

Kubernetes and GPUs

Ricardo Rocha
CERN IT-CM-RPS

Hardware

Some older cards: Nvidia K20s, AMD Vega

Mostly focusing on Nvidia enterprise cards: P100, V100, T4

Started looking at use cases for each type of card

- V100s show better performance

- T4s have less cores but lower power consumption (70W), passive cooling

Integrating GPUs

Baremetal is straightforward

For virtual machines there are two possibilities

PCI Passthrough: full device exposed and dedicated to one virtual machine

vGPUs: *partitions* of the GPU exposed to the virtual machine

At CERN we currently offer baremetal and PCI passthrough

Ongoing work for Nvidia vGPUs

Use cases for all three possible setups

OpenStack Integration

Our cloud infrastructure relies on OpenStack

Support status for the different modes

	Scheduling	Quotas	Monitoring	Sharing	NUMA	Performance	License
Baremetal	Flavors	No	No	No	Yes	Best	x
PCI Passthrough	Flavors	No	No	No	Yes	Best	x
vGPU	Yes	Ongoing	Possible	Yes	No	< 10% loss?	vComputeServer *

*

Nvidia
~\$50 / year
up to 8 VMs / GPU

OpenStack Integration

vGPUs seem like the easiest path to get GPUs as first class resources

Required integration almost finished for proper scheduling and quotas

Won't work for other type of resources, FPGAs, ...

We plan to add this mode to our offering very soon

Kubernetes Integration

Relying on *node groups/pools* to get heterogeneous clusters

Different flavor for each node group: m2.xlarge, g1.xlarge, ...

Automated detection and configuration of GPUs

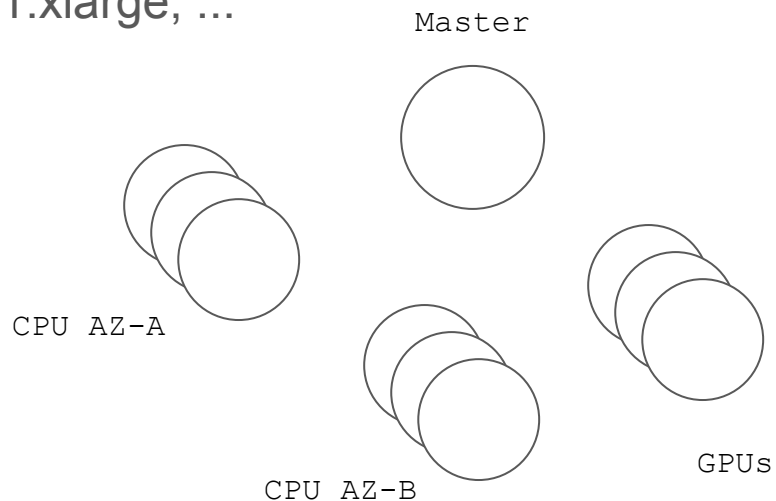
Managed by a Helm chart

<https://gitlab.cern.ch/helm/charts/cern/tree/master/nvidia-gpu>

Alternative with the GPU operator

<https://github.com/NVIDIA/gpu-operator>

<https://github.com/NVIDIA/k8s-device-plugin>



Kubernetes Integration

GPUs are already treated as first class resources in Kubernetes

Scheduling	Quotas	Monitoring	Sharing	NUMA	Performance	License
Yes	Yes	Yes	No	Yes	Best	x

apiVersion: v1

kind: Pod

metadata:

name: mypod

spec:

containers:

- name: mycontainer

image: "mycontainerimage:tag"

resources:

limits:

