WHITE PAPER

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

Bitfusion Guides

vmware®

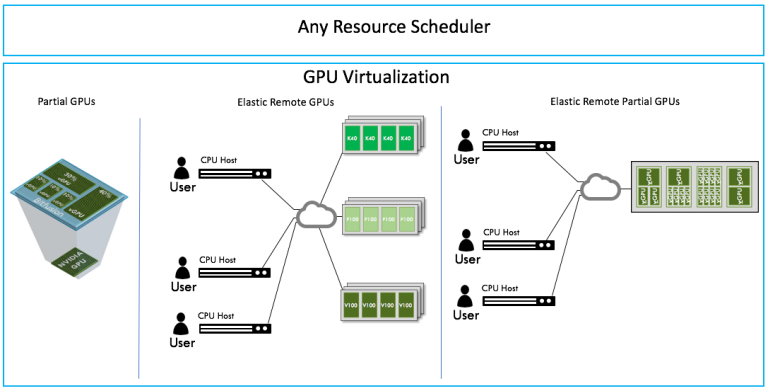
Table of Contents

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

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

Bitfusion FlexDirect 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. FlexDirect runs in user-space, in the guest operating system, and can be used with any cluster resource scheduler transparently. In the AI space, instead of providing dedicated GPU machines to each of the data scientists, GPUs can be pooled across the cluster for access in any granularity by multiple data scientists using FlexDirect. Before continuing, become familiar with Bitfusion itself with the resources at *https://www.vmware.com/solutions/business-critical-apps/hardwareaccelerators-virtualization.html*.

Fig 1. FlexDirect Partial GPUs, Remote Attach GPUs, Remote Partial GPUs

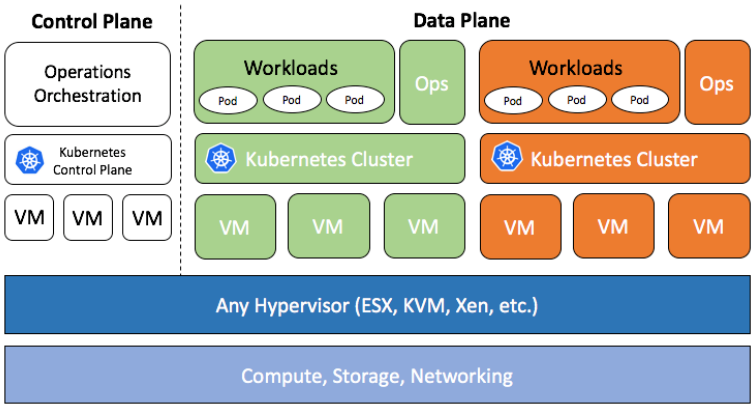


The following sections provide the flow of setting elastic fractional network attached GPUs within Kubernetes using FlexDirect.

Kubernetes Environment

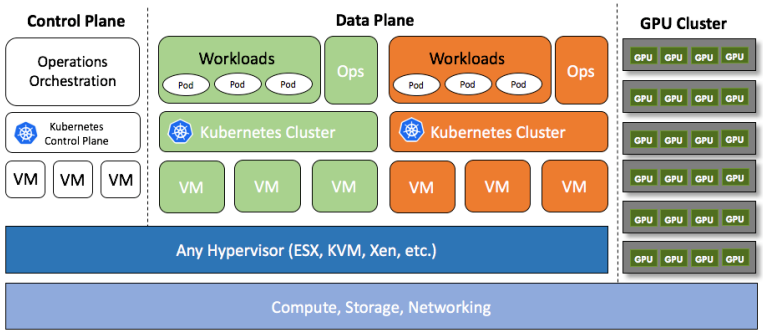
Figure 2 is an overview of a typical Kubernetes environment. Usually Kubernetes environments have a control plane and a data plane.

Fig 2. Typical Kubernetes Environment



When GPUs are brought to your environment, FlexDirect recommends connecting the GPU servers connected to the Ethernet but separate from the Kubernetes cluster, as shown in Figure 3, to enable easier integration with FlexDirect.

