VMware vSphere Bitfusion Performance Best Practices Guide

Performance Best Practices - July 14, 2020

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com

Table of Contents

1. Introduction

VMware vSphere® Bitfusion® leverages its client-server architecture to share GPUs among multiple clients. Bitfusion dynamically allocates a GPU or multiple GPUs to a remote client and releases the GPU resources when they are no longer being used. Bitfusion works across AI frameworks, clouds, networks, and in environments such as virtual machines, containers, and notebooks.

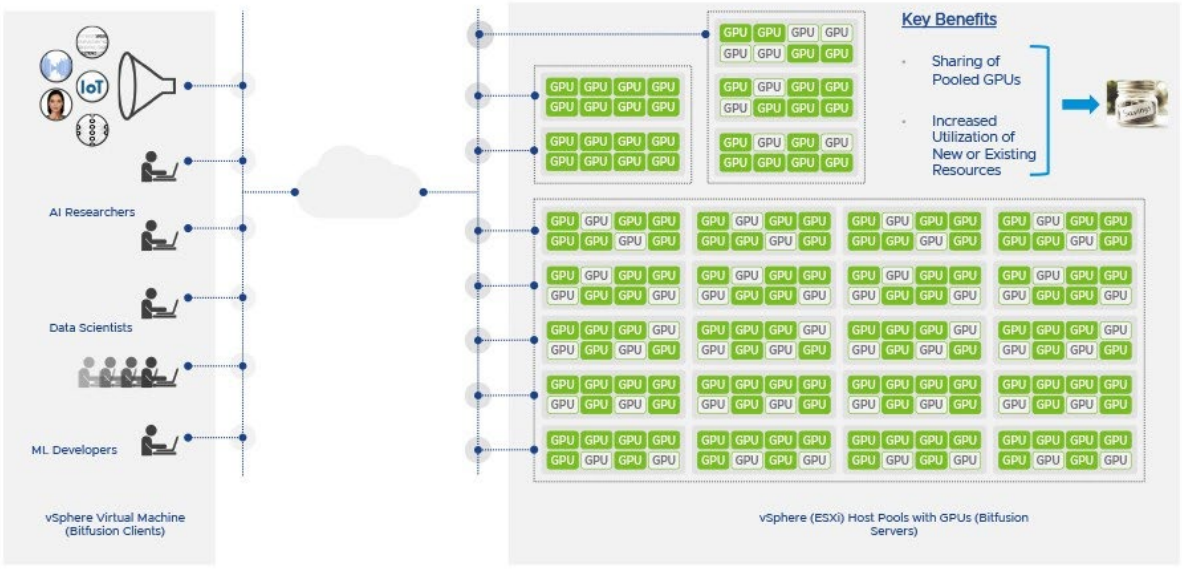

Figure 1. Diagram of the vSphere Bitfusion solution

2. GPU Hardware Considerations

Bitfusion is compatible with NVIDIA® data center GPUs Pascal and newer.

The recommendations in this guide are based on our performance testing using NVIDIA V100 (Volta architecture) GPUs with 16GB of graphics memory and NVIDIA T4 (Turing architecture) GPUs with 16GB of graphics memory. Our testbed configurations are in **table 1**, below.

Host	GPUs	Network Cards
Dell EMC DSS 8440	10x V100 Volta 16GB or 32GB	Mellanox ConnectX-5, Intel Ethernet Controller 10G X550
Dell PowerEdge R640	4x T4 Turing	Mellanox ConnectX-4, QLogic QL41xxx/1/20/25Gb
Dell PowerEdge C4140	4x V100 Volta 16GB or 32GB	Mellanox ConnectX-5, Intel Ethernet Controller 10G X550

Table 1. Hardware used in our testing

The V100 Volta GPU has 5120 CUDA cores, while the T4 Turing GPU has 2560 CUDA cores. Both GPUs are capable of machine learning (ML) training and inferencing. The training times are inversely proportional to the number of CUDA cores. That's why the V100 Volta results in better training times. Both GPUs are capable of mixed-precision arithmetic.

The T4 Turing GPU is capable of INT 4 operations, which significantly boost inferencing throughput. Another important consideration in choosing a GPU is power consumption. The T4 Turing GPU consumes 70 watts of power, while the V100 Volta consumes 300 watts.

3. Software Requirements

Table 2 lists the minimum software versions required to use Bitfusion.