nvidia.com/gpu: 1 # no fractions possible

```
$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
...				
hub-001-gpu-nvi...7iz-node-0	Ready	gpu	63d	v1.15.3
hub-001-gpu-nvi...7iz-node-1	Ready	gpu	63d	v1.15.3

Kubernetes Integration

Monitoring is done with a Nvidia Prometheus exporter

Metrics for temperature, SM/Memory clocks, power usage, throughput, ...



<https://github.com/NVIDIA/gpu-monitoring-tools/>

Next Steps

Evaluate performance penalty of using vGPUs

- Likely to keep offering both PCI passthrough and vGPUs

- Explore the Kubernetes TopologyManager for NUMA affinity

Try out SR-IOV for GPU virtualization of non Nvidia vendors

No concrete requests for accelerators other than GPUs on Kubernetes yet

- Occasional requests to integrate special devices (tape drives, USB dongles)

Backup

Ongoing Work - OpenStack

Quota integration for vGPUs in OpenStack Nova using unified limits, placement

<https://blueprints.launchpad.net/nova/+spec/count-quota-usage-from-placement>

<https://specs.openstack.org/openstack/nova-specs/specs/ussuri/approved/unified-limits-nova.html>

Missing NUMA topology knowledge when scheduling vGPUs

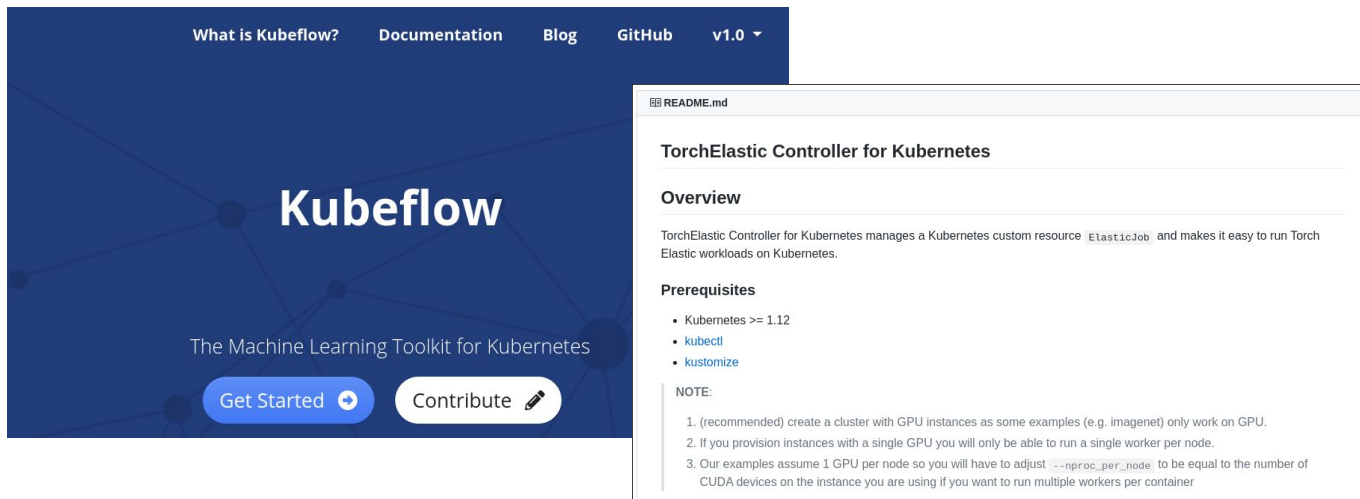
https://bugzilla.redhat.com/show_bug.cgi?id=1553832

Performance penalty when CPU/GPU in different NUMA nodes

Machine Learning and Kubernetes

Kubernetes is a popular way to scale out ML workloads

Good integration in popular frameworks



The image shows a composite view of the Kubeflow project. On the left is the main website landing page, which has a dark blue background with a network diagram. The word "Kubeflow" is prominently displayed in white. Below it, the tagline "The Machine Learning Toolkit for Kubernetes" is visible. At the bottom of the page are two buttons: "Get Started" with a plus icon and "Contribute" with a pencil icon. The top navigation bar includes links for "What is Kubeflow?", "Documentation", "Blog", "GitHub", and a version dropdown set to "v1.0".

On the right is a preview of the "README.md" file. The title is "TorchElastic Controller for Kubernetes". Under the "Overview" section, it states: "TorchElastic Controller for Kubernetes manages a Kubernetes custom resource `ElasticJob` and makes it easy to run Torch Elastic workloads on Kubernetes." The "Prerequisites" section lists:

- Kubernetes ≥ 1.12
- [kubectf](#)
- [kustomize](#)

A "NOTE:" section follows with three numbered points:

1. (recommended) create a cluster with GPU instances as some examples (e.g. imagenet) only work on GPU.
2. If you provision instances with a single GPU you will only be able to run a single worker per node.
3. Our examples assume 1 GPU per node so you will have to adjust `--nproc_per_node` to be equal to the number of CUDA devices on the instance you are using if you want to run multiple workers per container