Speakers

• Yuan Tang
  • GitHub: terrytangyuan
  • Member: Kubeflow
  • Maintainer: MXNet, XGBoost, TensorFlow
  • Senior Software Engineer, Ant Financial

• Yong Tang
  • GitHub: yongtang
  • Maintainer: CoreDNS and Docker
  • Director of Engineering, MobileIron
TensorFlow 2.0 Workflow

Data Input Preprocess → Model Building → Training → Saving Inference

tf.data → tf.keras → tf.estimator → eager → autograph → saved model
## Orchestration for DL

<table>
<thead>
<tr>
<th>IO</th>
<th>IO</th>
<th>IO</th>
<th>IO</th>
<th>IO</th>
<th>IO</th>
<th>IO</th>
<th>IO</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Idle</td>
<td>CPU</td>
<td>CPU</td>
<td>CPU</td>
<td>CPU</td>
<td>CPU</td>
<td>CPU</td>
</tr>
<tr>
<td>GPU</td>
<td>Idle</td>
<td>Idle</td>
<td>GPU</td>
<td>GPU</td>
<td>GPU</td>
<td>GPU</td>
<td>GPU</td>
</tr>
</tbody>
</table>

![Diagram showing the orchestration of IO, CPU, and GPU resources](image_url)
Parameter Server

Worker Node

Worker Node

Worker Node
Reduce
AllReduce

AllReduce == Reduce + Broadcast
Parameter Server

Parallelize on a machine
Parallelize in a cluster
Controversial
Cross device communication cost
Huge efforts invested over the years
Orchestration for DL

Stateful Metadata

Lifecycle Management

Kubernetes for Orchestration

Kubernetes Operators for ML
Kubernetes Operators

- TF Operator
- PyTorch Operator
- MPI Operator
# Kubernetes Operators

<table>
<thead>
<tr>
<th>Framework Support</th>
<th>TF Operator</th>
<th>PyTorch Operator</th>
<th>MPI Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>TensorFlow</td>
<td>tf.distribute MPI/NCCL/PS/TPU</td>
<td>torch.distributed Gloo/MPI/NCCL</td>
<td>horovod DistributedOptimizer (MPI Only)</td>
</tr>
<tr>
<td>PyTorch</td>
<td>PyTorch</td>
<td>PyTorch</td>
<td>horovod DistributedOptimizer (MPI Only)</td>
</tr>
<tr>
<td>TF/Keras/MXNet/PyTorch OpenMPI</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TFJob vs. MPIJob

```
apiVersion: "kubeflow.org/v1beta1"
kind: TFJob
metadata:
  name: distributed-training
spec:
  tfReplicaSpecs:
    Worker:
      replicas: 4
      template:
        spec:
          containers:
            - name: tensorflow
              image: distributed_training_tf:latest
              resources:
                limits: nvidia.com/gpu: 4
              command: "python tf_benchmarks.py"

apiVersion: "kubeflow.org/v1alpha2"
kind: MPIJob
metadata:
  name: distributed-training
spec:
  mpiReplicaSpecs:
    Worker:
      replicas: 4
      template:
        spec:
          containers:
            - name: tensorflow
              image: distributed_training_hovorod:latest
              resources:
                limits: nvidia.com/gpu: 4
              command: "mpirun python hovorod_benchmarks.py"
```
$ pip install tensorflow
$ pip install tensorflow-io

```python
import tensorflow as tf
import tensorflow_io.mnist as mnist_io

dataset = mnist_io.MNISTDataset(image_filenames, label_filenames)
dataset = dataset.map(
    lambda x, y: (tf.image.convert_image_dtype(x, tf.float32), y)).batch(1000)

model = tf.keras.models.Sequential([  
    tf.keras.layers.Flatten(input_shape=(28, 28)),  
    tf.keras.layers.Dense(512, activation=tf.nn.relu),  
    tf.keras.layers.Dropout(0.2),  
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)  
])
model.compile(loss='mse', optimizer='sgd')
model.fit(dataset, epochs=2000)
model.evaluate(dataset)
```
import tensorflow as tf
import tensorflow_io.mnist as mnist_io

dataset = mnist_io.MNISTDataset(...)

model = tf.keras.Sequential([...])

mirrored_strategy = tf.distribute.MirroredStrategy()
with mirrored_strategy.scope():
    model.compile(loss='mse', optimizer='sgd')

model.fit(dataset, epochs=2000)
model.evaluate(dataset)
import tensorflow as tf
import tensorflow_io.mnist as mnist_io
import horovod.keras as hvd

dataset = mnist_io.MNISTDataset(...)

model = tf.keras.Sequential([...])

