WHITE PAPER–MAY 2020

Network-attached Full or Partial GPUs to Any Kubernetes Cluster Using vSphere Bitfusion

vSphere Bitfusion Guides

vmware®

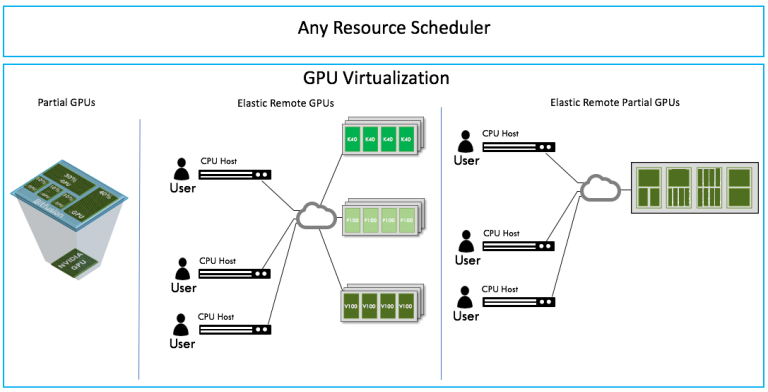
Table of Contents

Network-Attached Full or Partial GPUs to Any Kubernetes Cluster Using vSphere Bitfusion

Kubernetes is an open-source platform for automating deployment, for scaling, and for managing containerized applications.

vSphere Bitfusion is a transparent virtualization layer combining heterogeneous GPUs across multiple vendor GPU systems into a single GPU pool to support slicing, sharing, and pooling of GPUs anywhere. vSphere Bitfusion runs in user space, on the guest operating system, and can be used with any cluster resource scheduler transparently. In the AI space, data scientists no longer need dedicated GPU machines; using vSphere Bitfusion instead, multiple data scientists can share GPUs pooled across the cluster at any granularity. Before continuing, become familiar with vSphere Bitfusion itself with the resources at *https://www.vmware.com/solutions/business-critical-apps/hardwareaccelerators-virtualization.html*.

Figure 1. vSphere Bitfusion Partial GPUs, Remote Attach GPUs, Remote Partial GPUs

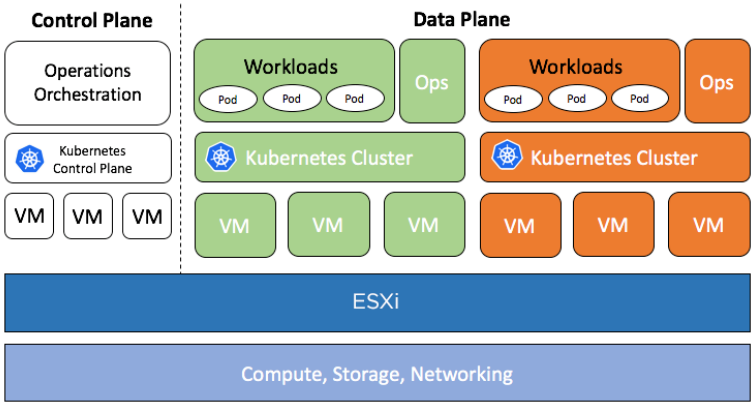


The following sections provide the flow of setting elastic fractional network-attached GPUs within Kubernetes using vSphere Bitfusion.

Kubernetes Environment

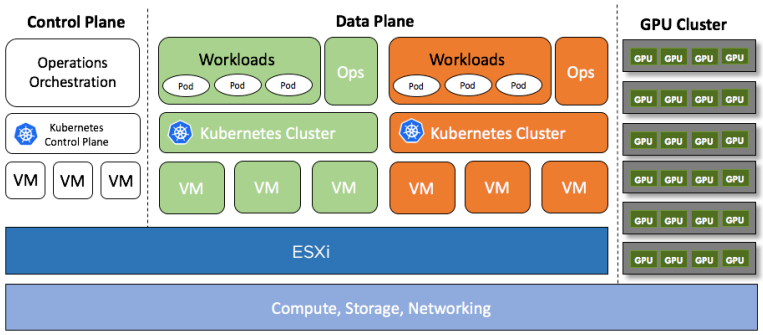
Figure 2 is an overview of a typical Kubernetes environment. Usually Kubernetes environments have a control plane and a data plane.

