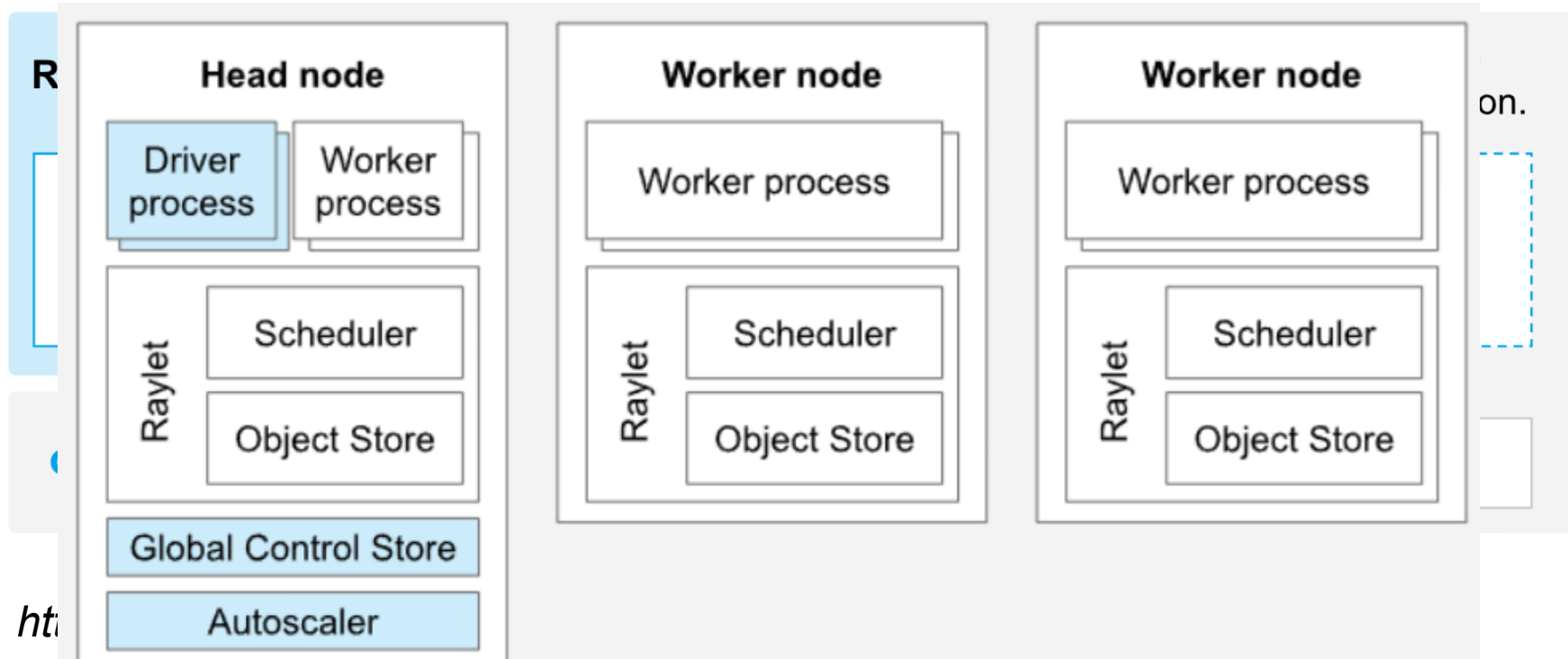
# Enable RISC-V Accelerated Ray for AI Workloads

Tiejun Chen  
RISC-V International Ambassador

- Ray Overview
- Enable Ray on RISC-V
  - demos
- Ray extensions
  - Accelerate Ray
  - Remote ML/AI
- What's likely next