opt = tf.train.AdagradOptimizer(0.01 * hvd.size())
opt = hvd.DistributedOptimizer(opt)

model.compile(loss='mse', optimizer=opt)

callbacks = [
    hvd.callbacks.BroadcastGlobalVariablesCallback(0),
]

model.fit(dataset, epochs=2000, callbacks=callbacks)
model.evaluate(dataset)
import tensorflow as tf
import tensorflow_io.mnist as mnist_io

dataset = mnist_io.MNISTDataset(...)  

model = tf.keras.Sequential([...])

mirrored_strategy = tf.distribute.MirroredStrategy()  
with mirrored_strategy.scope():
  model.compile(loss='mse', optimizer='sgd')

model.fit(dataset, epochs=2000)
model.evaluate(dataset)

import tensorflow as tf
import tensorflow_io.mnist as mnist_io

import horovod.keras as hvd

dataset = mnist_io.MNISTDataset(...)  

model = tf.keras.Sequential([...])

opt = tf.train.AdagradOptimizer(0.01 * hvd.size())

opt = hvd.DistributedOptimizer(opt)

model.compile(loss='mse', optimizer=opt)

callbacks = [hvd.callbacks.BroadcastGlobalVariablesCallback(0),]
model.fit(dataset, epochs=2000, callbacks=callbacks)
model.evaluate(dataset)
import torch
import horovod.torch as hvd

data_loader = torch.utils.data.DataLoader(train_dataset, batch_size=100)

model = ...

optimizer = torch.optim.SGD(model.parameters())
optimizer = hvd.DistributedOptimizer(
    optimizer, named_parameters=model.named_parameters())
hvd.broadcast_parameters(model.state_dict(), root_rank=0)

for epoch in range(100):
    for batch_idx, (data, target) in enumerate(data_loader):
        optimizer.zero_grad()
        output = model(data)
        loss = torch.nn.functional.F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
Recall: TFJob vs. MPIJob

```
apiVersion: "kubeflow.org/v1beta1"
kind: TFJob
metadata:
  name: distributed-training
spec:
  tfReplicaSpecs:
    Worker:
      replicas: 4
      template:
        spec:
          containers:
            - name: tensorflow
              image: distributed_training_tf:latest
              resources:
                limits: nvidia.com/gpu: 4
              command: "python tf_benchmarks.py"

apiVersion: "kubeflow.org/v1alpha2"
kind: MPIJob
metadata:
  name: distributed-training
spec:
  mpiReplicaSpecs:
    Worker:
      replicas: 4
      template:
        spec:
          containers:
            - name: tensorflow
              image: distributed_training_hovorod:latest
              resources:
                limits: nvidia.com/gpu: 4
              command: "mpirun python hovorod_benchmarks.py"
```
Common APIs and libraries shared by other Kubeflow operator repositories.

<table>
<thead>
<tr>
<th>Branch: master</th>
<th>New pull request</th>
<th>Latest commit f38f5dc 2 days ago</th>
</tr>
</thead>
<tbody>
<tr>
<td>terrytangyuan and k8s-ci-robot</td>
<td>Correct function names in the comment (#32)</td>
<td></td>
</tr>
<tr>
<td>client</td>
<td>Common job controller library (#5)</td>
<td>28 days ago</td>
</tr>
<tr>
<td>hack</td>
<td>Common job controller library (#5)</td>
<td>28 days ago</td>
</tr>
<tr>
<td>job_controller</td>
<td>Correct function names in the comment (#32)</td>
<td>2 days ago</td>
</tr>
<tr>
<td>operator/v1</td>
<td>Fix incorrect name for restart policy exit code (#20)</td>
<td>10 days ago</td>
</tr>
<tr>
<td>test_job/v1</td>
<td>Remove mentions of tensorflow in test job (#21)</td>
<td>10 days ago</td>
</tr>
<tr>
<td>test_util/v1</td>
<td>chore: Fix package name (#27)</td>
<td>8 days ago</td>
</tr>
<tr>
<td>util</td>
<td>Move public util functions to util/status.go</td>
<td>6 days ago</td>
</tr>
<tr>
<td>.gitignore</td>
<td>Added .gitignore file (#16)</td>
<td>15 days ago</td>
</tr>
<tr>
<td>.travis.yml</td>
<td>Update goveralls ignore pattern</td>
<td>9 days ago</td>
</tr>
<tr>
<td>LICENSE</td>
<td>Create LICENSE</td>
<td>a month ago</td>
</tr>
<tr>
<td>OWNERS</td>
<td>Add terrytangyuan to OWNERS</td>
<td>a month ago</td>
</tr>
<tr>
<td>README.md</td>
<td>Add Travis badge and Go report card (#9)</td>
<td>27 days ago</td>
</tr>
</tbody>
</table>
Common and standardized API spec

Base JobController interface

JobController implementation utilities

Testing utilities
THANK YOU