Figure 2. Typical Kubernetes



When GPUs are brought to your environment, vSphere Bitfusion GPU servers connect directly to the network, separately from the Kubernetes cluster, as shown in Figure 3, simplifying integration with vSphere Bitfusion.

Figure 3. Kubernetes Environment with GPU Cluster

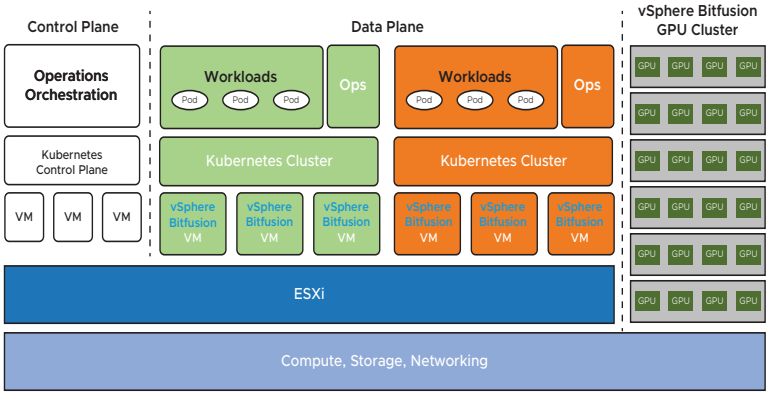


You can integrate vSphere Bitfusion within your Kubernetes environment without any changes to Kubernetes. When integrating with vSphere Bitfusion, Kubernetes does not need to be aware of the GPU cluster (or clusters). vSphere Bitfusion will be the entity that ties the Kubernetes cluster to the GPU clusters.

The following assumes that you have already installed Kubernetes on your CPU cluster. There are a lot of solutions for installing and managing Kubernetes, such as VMware's PKS (also Kubespray, Kubeadm, Kops on AWS, Rancher Kubernetes Engine, etc.). Refer to https://kubernetes.io/docs/setup/ for instructions to install and set up Kubernetes either on-premises or in the cloud.

Note that when using vSphere Bitfusion GPU allocation, you do not need to use any other GPU scheduling/allocation within Kubernetes. In other words, just install and set up Kubernetes as if it is being set up for CPU nodes. The GPUs (from multiple clusters) will be managed as a separate pool via vSphere Bitfusion. Figure 4 shows what the environment looks like after vSphere Bitfusion is installed. The control plane of Kubernetes doesn't need to be vSphere Bitfusion-aware (no need for new configuration on integration of Kubernetes).

Figure 4. vSphere Bitfusion Integrated into Kubernetes Environment with GPU Cluster



Launching Kubernetes Pods with Bitfusion GPUs

The following outlines the steps for dynamically attaching GPUs to a Kubernetes job at runtime, much like storage, using vSphere Bitfusion; this doesn't require any major modifications to Kubernetes. Note that for your specific Kubernetes environment, there may be some modifications to the steps outlined below, but at a generic level, the principles will apply.

1. Deployment of vSphere Bitfusion GPU Servers

vSphere Bitfusion servers are VMware appliances shipped as OVAs. Obtain the OVA and deploy it as a VM on each GPU host, passing the GPUs with DirectPath I/O. An installation guide is delivered with the OVA.

2. Run the "Enable vSphere Bitfusion" action, from vCenter, on client machines

The vSphere Bitfusion installation guides will also describe how to set up client machines to run vSphere Bitfusion software.

3. Create vSphere Bitfusion base images for all containers that will run apps using GPUs

The vSphere Bitfusion user guides include instructions on using vSphere Bitfusion in Docker containers. Briefly, in addition to installing the application and dependencies, install two other packages: a vSphere Bitfusion client package and open-vm-tools.