# Ray Overview I

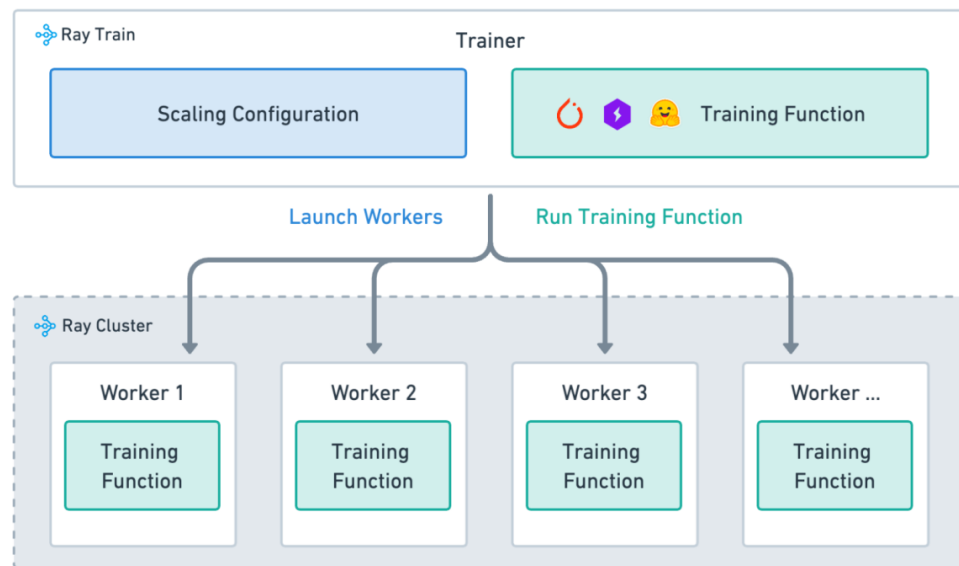
## Brief



# Ray Overview II

## Ray Train

- A scalable machine learning library for distributed training and fine-tuning
- Main components
  - Training function
  - Worker
  - Scaling configuration
  - Trainer



❖ <https://docs.ray.io/en/master/train/overview.html>

## Rat Serve

- A scalable model serving library for building online inference APIs
  - Be framework-agnostic
    - Tensorflow, Pytorch ,etc

```
@serve.deployment
class TFMnistModel:
    def __init__(self, model_path: str):
        import tensorflow as tf

        self.model_path = model_path
        self.model = tf.keras.models.load_model(model_path)

    async def __call__(self, starlette_request: Request) -> Dict:
        # Step 1: transform HTTP request -> tensorflow input
        # Here we define the request schema to be a json array.
        input_array = np.array((await starlette_request.json())["array"])
        reshaped_array = input_array.reshape((1, 28, 28))

        # Step 2: tensorflow input -> tensorflow output
        prediction = self.model(reshaped_array)

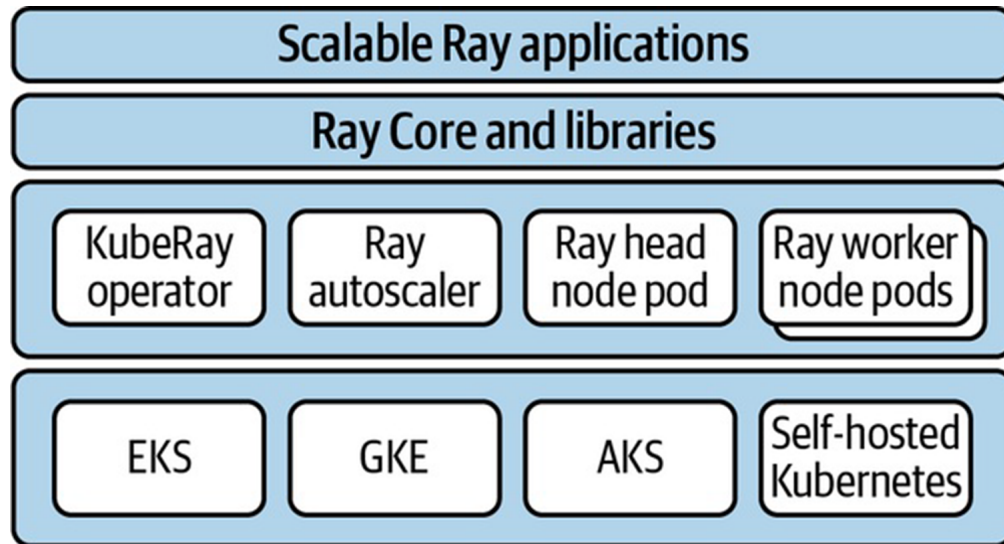
        # Step 3: tensorflow output -> web output
        return {"prediction": prediction.numpy().tolist(), "file": self.model_path}
```

