Multi-Instance GPU Support with the GPU Operator v1.7.0

June 15, 2021 | Kevin Pouget | 5-minute read

Hybrid cloud

< Back to all posts

The Performance and Latency Sensitive Applications (PSAP) team at Red Hat works on the enablement and optimization of Red Hat OpenShift to run compute-intensive enterprise workloads and AI/ML applications effectively and efficiently. As a team of Linux and performance enthusiasts who are always pushing the limits of what is possible with the latest and greatest upstream technologies, we have been collaborating with NVIDIA on the GPU Operator to see how we can leverage the technology moving forward.

GPU Operator MIG support

Version 1.7.0 of the GPU Operator has just landed in OpenShift OperatorHub, with many different updates. We are proud to announce that this version comes with the support of the NVIDIA Multi-Instance GPU (MIG) feature for the A100 and A30 Ampere cards. MIG is the capability of the NVIDIA GPU card to be partitioned into multiple instances and exposed into

pods as independent GPUs. See NVIDIA documentation for further details.

This MIG support on the GPU Operator comes from a joint effort between the NVIDIA Cloud Native team and Red Hat Performance and Latency Sensitive Applications (PSAP) team.

In the following sections, we give an overview of the MIG feature of the NVIDIA A100 GPU. Then, we explain how to configure the MIG advertisement strategy while creating the GPU Operator ClusterPolicy. We continue with a step-by-step guide to (re)configure the MIG geometry. Please keep in mind, though, that one key limitation of this version is that the draining of the workloads running on the GPU being configured is the responsibility of the cluster administrator. If the GPU is busy when the reconfiguration is requested, the GPU Operator will fail to delete the MIG instances. See the troubleshooting section at the end of this document for further details.

NVIDIA Multi-Instance GPU

The Multi-Instance GPU (MIG) capability of the new NVIDIA Ampere architecture allows the partitioning of the hardware resources into multiple GPU instances, each exposed as an independent CUDA-capable GPU to the Operating System. These GPU instances are designed to accommodate multiple independent CUDA applications (up to seven), so they operate in full isolation from each other, with dedicated hardware resources.

As an example, the NVIDIA A100-SXM4-40GB product has 40GB of RAM (gb) and seven GPU-compute units (g). It can be configured with the following profiles (source NVIDIA):

In the following section, we present how to configure the MIG geometry in OpenShift. The illustrations use NVIDIA MIG device name convention (source NVIDIA):

To learn more about NVIDIA A100 GPU and its GPU Operator support, see NVIDIA documentation:

- https://www.nvidia.com/en-us/technologies/multi-instance-gpu
- https://docs.nvidia.com/datacenter/tesla/mig-user-guide/index.html

Configuring MIG Devices in OpenShift

MIG Advertisement Strategy

The GPU Operator exposes GPUs to Kubernetes as extended resources that can be requested and exposed into Pods and containers. The first step of the MIG configuration is to decide how the MIG instances will be advertised to Kubernetes:

- **Single**: this strategy exposes the MIG instances a nvidia.com/gpu resources, identically, as usual non-MIG capable (or with MIG disabled) devices. In this strategy, all the GPUs in a single node are configured in a homogenous manner (same number of compute units, same memory size). This strategy is appropriate for a large cluster where the infrastructure teams can configure “node pools” of different MIG geometries and make them available to users.

Examples for the A100-40GB:

- 1g.5gb: 7 nvidia.com/gpu instances, or
- 2g.10gb: 3 nvidia.com/gpu instances, or
- 3g.20gb: 2 nvidia.com/gpuinstances, or
- 7g.40gb: 1 nvidia.com/gpu instances

- **Mixed**: this strategy exposes to Kubernetes the number of compute units and the memory size of each instance available. There is no constraint on the geometry; all the combinations allowed by the GPU are permitted. This strategy is appropriate for a smaller cluster, where on a single node with multiple GPUs, each GPU can be configured in a different MIG geometry.