4. Running a Job with vSphere Bitfusion Health Check

To test your system at this stage, run a vSphere Bitfusion health check as a Kubernetes job. This will run the health check across all GPU servers and on the local CPU host (where Kubernetes jobs are scheduled). Here is a sample Kubernetes podspec which can be kicked off as a Kubernetes job to do the same.

```
containers:
  - name: test
    args:
    - /bin/bash
    - -c
    - |
      bitfusion health
```

Once the health check job is completed, you will see the health check log to looks something like below.

```
Health report for server: 172.16.31.206:56001
=============================================


Software Versions checks:
=========================
[PASS    ]  Check library dependency
[PASS    ]  Check CUDA version >= 9000: Current version: 10020
[PASS    ]  Check GPU driver version >= 440.33: Current version: 440.64.00

System Resources checks:
========================
[PASS    ]  Check bitfusion install
[PASS    ]  Check external connectivity to Bitfusion license server
[PASS    ]  Check shadow memory 96679MB and total gpu mem 65020MB
```

```
Performance checks:
=======================

[PASS     ]  Check Interface/Subnet Compatibility: network interfaces and subnets configured correctly
[PASS     ]  Check PCIe Width:: GPUs are all set to use their maximum PCIe lane capacity
[PASS     ]  Check stream memory ops support
[PASS     ]  Check host pointer on device support
[PASS     ]  Check ulimit -n >= 4096: 4096
[PASS     ]  Check MTU Size: hi-speed interfaces MTU >= 4K


Stability checks:
=======================

[PASS     ]  Check PCIe P2P support: PCIe p2p supported
[PASS     ]  Check GPU API mismatch
[PASS     ]  Check GPU Xid errors
[PASS     ]  Check temperature <= 100.00: current temp: 32.00
[PASS     ]  Check GPU ECC errors
[PASS     ]  Check compute mode
[MARGINAL]  Check Network Errors/Drops: drops reported in file: /sys/class/net/ens160/statistics/rx_dropped


Consistency checks:
=======================

PASS: 17
INFO: 0
SKIPPED: 0
MARGINAL: 2
FATAL: 0


Health report for localhost
=========================================


Software Versions checks:
=======================

[PASS     ]  Check library dependency
[MARGINAL]  Unable to parse the version of the CUDA toolkit that cuDNN was compiled for


System Resources checks:
=======================

[PASS     ]  Check bitfusion install
[PASS     ]  Check external connectivity to Bitfusion license server


Performance checks:
=======================

[PASS     ]  Check Interface/Subnet Compatibility: network interfaces and subnets configured correctly
[MARGINAL]  Check MTU Size: 10000Mbps interface ens192 has low MTU: 1500 < 4K
[SKIPPED ]  Cannot perform PCIe diagnosis without root access. Run the binary with sudo to get diagnosis


Stability checks:
=======================

[PASS     ]  Check Network Errors/Drops: found no errors or packet drops
[PASS     ]  Check PCIe P2P support: PCIe p2p supported


Consistency checks:
=======================

PASS: 6
INFO: 0
SKIPPED: 1
MARGINAL: 2
FATAL: 0
```

The health check will run checks on the cluster nodes appropriate to their hardware (e.g., GPUs) or to their configuration (e.g., RoCE), so your output may differ from that shown above.

vSphere Bitfusion health checks try to find settings or problems that will limit or prevent the high-bandwidth, low-latency communication needed for the best performance. The results should be self-explanatory. Checks will be performed on the client and on all configured GPU servers.

5. Running with Network-attached vSphere Bitfusion GPUs

For every job that needs a GPU, prepend the command (under the container's args) with

```
bitfusion run -n [-p | -m ] --
```

(as shown below) to dynamically attach full or fractional GPUs to the job as needed at runtime. Note that the container can also consume locally available GPUs directly (if you've configured the system for this) as you would without vSphere Bitfusion, simply by not specifying

```
bitfusion run - ….
containers:
 - name: test
 args:
 - /bin/bash
 - -c
 - |
 bitfusion run -n 1 -p 0.5 -- python benchmarks/scripts/tf_cnn_benchmarks/tf_cnn_
benchmarks.py \
 --local_parameter_device=gpu --variable_update=replicated \
 --model=inception3 \
 --batch_size=32 \
 --num_gpus=1 \
 --use_nccl=false \
```

Figure 5 is the logical structure of what the containers launched in Kubernetes with vSphere Bitfusion GPUs look like; Figure 6 summarizes the above steps in a flowchart.