Software Component	Version
VMware ESXi	7.0
VMware vCenter Server	7.0
vSphere Bitfusion server	7.0
vSphere Bitfusion client	6.7, 7.0
NVIDIA driver (server-side requirement)	NVIDIA-Linux-x86_64-440.64
CUDA (client-side requirement)	9.2-10.2
Guest operating systems	Ubuntu 18.04, Ubuntu 16.04, CentOS 7.0+, RHEL 7.4+

Table 2. Minimum software versions required to use Bitfusion

4. Network Hardware Considerations and Configuration Recommendations

Here, we describe network tunings that we used to achieve the best performance on our testbeds.

4.1. Network Hardware Considerations

After the GPU selection, the choice of networking hardware plays a significant role in determining the performance of your applications on vSphere Bitfusion.

For the best performance, the network adapters should support the following features:

- Checksum offload
- TCP segmentation offload (TSO)
- Ability to handle high-memory DMA (that is, 64-bit DMA addresses)
- Ability to handle multiple scatter/gather elements per Tx frame
- Jumbo frames (JF)
- Receive side scaling (RSS)

Our recommendations are based on performance studies on the hardware in our labs. You might need to tune the above parameters to get the best performance for your application on your hardware.

Table 3, below, shows the network hardware we used in our testbeds.

Speed	Protocol	Hardware Used in Our Setups
100Gb/s	PVRDMA/RoCE	Mellanox ConnectX-5 and ConnectX-4
40Gb/s	InfiniBand	Mellanox ConnectX-3 and ConnectX-3 Pro
10Gb/s	Ethernet	Intel Ethernet Controller 10G X550
10Gb/s	Ethernet	QLogic FastLinQ QL41xxx 1/10/25 GbE Ethernet Adapter

Table 3. Testbed networking hardware

In addition to the wire speed of your network hardware, and the features listed above, the device configuration can also affect performance. Many of these configuration options are addressed in the following sections.

4.2. Jumbo Frames

Jumbo frames, which are Ethernet frames with a payload size larger than 1500 bytes and less than 9000 bytes, can also improve network performance. A jumbo frame size of 4096 or larger was ideal in our experimentation. We encourage you to evaluate these settings in your testbed. Also see 4.6 DirectPath I/O (Passthrough) Networking Settings > Increasing Rx/Tx rings. You need to set jumbo frames in the guest operating system and on the switch.

4.3. Virtual Network Adapters or Passthrough

We studied three configurations for the VMs to connect to the physical NIC on the ESXi host:

- VMXNET3 - VMware's virtual NIC
- PVRDMA - VMware's solution to let you use RDMA
- Passthrough using Direct I/O

In the following sections, we provide information about each of these configurations.

Table 4, below, shows the CPU, throughput, and latency for the three network device configurations. For CPU and latency, lower is better. For throughput, higher is better.

Network Device Configuration	CPU	Throughput	Latency	Recommended Workload
VMXNET3	Highest usage	Lowest	Highest	Recommended for typical workloads like office workers completing tasks in MS Office, etc.
PVRDMA	Higher usage	High*	Low	Recommended for more complex workloads.
Passthrough (DirectPath I/O)	Lowest usage	Highest	Lowest	Recommended for high performance computing (HPC) and other very compute-intensive workloads.

Table 4. Network device configuration tradeoffs

** With PVRDMA, network traffic between VMs on the same host doesn't go through the physical NIC, so it might perform better than passthrough.*

Passthrough (using Direct I/O) achieved the best performance across all workloads, but it doesn't support vSphere virtualization features. For high performance computing workloads, use passthrough.

4.4. VMXNET3 Settings

VMXNET3 is a paravirtualized NIC. Its driver was designed to minimize I/O virtualization overhead. VMXNET3 is recommended for virtualized networking. If using the VMXNET3 NIC, we recommend the following network settings for the Bitfusion client and server.

Also see "Change the Virtual Machine Network Adapter" at https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.vm_admin.doc/GUID-3719A0BE-4B4A-44FF-8A21-290950918FBD.html.

4.4.1. Bitfusion Client VM VMXNET3 Settings

Our studies used QLogic FastLinQ QL41xxx 1/10/25 gigabit Ethernet adapters on the client side. **Table 5** shows the settings that provided the best performance in our test environment, but note that different hardware and workloads might perform best with other settings.

