Composable Infrastructure for On-Prem Kubernetes-Based Systems

Subrahmanyam Ongole
Architect
One Convergence, Inc.
Agenda

- Introduction
- State of the art
- Problem description
- Proposal
- Scale-Out performance
Introduction

- One Convergence Products
  - http://www.oneconvergence.com

- Topic
  - GPU Composition for Kubernetes workloads
Why Scale-out?

- Scale-up vs Scale-out
  - Affordable GPU servers
  - Incrementally add new GPU hardware
  - Resiliency - No single point of failure
  - Higher network speeds via RDMA NICs

- Challenges
  - Cluster management
  - Workload orchestration
  - Resource management
  - Achieving best performance

- On-Prem
  - Cloud providers address this
  - On-Prem needs to be solved
Platform of Choice

- Kubernetes
  - Cluster management
  - Container orchestration
  - Standard interfaces for Network and Storage
    - CNI & CSI
  - Node-specific resource management
    - Device plugins for GPUs, RDMA, etc
GPU Allocation

- **POD Spec**
  - resources:
    - limits:
      - nvidia.com/gpu: 2 # requesting 2 GPUs

- **Different types of GPUs**
  - Label each node with the type of GPU
    - `kubectl label nodes <node-with-k80> accelerator=nvidia-tesla-k80`
    - `kubectl label nodes <node-with-p100> accelerator=nvidia-tesla-p100`
  - Specify using node selectors in the POD spec
    - `nodeSelector:`
      - `accelerator: nvidia-tesla-p100` # or `nvidia-tesla-k80` etc.
Challenges

- User needs to be aware of
  - GPU vendor, Type of GPU and GPU nodes
- Resource segmentation
  - Experimental vs Production jobs
- Better utilization of GPUs
  - Schedule by mutual agreement
- Multi-user
  - Isolation of workloads
- Cluster changes
  - Scale-out/scale-down
  - GPU health
- Topology
  - RDMA, NVLink®, etc
- Complex with increasing number of users/nodes
Extending Kubernetes

- Custom Resources
  - Dynamically extend Kubernetes API
  - CRDs - Custom Resource Definitions
    - Handled by API server
    - Uses Kubernetes storage
    - Custom Controller provides Declarative API
  - Aggregated APIs
    - Separate service, Complex
    - Custom storage

- Operators
  - Combines Custom Resources & Custom Controllers
  - Domain knowledge
  - Examples
    - Etcd, Prometheus operators
    - Tf operator in Kubeflow
Pool & Group Benefits

- Abstracts resources
  - User doesn't need to be aware of GPU hardware
  - Groups determine GPU association
- Better utilization of GPUs
  - Better distribution of workload
- Isolation of workloads
  - Separate Namespace per user
- Topology awareness
  - Schedules RDMA/GD wherever applicable
- Monitors changes to cluster
  - Scale-out/Scale-down
  - GPU health
Disaggregated PCIe JBOG

- **Introduction**
- **Static composition**
  - Fixed at node composition time
- **Dynamic composition**
  - Dynamically attaches to POD
  - GPUs move across nodes
  - Device plugin requirements
Scale-Out Performance

**3 Node Cluster**

Each node contains:

- Lenovo™ Thinksystem™ SD530
- Intel® Xeon® Gold 6148 @2.4 GHz
  - 384 GB RAM
  - 20 Cores
- 2 NVIDIA® V100 GPUs / 16GB
- Mellanox® 100Gbps ConnectX®-5
  - RDMA NIC
- CUDA 9.0
- Cudnn 7.4.1.5-1
- TensorFlow 1.12
- Mellanox OFED 4.5-1.0.1.0
- NCCL openmpi-3.0.0
- Horovod: 0.15.2
- DKube/DFabric™ 1.0.3
Summary

- Scale out architecture
  - http://www.oneconvergence.com/blogs/
- Platform requirements
  - DFabric
  - http://www.oneconvergence.com/dfab

Thank You Questions?