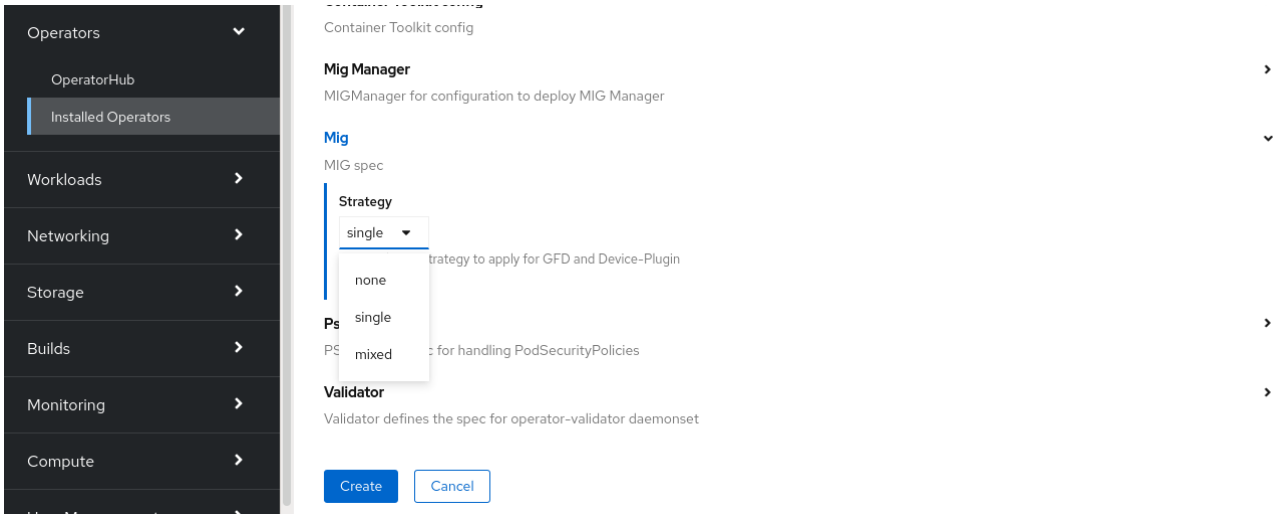
  Examples for the A100-40GB:

  - All the **single** configurations are possible
  - A "balanced" configuration:
    - 1g.5gb: 2 nvidia.com/mig-1g.5gb instances, and
    - 2g.10gb: 1 nvidia.com/mig-2g.10gb instance, and
    - 3g.20gb: 1 nvidia.com/mig-3g.20gb instance

This field must be configured in the GPU Operator ClusterPolicy during its **initial** deployment.

Please note that updates to the ClusterPolicy *after* it has been deployed are currently silently ignored.

```
ClusterPolicy.spec.mig.strategy = [single|mixed|none]
```



Then proceed with the rest of the ClusterPolicy configuration as per the documentation.

MIG Configuration

The actual MIG configuration is performed by a kubernetes controller running on each of the nodes: the mig-manager. This controller watches the node's nvidia.com/gpu.deploy.mig-manager label, and when it detects that the value changed, it triggers nvidia-mig-parted system tool to perform the actual GPU reconfiguration.

The valid MIG configurations are stored as a config-map object. See NVIDIA MIG-Parted repository for further information about this configuration file.

1. Decide the configuration to apply to the node from the pre-populated list, or add a custom one:

```
$ oc edit ConfigMap/mig-parted-config -n gpu-operator-resources
```

2. Verify that the MIG device of the node has been properly discovered by the GPU Operator

```
$ oc get nodes -oname -lnvidia.com/gpu.deploy.mig-manager=true | grep $NODE_NA
```

3. Apply the desired MIG configuration

```
$ MIG_CONFIGURATION=all-1g.5gb
$ oc label node/$NODE_NAME nvidia.com/mig.config=$MIG_CONFIGURATION --overwrit
```

4. Wait for the mig-manager to perform the reconfiguration

   - Follow the reconfiguration logs:

   ```
   $ oc logs ds/nvidia-mig-manager --all-containers -f --prefix
   ```

   - Follow the status of the reconfiguration (success → pending → success, hopefully avoid failure):

5. Confirm that the MIG resources have been exposed

```
$ oc get nodes/$NODE_NAME -ojsonpath={.status.allocatable}
        | jq -c 'with_entries(select (.key | startswith("nvidia.com")))' | jq
```

(It is expected that old MIG configurations remain listed with a count of 0 instances available.)

