Multi-cloud Machine Learning Data and Workflow with Kubernetes

Lei Xue, Momenta, Infrastructure Tech Lead
Fei Xue fayexuexue@gmail.com
About Us

Lei Xue
• Infrastructure Tech Lead of Momenta
• Contributor of KubeFlow
• Creator of KubeFlow Caffe2 operator
• Maintainer of Kubernetes RDMA device plugin
• Current Interest: AI infrastructure, Storage

Fei Xue
• Early member of the KubeFlow team at Google
• Interests: ML infrastructure, distributed systems
ML-enabled Self-driving
Why Kubernetes for ML?

Portability
Scalability
Isolation
Why multi/hybrid-cloud?

• **On-prem**
  - Built DCs around 2016
  - Customized CPU/memory/network

• **Cloud**
  - Cloud vendors provided stronger GPU/high bandwidth network
  - Better scalability and infrastructure
Multi-Cloud ML Challenges

Data Management

Workflow Orchestration

Heterogeneous Hardware
Challenge #1 Data Management
Data Management Challenges

- Data is critical in deep learning use cases
- Copying data among multiple DCs and cloud regions is a huge pain
- Training job each
  - consumes images in the 10M~
  - Image size 16KB – 5MB
- Typical large-scale small file access problem
  - High requirement for read latency
  - No modification needed to files
- CephFS and NFS are easy to use for training
  - Both CephFS and NFS can be mounted by multiple readers
  - Both of them are “in-tree” in Kubernetes main repository
  - However, it is hard to customize for AI training
Data Service - CSI

expose storage systems in K8s
GA in v1.13
FUSE (Filesystem in Userspace) is an interface for userspace programs to export a filesystem to the Linux kernel. The FUSE project consists of two components:

- **fuse** kernel
- **gofuse** userspace library: gofuse provides the reference implementation for communicating with the FUSE kernel module.
NR (Not Replacement)
- The dataset of AI/ML are always bigger than memory
- The dataset will be read to process for each epoch
- NR will keep the hit ratio for each epoch

<table>
<thead>
<tr>
<th>Policy</th>
<th>Size(GB)</th>
<th>10</th>
<th>50</th>
<th>100</th>
<th>160</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radom</td>
<td></td>
<td>0~6.25</td>
<td>0~31.25</td>
<td>0~62.5</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>LRU</td>
<td></td>
<td>0~6.25</td>
<td>0~31.25</td>
<td>0~62.5</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>FIFO</td>
<td></td>
<td>0~6.25</td>
<td>0~31.25</td>
<td>0~62.5</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>NR</td>
<td></td>
<td>6.25</td>
<td>31.25</td>
<td>62.5</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
Multi-Cloud ML Challenges

- Data Management
- Workflow Orchestration
- Heterogeneous Hardware
Challenge #2
Machine Learning Workflow Orchestration

Multiple levels of distributed training
- single node
- multiple node - parameter server/worker
- multiple node with GPU - mpi

Support multiple training frameworks:
- TensorFlow
- PyTorch
- Caffe & Caffe2
- Horovod
Why Multi-cluster?
- Single-points of failure
- Highly customized hardware in our IDC
- Scaling in the Cloud

Sync resources across clusters
- Job lifecycle
- Node Resource

Why not federation?
My job acquired 8 GPUs and needs 8 more

My job acquired 8 GPUs and needs 8 more
Gang Scheduler

- Gang tasks are a single scheduling unit
  - Admitted, placed, preempted and killed as a group
  - Gang tasks are independent execution units
  - Gang execution is terminated if a gang task fails and cannot be restarted
- Using kube-batch
  - Added priority, preemptive orchestration
  - Connected kube-batch to multiple operators
  - Framework snapshot
My job has lower priority and was preempted...

My job acquired 8 GPUs and needs 8 more
My job has lower priority and was preempted...

My job acquired all 16 GPUs and running
My job was re-submitted and now running

My job is done.
Multi-Cloud ML Challenges

Data Management

Workflow Orchestration

Heterogeneous Hardware
Challenge #3
Heterogeneous Hardware
Kubernetes provides a device plugin framework for vendors to advertise their resources to the kubelet without changing Kubernetes core code. Instead of writing custom Kubernetes code, vendors can implement a device plugin that can be deployed manually or as a DaemonSet. The targeted devices include GPUs, High-performance NICs, FPGAs, InfiniBand, and other similar computing resources that may require vendor specific initialization and setup.

- Nvidia device plugin
- RDMA device plugin
# Heterogeneous Network

<table>
<thead>
<tr>
<th>On-prem</th>
<th>Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RDMA</strong></td>
<td><strong>Ethernet</strong></td>
</tr>
<tr>
<td>25G</td>
<td>25G</td>
</tr>
<tr>
<td>56G</td>
<td>100G</td>
</tr>
</tbody>
</table>

- Not available in China

---

RoCE™, Mellanox Technologies, Alibaba Cloud, AWS
Heterogeneous Network

VM-to-VM network latency is critical to large scale machine learning performance
hustcat/k8s-rdma-device-plugin is a device plugin for Kubernetes to manage RDMA device.

RDMA(remote direct memory access) is a high performance network protocol, which has the following major advantages:

- Zero Copy
- Kernel Bypass - No CPU involvement
DEMO
Multi-Cloud ML Challenges

Data Management
Workflow Orchestration
Heterogeneous Hardware