Fig 3. Kubernetes Environment with GPU Cluster

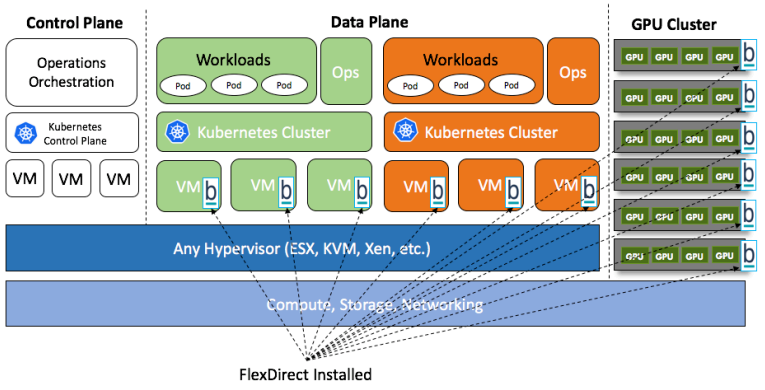


You can integrate FlexDirect within your Kubernetes environment without any changes to Kubernetes. When integrating with FlexDirect, Kubernetes do not need to be aware to the GPU cluster (or clusters). FlexDirect will be the entity that ties the Kubernetes cluster to the GPU clusters.

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

Note that when using FlexDirect 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 they are being setup for CPU nodes. The GPUs (from multiple clusters) will be managed as a separate pool via FlexDirect. Figure 4 shows how the environment looks like after FlexDirect is installed. The control plane of kubernetes doesn't need to be FlexDirect aware (or no need for new configuration on integration of Kubernetes).

Fig 4. FlexDirect Integrated into Kubernetes Environment with GPU Cluster



Launching Kubernetes Pods with FlexDirect GPUs

The following outline the steps for dynamically attaching GPUs to a Kubernetes job at runtime much like storage, using FlexDirect, and 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. Installation of FlexDirect on GPU Servers

Install FlexDirect daemon on the GPU compute server hosts. Note that licensing is done based on hosts, so every time a new host machine is installed, you will have to run 'flexdirect init' to re-initialize the license.

```
wget -O - getflex.bitfusion.io | sudo bash
```

2. Health Check on GPU Servers

Once FlexDirect is installed on the GPU hosts, you can run the FlexDirect health check.

```
flexdirect health
```

*** Output of health check on a GPU server ***

```
Health report for server: 192.168.1.132:56001
=============================================
[PASS    ]      Check flexdirect install
[PASS    ]      Check library dependency
[PASS    ]      Check Network Errors/Drops: found no errors or packet drops
[PASS    ]      Check external connectivity to Bitfusion license server. Ignore for
                on-premise installations.
[PASS    ]      Check GPU API mismatch
[PASS    ]      Check CUDA version >= 7050: Current version: 9020
[PASS    ]      Check GPU Xid errors
[PASS    ]      Check GPU driver version >= 367.0: Current version: 396.37
[PASS    ]      Check temperature <= 100.00: current temp: 38.00
[PASS    ]      Check ECC errors
[PASS    ]      Check shadow memory 96563MB and total gpu mem 24331MB
[PASS    ]      Check mem ops
[PASS    ]      Check ulimit -n >= 4096: 4096
[MARGINAL]      Check MTU Size: 10000Mbps interface vetha9c24cf MTU 1500 < 4K:
                performance hit
[MARGINAL]      Check multinode support: RDMA not supported
[SKIPPED ]      Cannot perform PCIe diagnosis without root access. Run the binary with
                sudo to get diagnosis

PASS: 13
MARGINAL: 2
FATAL: 0
SKIPPED: 1
```

3. Starting the FlexDirect Resource Scheduler on GPU Servers

Start the flexdirect resource_scheduler on every GPU server hosts.

```
flexdirect resource_scheduler [-n NUM_GPUS] [-s listening_port] [-l listen_address] [-d
devices]
```