Parameter	Value	How to set the Value
disable_tpa	1	On the ESXi host: esxcfg-module -s "disable_tpa=1" qedentv; reboot
sched.cpu.latencySensitivity	High	vSphere Client
ethernet0.ctxPerDev	1	vSphere Client
ethernet0.pnicFeatures	1	vSphere Client
ethernet0.maxRxQueues	32	vSphere Client
sched.cpu.latencySensitivity.sysContexts	1	vSphere Client
/Net/CoalesceFineTimeoutCPU: 23	CPU_CORES_PER_NUMA_NODE - 1	vSphere Client
tx/rx ring parameters		Guest OS: ethtool -G <network interface> rx 4096 rx-mini 2048 rx-jumbo 4096 tx 4096
Large Receive Offload (LRO)	off	Guest OS: ethtool -K <network interface> lro off

Table 5. Bitfusion client VM settings for VMXNET3

4.4.2. Bitfusion Server VM VMXNET3 Settings

Our studies used Intel® Ethernet Controller 10G X550 Ethernet adapters. **Table 6** shows the settings that provided the best performance in our test environment, although different hardware and workloads might perform best with other settings.

Parameter	Value	How to set the Value
sched.cpu.latencySensitivity	High	vSphere Client
ethernet0.ctxPerDev	1	vSphere Client
ethernet0.maxRxQueues	32	vSphere Client
ethernet0.pnicFeatures	1	vSphere Client
sched.cpu.latencySensitivity.sysContexts	1	vSphere Client
/Net/CoalesceFineTimeoutCPU	CPU_CORES_PER_NUMA_NODE -1	vSphere Client
tx/rx ring parameters		Guest OS: ethtool -G <network interface> rx 4096 rx-mini 2048 rx-jumbo 4096 tx 4096
Large Receive Offload (LRO)	Off	Guest OS: ethtool -K <network interface> lro off

Table 6. Bitfusion server VM settings for VMXNET3

4.5 PVRDMA Settings

RDMA allows direct memory access from the memory of one computer to the memory of another computer without involving the operating system or CPU. The transfer of memory is offloaded to the RDMA-capable host channel adapter (HCA). A *PVRDMA* network adapter (the "PV" stands for paravirtual) provides remote direct memory access in a virtual environment.

We recommend the default settings for PVRDMA using virtual hardware version 17.

For more information on how to set up PVRDMA, refer to the following documentation:

- "Remote Direct Memory Access for Virtual Machines" at https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.networking.doc/GUID-9AADBB49-876E-4E44-8149-D0523D8ADA6A.html
- "How to Configure PVRDMA in VMware vSphere 6.5/6.7" at https://docs.mellanox.com/pages/releaseview.action?pageId=15055422

NOTE: As of hardware version 17, PVRDMA-to-native RDMA endpoint configurations are not supported.

4.6 DirectPath I/O (Passthrough) Networking Settings

In all workloads we tested, DirectPath I/O (passthrough) achieved the best performance because this configuration has no virtualization overhead—the network devices traverse the ESXi hypervisor and connect directly to the physical host. As mentioned earlier, passthrough doesn't support vSphere virtualization features.

For passthrough networking, refer to the guest operating system instructions for tuning network performance for features such as LRO. Examples follow.

- Disabling LRO

```
ethtool -K <network interface> lro off
```

- Increasing Rx/Tx rings

```
ethtool -G <network interface> rx 4096 rx-mini 2048 rx-jumbo 4096 tx 4096
```

Also refer to "DirectPath I/O" at https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.networking.doc/GUID-BF2770C3-39ED-4BC5-A8EF-77D55EFE924C.html.

5. Bitfusion OVA Default Settings

Bitfusion OVA defaults are set to maximize the virtualization benefits offered by the vSphere stack. Refer to the vSphere Bitfusion installation guide, which you can find at https://docs.vmware.com.

6. Other Important Tunables

In this section we list the BIOS, NUMA, and other hardware features that we tuned in the physical host to get the best performance in our testbed. We also include some tunables for the Bitfusion server and client VMs.

6.1. BIOS Settings

Table 7, below, shows the BIOS settings we used in the physical machine that hosts ESXi to gain the best performance in our Bitfusion testbed.

Tunable	Setting
System Profile	Performance
CPU Power Management	Maximum Performance
Memory Frequency	Maximum Performance

Tunable	Setting
Turbo Boost	Enabled
C1E	Disabled
C States	Disabled
Write Data CRC	Disabled
Memory Patrol Scrub	Standard
Memory Refresh Rate	1x
Uncore Frequency	Maximum
Energy Efficient Policy	Performance
Number of Turbo Boost Enabled Cores for Processor 1	All
Number of Turbo Boost Enabled Cores for Processor 2	All
Monitor/Mwait	Enabled
CPU Interconnect Bus Link Power Management	Disabled
PCI ASPM L1 Link Power Management	Disabled

Table 7. BIOS settings we used to gain maximum Bitfusion performance

6.1.1. BIOS NUMA Settings

Some NUMA-capable systems provide an option in the BIOS to disable NUMA by enabling node interleaving. In most cases, don't disable it.