*Doesn't perform any model-specific optimizations*

# Ray Overview

## KubeRay – Managing Ray clusters on Kubernetes

- RayService
- RayJob
- RayCluster



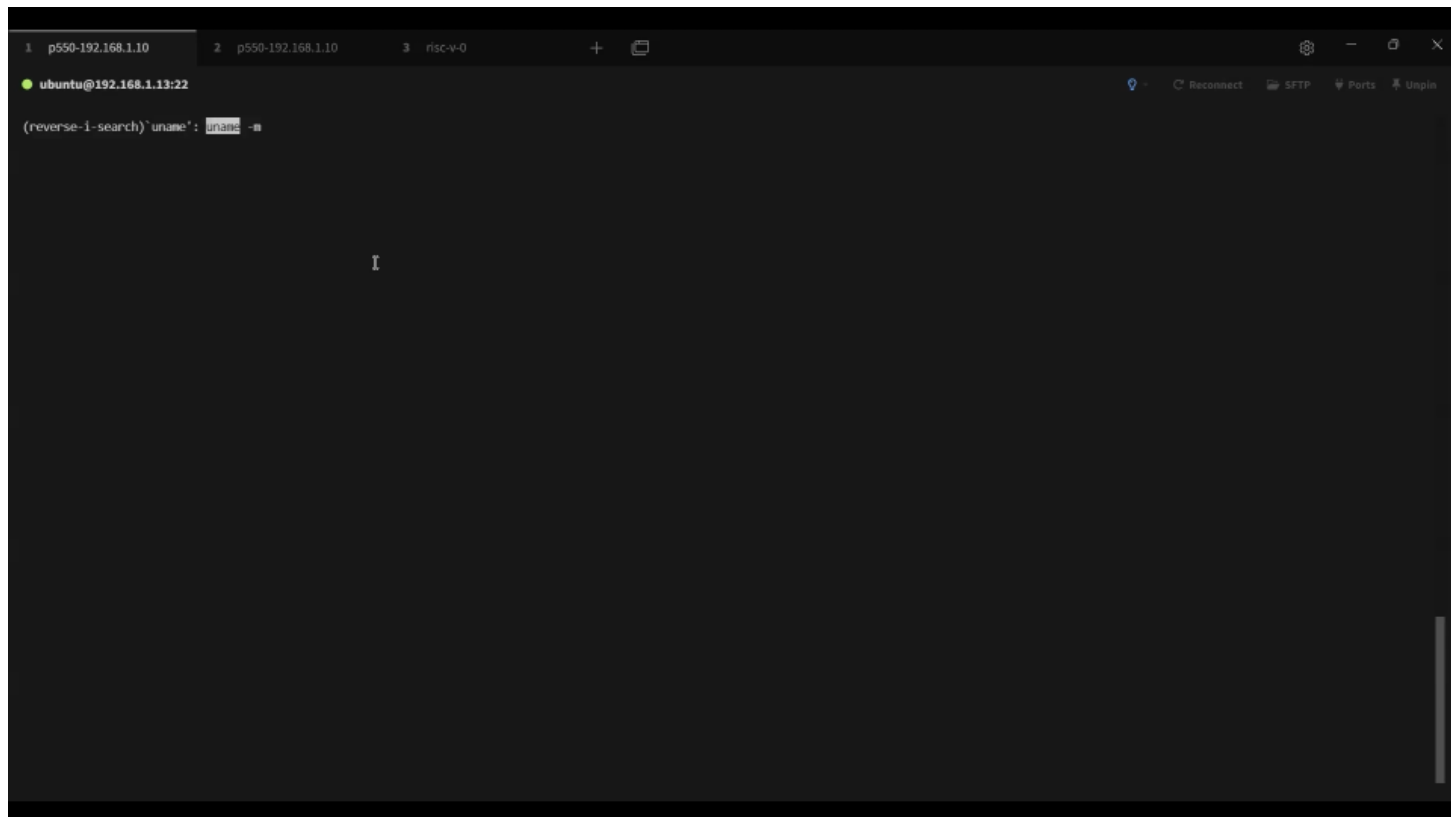
# Enable Ray to RISC-V

## Basic requirements | Challenges

- *ML frameworks – Pytorch, Tensorflow, etc*
- *Bazel == 6.5.0*
  - abseil-cpp (RDCYCLE )
- *Golang*
- *Docker*
- *50+packages*
  - pyarrow
  - relocate-perl

# Enable Ray to RISC-V – Demo I

## Ray Train on RISC-V



A terminal window with three tabs: 'p550-192.168.1.10', 'p550-192.168.1.10', and 'riscv-0'. The active tab is 'riscv-0'. The terminal shows a shell prompt 'ubuntu@192.168.1.13:22' and a command '(reverse-i-search)`uname': uname -m'. The cursor is positioned at the end of the command.

```
1 p550-192.168.1.10 2 p550-192.168.1.10 3 riscv-0
● ubuntu@192.168.1.13:22
(reverse-i-search)`uname': uname -m
```

# Enable Ray to RISC-V – Demo II

## Ray Serve on RISC-V

```
1 xdev-laptop 2 p550-192.168.1.10 3 p550-192.168.1.10 4 riscv-0
ubuntu@192.168.1.13:22