Figure 5. Containers Launched in Kubernetes with Bitfusion

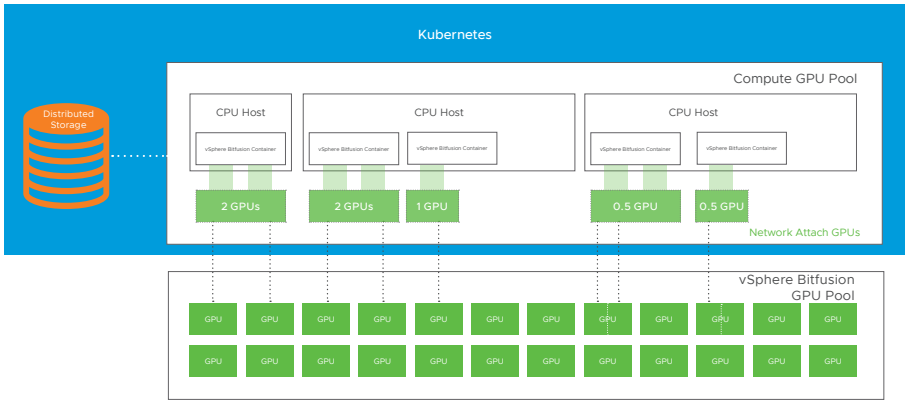


Figure 6. Flowchart of Enabling Network-Attached GPUs on Kubernetes with vSphere Bitfusion

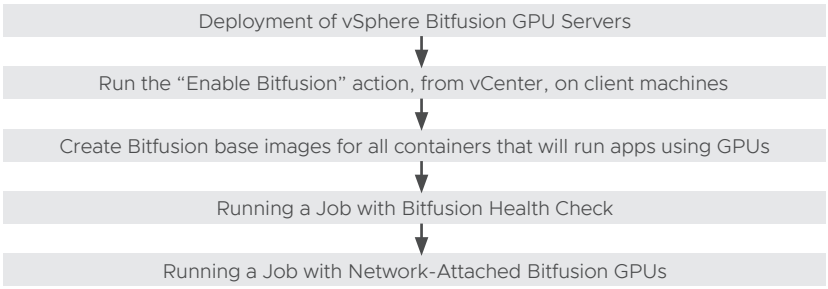


Figure 7 shows a sample client job podspec in Kubernetes to spin up a job to run a tf_cnn benchmark with 0.5 GPUs using vSphere Bitfusion. Any of the vSphere Bitfusion run command options can be prepended with the actual application commands (in this case the command is python benchmarks/scrips/tf_cnn_benchmarks/tf_cnn_benchmarks.py …)

Figure 7. Sample Kubernetes Podspec with vSphere Bitfusion GPUs

```
podSpec:
  volumes:
  - name: bitfusion
  containers:
  - name: test
    args:
    - /bin/bash
    - -c
    - |
      bitfusion run -n 1 -p 0.5 -- python benchmarks/scripts/tf_cnn_benchmarks/tf_cnn_
benchmarks.py \
        --local_parameter_device=gpu -variable_update=replicated \
        --model=inception3 \
        --batch_size=32 \
        --num_gpus=1 \
        --use_nccl=false \
```

Conclusion

vSphere Bitfusion makes it possible to disaggregate GPUs from CPU instances and to request them in any granularity only when needed. This leads to much more flexible cluster designs, better utilization, and an overall lower cost of ownership. vSphere Bitfusion can be leveraged to elastically attach full or partial GPUs over the network at application run time to any Kubernetes cluster.

vSphere Bitfusion has many more advanced features for managing and allocating virtual GPUs, and this brief guide serves only as an introduction. For some of the more advanced features, see the technical documentation at *https://www-review.vmware.com/solutions/business-critical-apps/hardwareaccelerators-virtualization.html*.

ItemNo:VMW-0520-2210_VMW_CPBUTechnicalWhitePapers_BitfusionDocs_22NetworkAttachedFullorPartialGPUstoAnyKibernetesClusterusingvSphereBitfusionFlexDirect_1.3_MMT5/20