4. Installing FlexDirect on CPU Hosts

Install FlexDirect daemon on the CPU hosts. Note that licensing is done based on hosts, so every time a new CPU host machine is installed, you will have to run 'flexdirect init' to re-initialize the license. You are expected to install FlexDirect in every CPU server that will run the Kubernetes Pod.

```
wget –O – getflex.bitfusion.io | sudo bash
```

5. Configuring the servers.conf on CPU Hosts

On all the CPU hosts, create a /etc/bitfusionio/servers.conf file with the IP addresses (with optional ports) of all the flexdirect GPU servers. If you have some service discovery mechanism, this servers.conf can also be scripted to be auto populated.

Here are examples of servers.conf files.

```
K80_servers.conf
172.31.44
172.31.43
172.31.42

M60_servers.conf
173.22.12
173.22.13
173.22.14

v100_servers.conf
174.21.8
174.21.9
174.21.10
```

6. Mounting the FlexDirect Config and Binary Directories to Every Container

Every container that gets spun up as a kubernetes job needs to have the '/etc/bitfusionio/' directory mounted to it (by specifying in the job podspec), thereby having access to the flexdirect daemon. We also suggest that the host and the container have the same operating system that's support by Bitfusion FlexDirect (Ubuntu 16.04, Centos, Redhat for instance on both host and the container).

```
volumeMounts:

-name: bitfusionio

– mountPath: /etc/bifusionio
```

7. Running a Job with FlexDirect Health Check

One of the first Kubernetes jobs to run after setting your Kubernetes environment to run with FlexDirect as described above, is running the FlexDirect health check. This will run the health check across all GPU servers from the CPU host where the job is scheduled on. 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
    – |
      flexdirect health
```

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

```
Health report for server: 192.168.1.132:56001
=============================================
[PASS    ]      Check flexdirect install
[PASS    ]      Check library dependency
[PASS    ]      Check Network Errors/Drops: found no errors or packet drops
[PASS    ]      Check external connectivity to Bitfusion license server. Ignore for
                on-premise installations.
[PASS    ]      Check GPU API mismatch
[PASS    ]      Check CUDA version >= 7050: Current version: 9020
[PASS    ]      Check GPU Xid errors
[PASS    ]      Check GPU driver version >= 367.0: Current version: 396.37
[PASS    ]      Check temperature <= 100.00: current temp: 38.00
[PASS    ]      Check ECC errors
[PASS    ]      Check shadow memory 96563MB and total gpu mem 24331MB
[PASS    ]      Check mem ops
[PASS    ]      Check ulimit -n >= 4096: 4096
[MARGINAL]      Check MTU Size: 10000Mbps interface vetha9c24cf MTU 1500 < 4K:
                performance hit
[MARGINAL]      Check multinode support: RDMA not supported
[SKIPPED ]      Cannot perform PCIe diagnosis without root access. Run the binary with
                sudo to get diagnosis

PASS: 13
MARGINAL: 2
FATAL: 0
SKIPPED: 1


Health report for localhost
=============================================
[PASS    ]      Check flexdirect install
[PASS    ]      Check library dependency
[PASS    ]      Check Network Errors/Drops: found no errors or packet drops
[PASS    ]      Check external connectivity to Bitfusion license server. Ignore for
                on-premise installations.
[PASS    ]      Check GPU API mismatch
[PASS    ]      Check CUDA version >= 7050: Current version: 9020
[PASS    ]      Check GPU Xid errors
[PASS    ]      Check GPU driver version >= 367.0: Current version: 396.37
[PASS    ]      Check temperature <= 100.00: current temp: 38.00
[PASS    ]      Check ECC errors
[PASS    ]      Check shadow memory 96563MB and total gpu mem 24331MB
[PASS    ]      Check mem ops
[PASS    ]      Check ulimit -n >= 4096: 4096
[MARGINAL]      Check MTU Size: 10000Mbps interface vetha9c24cf MTU 1500 < 4K:
                performance hit
[MARGINAL]      Check multinode support: RDMA not supported
[SKIPPED ]      Cannot perform PCIe diagnosis without root access. Run the binary with
                sudo to get diagnosis