MIG Instance Discoverability With Node Labels

Once the MIG geometry has been configured, NVIDIA GPU-Feature-Discovery (a sidecar container of the Node Feature Discovery Operator) Pods will label the node with various

information about the physical GPU, the current MIG configuration, as well as the software versions (CUDA and driver). Used in conjunction with the resource reservation system, these labels allow users to provide additional constraints and requirements for the selection of the node that will execute their workload:

```
$ oc get nodes/$NODE_NAME --show-labels | tr ',' '\n' | grep nvidia.com
nvidia.com/gpu.count=1
nvidia.com/gpu.family=ampere
nvidia.com/gpu.machine=PowerEdge-R7425
nvidia.com/gpu.memory=40536
nvidia.com/gpu.product=A100-PCIE-40GB
nvidia.com/mig-2g.10gb.count=3
nvidia.com/mig-2g.10gb.engines.copy=2
nvidia.com/mig-2g.10gb.engines.decoder=1
nvidia.com/mig-2g.10gb.engines.encoder=0
nvidia.com/mig-2g.10gb.memory=9984
nvidia.com/mig-2g.10gb.multiprocessors=28
nvidia.com/mig-2g.10gb.slices.ci=2
nvidia.com/mig-2g.10gb.slices.gi=2
nvidia.com/mig.config=all-2g.10gb
nvidia.com/mig.config.state=success
nvidia.com/mig.strategy=mixed
nvidia.com/cuda.driver.major=460
nvidia.com/cuda.driver.minor=73
nvidia.com/cuda.driver.rev=01
nvidia.com/cuda.runtime.major=11
nvidia.com/cuda.runtime.minor=2
[...]
```

Troubleshooting

The MIG reconfiguration is handled exclusively by the controller deployed within the nvidia-mig-manager DaemonSet. Looking at the logs of these Pods should give a clue about what went wrong.

In particular, the cluster admin is currently expected to drain the node from any GPU workload, before requesting the MIG reconfiguration. If the node is not properly drained, the mig-manager will fail with this error in the logs:

```
 Updating MIG config: map[2g.10gb:3]
Error clearing MigConfig: error destroying Compute instance for profile '(0, 0)':
Error clearing MIG config on GPU 0, erroneous devices may persist
Error setting MIGConfig: error attempting multiple config orderings: all ordering
```

```
Restarting all GPU clients previously shutdown by reenabling their component-spec
Changing the 'http://nvidia.com/mig.config.state' node label to 'failed'
```

In this case, the cluster admin should properly drain the node, then retrigger the reconfiguration by forcing the label update:

```
$ oc label node/$NODE_NAME nvidia.com/mig.config- --overwrite
$ oc label node/$NODE_NAME nvidia.com/mig.config=$MIG_CONFIGURATION --overwrite
```

This release of the MIG support in OpenShift is the first milestone in the road of dynamic management of GPU resources. Stay tuned for the following steps coming soon, such as the benchmarking of the AI/ML capabilities of the different MIG instances and a validation of their guaranteed quality of service (QoS). In the longer term, we can also think of an operator able to track the GPU resource requests and decide how to reconfigure the MIG-capable GPUs to achieve the best throughput (similarly to what the Kubernetes Cluster autoscaler is doing at the node level). We can also imagine the operator monitoring the GPU usage of workload Pods, and suggest more appropriate slice profiles.

We believe that Linux containers and container orchestration engines, most notably Kubernetes, are well positioned to power future software applications spanning multiple industries. Red Hat has embarked on a mission to enable some of the most critical workloads like machine learning, deep learning, artificial intelligence, big data analytics, high-performance computing, and telecommunications, on Red Hat OpenShift. By developing the necessary performance and latency-sensitive application platform features of OpenShift, the PSAP team is supporting this mission across multiple footprints (public, private, and hybrid cloud), industries and application types.

About the author

More like this

Blog post

Red Hat and Sylva unify the future for telco cloud

Blog post

NetApp's certified OpenShift operator for Trident

Original podcast

Crack the Cloud_Open | Command Line Heroes

Original podcast

Edge computing covered and diced | Technically Speaking