root@c187fd2da4a4:/# uname -m
riscv64
root@c187fd2da4a4:/# python
Python 3.12.3 (main, Jan 17 2025, 18:03:48) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import ray
>>> print(ray.__version__)
3.0.0.dev0
>>>
root@c187fd2da4a4:/# cat tutorial_pytorch.py
from ray import serve

from io import BytesIO
from PIL import Image
from starlette.requests import Request
from typing import Dict

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

@serve.deployment
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: Request) -> Dict:
        image_payload_bytes = await starlette_request.body()
        pil_image = Image.open(BytesIO(image_payload_bytes))

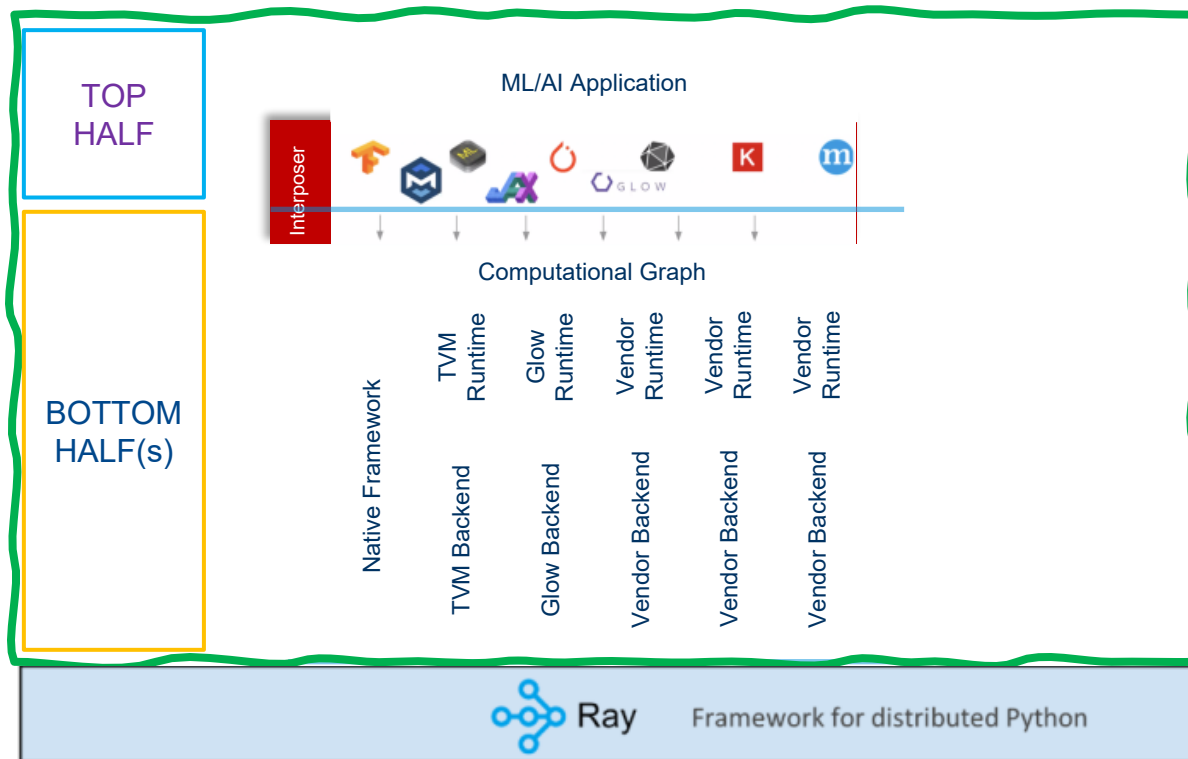
root@c187fd2da4a4:/# cat infer.py
import requests

ray_logo_bytes = requests.get(
    "https://raw.githubusercontent.com/ray-project/"
    "ray/master/doc/source/images/ray_header_logo.png"
).content

resp = requests.post("http://127.0.0.1:8000/", data=ray_logo_bytes)
print(resp.json())
root@c187fd2da4a4:/# python infer.py []
```