PASS: 13
MARGINAL: 2
FATAL: 0
SKIPPED: 1

All cuda versions are the same
```

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

FlexDirect 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.

8. Running a Job with Network-Attached FlexDirect GPUs

For every job that needs a GPU, the container args must be prepended with 'FlexDirect run –n <number of gpus> [-p <fraction of the gpu> -m <gpu memory needed>]' (as shown below) to dynamically attach full or fractional GPUs to the job as needed at runtime. Note that the container can also consume GPU without FlexDirect, simply by not specifying "flexdirect run …" .

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

      flexdirect 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 how the containers launched in Kubernetes with FlexDirect GPUs look like and Figure 6 summarizes the above steps in a flowchart.

Fig 5. Containers Launched in Kubernetes with FlexDirect

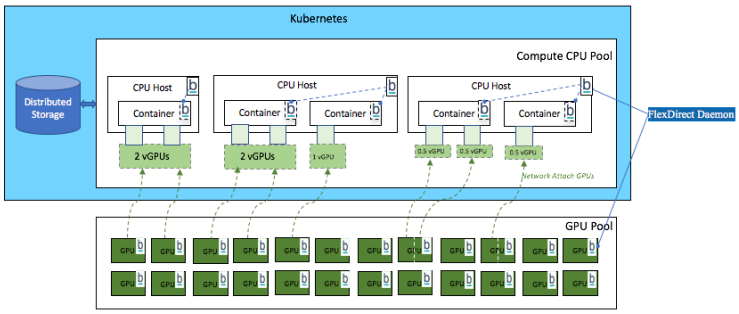


Fig 6. Flowchart of Enabling Network-Attached GPUs on Kubernetes with FlexDirect

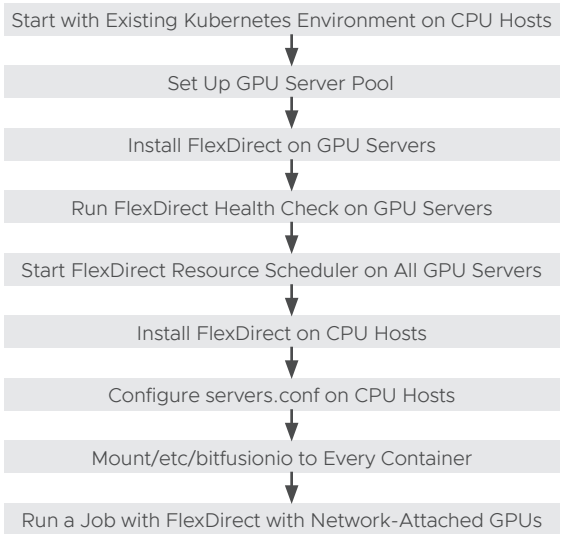


Figure 7 shows a sample client job podspec in Kubernetes to spin up a job to run a tf_cnn benchmark with 0.5 gpus using FlexDirect. Any of the FlexDirect run command options can be prepended with the actual application commands (in this case its python benchmarks/scrips/tf_cnn_benchmarks/tf_cnn_benchmarks.py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

Fig 7. Sample Kubernetes Podspec with FlexDirect GPUs

```
podSpec:
  volumes:
  - name: bitfusion
    hostPath:
      path: /etc/bitfusionio
  containers:
  - name: test
    args:
    - /bin/bash
    - -c
    - |
      flexdirect 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 \

    volumeMounts:
    - name: bitfusionio
      mountPath: /etc/bitfusionio
```

Conclusion

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

FlexDirect 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, take a look at the help documentation available at *https://www-review.vmware.com/solutions/business-critical-apps/hardwareaccelerators-virtualization.html*.