Everything you need to know about using GPUs with Kubernetes

Rohit Agarwal <agarwalrohit@google.com>
Software Engineer, Google Cloud
@mindprince

Agenda

How? Not Why. Not When.

What makes it hard.

History

As a user

As an operator

What’s missing

Containers and GPUs

Containers: package your application and its dependencies, run-anywhere.

Except when dependency is a kernel module.

Using NVIDIA GPUs require: NVIDIA kernel module, user-level libraries (`libnvidia-ml.so, libcuda.so` etc.)

User-level library version needs to be the same as the kernel module version.

Images with user-level libraries not portable.

alpha.kubernetes.io/nvidia-gpu

First attempt: let the user deal with dependencies.

Kubernetes would expose GPU devices as schedulable resources.

Add them to the container when requested.

alpha.kubernetes.io/nvidia-gpu

How to access the device?

Install kernel module and libraries on host. Use `hostPath` volumes.

Worked. Terrible. Not portable.

```
spec:
  volumes:
    - name: "nvidia-libraries"
      hostPath:
        path: "/usr/lib/nvidia-375"
  containers:
    - name: my-gpu-container
      image: "gcr.io/proj/gpu-image:v0.1"
      resources:
        limits:
          alpha.kubernetes.io/nvidia-gpu: 2
      volumeMounts:
        - name: "nvidia-libraries"
          mountPath: "/usr/local/nvidia/lib64"
```

alpha.kubernetes.io/nvidia-gpu

In-tree. What about AMD GPUs, Intel GPUs, Xilinx FPGAs etc.?

Deprecated in v1.10 (#57384)

Removed in v1.11 (#61498)

Device Plugins

Support generic devices.

Vendor specific code out-of-tree.

Enable portable `PodSpec`.

Device Plugins: `nvidia.com/gpu`

How's the container getting access to the user-level libraries present on the host?

Device plugin APIs.

Portable container image and `PodSpec`.

```
spec:
  containers:
    - name: my-gpu-container
      image: "gcr.io/proj/gpu-image:v0.1"
      resources:
        limits:
          nvidia.com/gpu: 2
```

Device Plugins

Introduced in v1.8.

Beta in v1.10.

Start using them!

Recap: as a user

Build your images without user-level shared libraries.

Images still include the CUDA toolkit. Some dependence on host driver version.

Request for `nvidia.com/gpu` resources.

Cluster with multiple GPU types.

Application should run on a particular type of GPU.

No native/portable way of targeting.

Label nodes with GPU type. Use `nodeSelector`.

```
spec:
  containers:
    - name: my-gpu-container
      image: "gcr.io/proj/gpu-image:v0.1"
      resources:
        limits:
          nvidia.com/gpu: 2
  nodeSelector:
    cloud.google.com/gke-accelerator: nvidia-tesla-k80
```

As an operator

Have nodes with GPUs!

Multiple types of GPU nodes, label them.

Install the NVIDIA driver.

- Parts of driver closed source. Linux is GPL licensed.
- Keep up with driver version required by the latest CUDA release.

Install the device plugin. NVIDIA's. Google's. Possible future convergence.

Resource Quota

Added in v1.10.

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: my-gpu-quota
spec:
  hard:
    requests.nvidia.com/gpu: 4
```

GPU Monitoring

Support for two metrics that users care about the most:

```
memory_used, memory_total
duty_cycle
```

Collected by cAdvisor using NVML.

Accessed using cAdvisor's prometheus endpoint, Heapster or Stackdriver.

Added in v1.9.

Dedicated nodes for GPU workloads

GPUs are expensive.

Prevent pods not requesting GPUs from scheduling on GPU nodes.

Aggressively downscale GPU nodes.

Taints.

ExtendedResourceToleration admission controller. Added in v1.9.

What’s missing?

No GPU support in minikube.

No fine grained quota control.

More GPU metrics can be added.

No support for GPU sharing.

Not aware of GPU topology.

Autoscaling support is non-ideal.

On GKE

```
$ gcloud beta container clusters create my-gpu-cluster \
--accelerator=type=nvidia-tesla-p100 --cluster-version 1.9
```

```
$ kubectl apply -f \
https://raw.githubusercontent.com/GoogleCloudPlatform/container-engine-acc
elerators/k8s-1.9/nvidia-driver-installer/cos/daemonset-preloaded.yaml
```

Questions?

Thank you!