# Ray extensions – Accelerate Ray

## Leverage graph compilers



ML Frameworks



Computational Graph



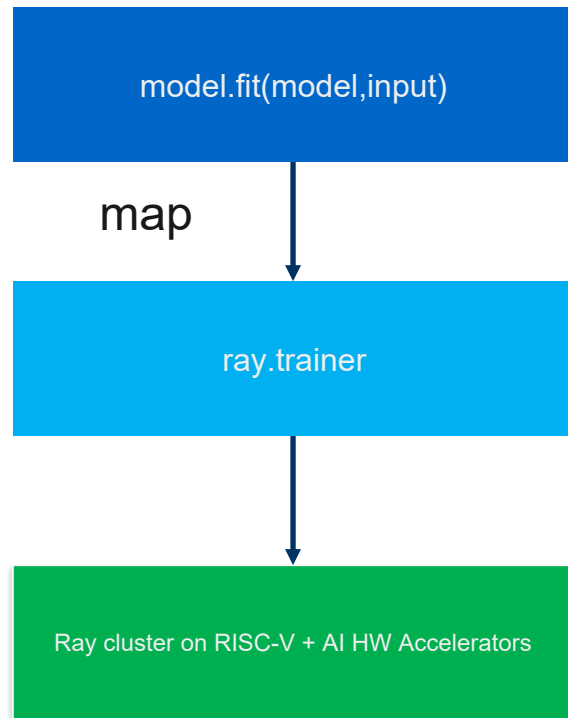
Low-level Libs/Device Drivers

# Ray extensions – Remote ML/AI I

## Empower native ML frameworks

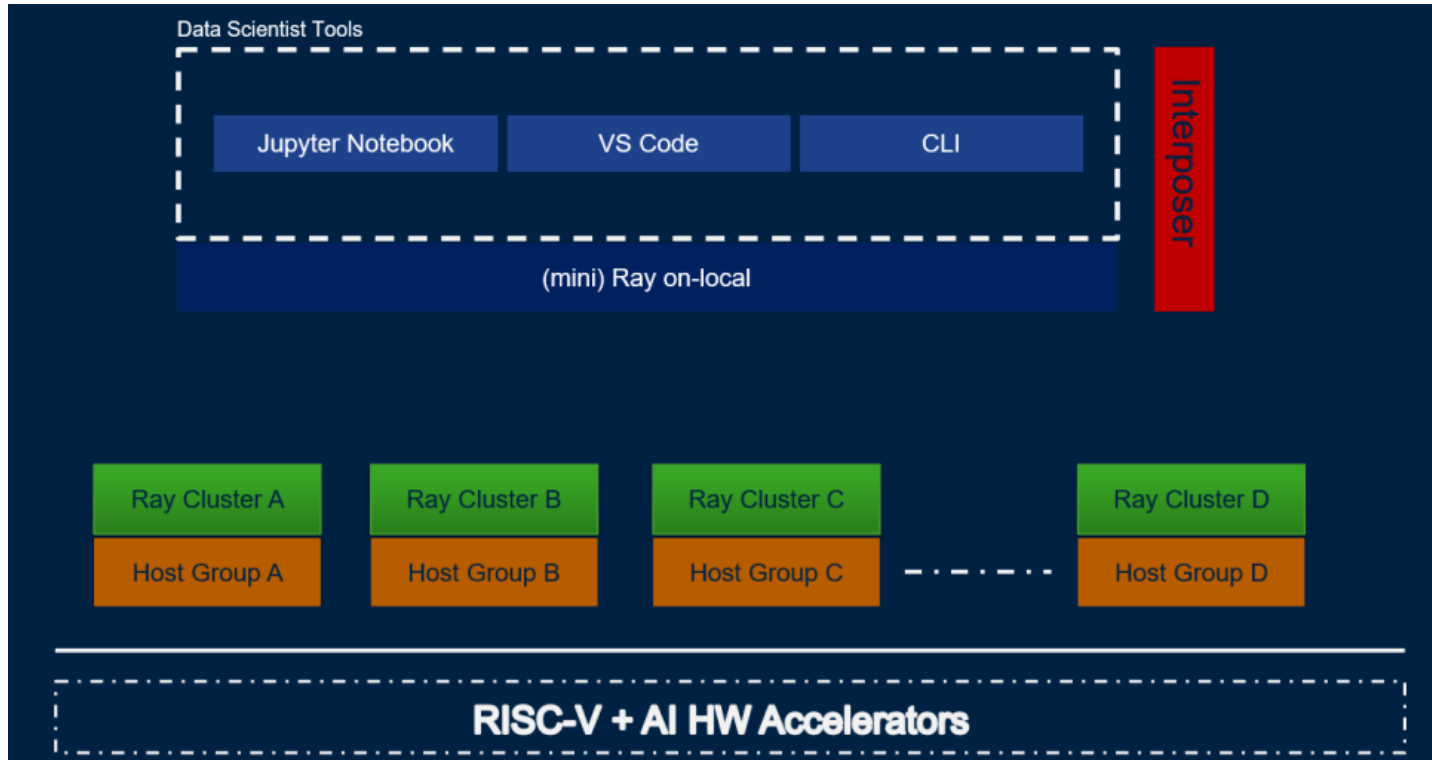
- How - Intercept key APIs at runtime
- e.g – Tensorflow Training

1. loading data
2. define model
3. model.compile()
4. model.fit(model, input, epoch)



# Ray extensions – Remote ML/AI II

## Empower data science tools



# What's likely next ?

## Empower AI everywhere with RISC-V

- Get into Ray community
- Get on Ray cluster (KubeRay)
- Get more cases
  - RISC-V host with AI hardware accelerators

# Accelerate the adoption of RISC-V on AI

Q & A

Thank you!

@Tiejun\_Chen

[tiejun.china@gmail.com](mailto:tiejun.china@gmail.com)

<https://www.linkedin.com/in/tiejun-chen-41a6654b/>