You should enable hyperthreading.

6.2. ESXi CPU Considerations

The best performance can usually be obtained if each ESXi host runs either Bitfusion server VMs or Bitfusion client VMs, but not both. However, combining Bitfusion server and client VMs on the same host is supported, and can perform well for some workloads because it can reduce networking overhead.

In most environments, ESXi allows significant levels of CPU overcommitment (that is, running more vCPUs on a host than the total number of physical processor cores in that host) without impacting virtual machine performance. If an ESXi host becomes CPU saturated (that is, the virtual machines and other loads on the host demand all the CPU resources the host has), latency-sensitive workloads might not perform well. In this case, you might want to reduce the CPU load—for example, by powering off some virtual machines or migrating them to a different host (or allowing DRS to migrate them automatically). Because Bitfusion servers use GPUs in passthrough mode, they can't be vMotioned. However, other non-GPU VMs, including Bitfusion client VMs, can be migrated to other hosts.

Using esxtop or resxtop, you should monitor the CPU load of the hosts running Bitfusion client and server VMs. A load average of the first line of the esxtop CPU panel equal to or greater than 1 indicates that the CPU is overloaded. In general, 80% load of the physical CPU is an upper bound with some room for periodic spikes in CPU load.

6.3. Bitfusion Server and Client VM Sizing

For a Bitfusion server running on a dedicated host, most of the memory on the host should be allocated to the Bitfusion server. If the Bitfusion server and clients are sharing hosts with other VMs, follow the guidelines listed below.

6.3.1. Memory Sizing

Bitfusion leverages GPUs. Therefore, the amount of graphics memory on all the GPU cards determines the minimum memory requirement for the Bitfusion server and client. The minimum memory allocated to the Bitfusion server VM should be 1.5 times the single largest graphics memory of all the GPU cards attached to the Bitfusion server. Even though the Bitfusion client does not have GPUs attached to it, it still requires the minimum memory allocated that is equal to the 1.5 times the graphics memory on **all** the GPUs that the client workloads will be requesting. You'll need to add to this the memory required for your guest operating system and its applications. **Table 8**, below, shows the formulas to obtain the minimum memory requirements.

Server/Client	Minimum Memory Requirement
Bitfusion Server	1.5 * aggregate of all GPU memory on all GPU cards + minimum memory for your application
Bitfusion Client	1.5 * graphics memory on the requested GPUs + minimum memory for your application

Table 8. Minimum memory requirements

6.3.2. Allocating vCPUs to Bitfusion Client and Bitfusion Server VMs

Even though Bitfusion workloads are GPU-intensive, allocating an optimal number of virtual CPUs (vCPUs) can improve the performance.

Configuring a virtual machine with more vCPUs than its workload can use might cause slightly increased resource usage, potentially impacting performance on very heavily loaded systems. Common examples of this include a single-threaded workload running in a multiple-vCPU virtual machine or a multi-threaded workload in a virtual machine with more vCPUs than the workload can effectively use. Even if the guest operating system doesn't use some of its vCPUs, configuring virtual machines with those vCPUs still imposes some small resource requirements on ESXi that translate to real CPU consumption on the host.

Based on our performance studies, Bitfusion client and server VMs require a minimum of 4 vCPUs. You might need to increase the number of vCPUs depending on the number of GPUs.

6.3.3. NUMA Settings

For multi-socket host systems, NUMA settings have a considerable impact on performance. The location of the NIC is also an important factor for performance. If possible, the GPU, the NIC, and the Bitfusion server should all be on the same NUMA node. If that configuration isn't physically possible, then placing the GPU and the Bitfusion server on the same NUMA node should take priority over the NIC.

6.3.3.1. GPU PCIe Socket, Network Adapter PCIe Socket, and NUMA Affinity Settings in Bitfusion Server VM

GPU cards and NICs are placed in PCIe physical slots. Each physical slot is associated with a particular NUMA node on the host. Refer to **figure 2**, on the next page. The VM using the GPU with PCI address 0000:3b:0000 is associated with NUMA Node 0. However, the VM's vCPUs are associated with NUMA Node 1 on the GPU with the PCI address 0000:d8:0000. The CPU transfers the data first from the storage to its main memory and then to high speed memory on the GPU card.

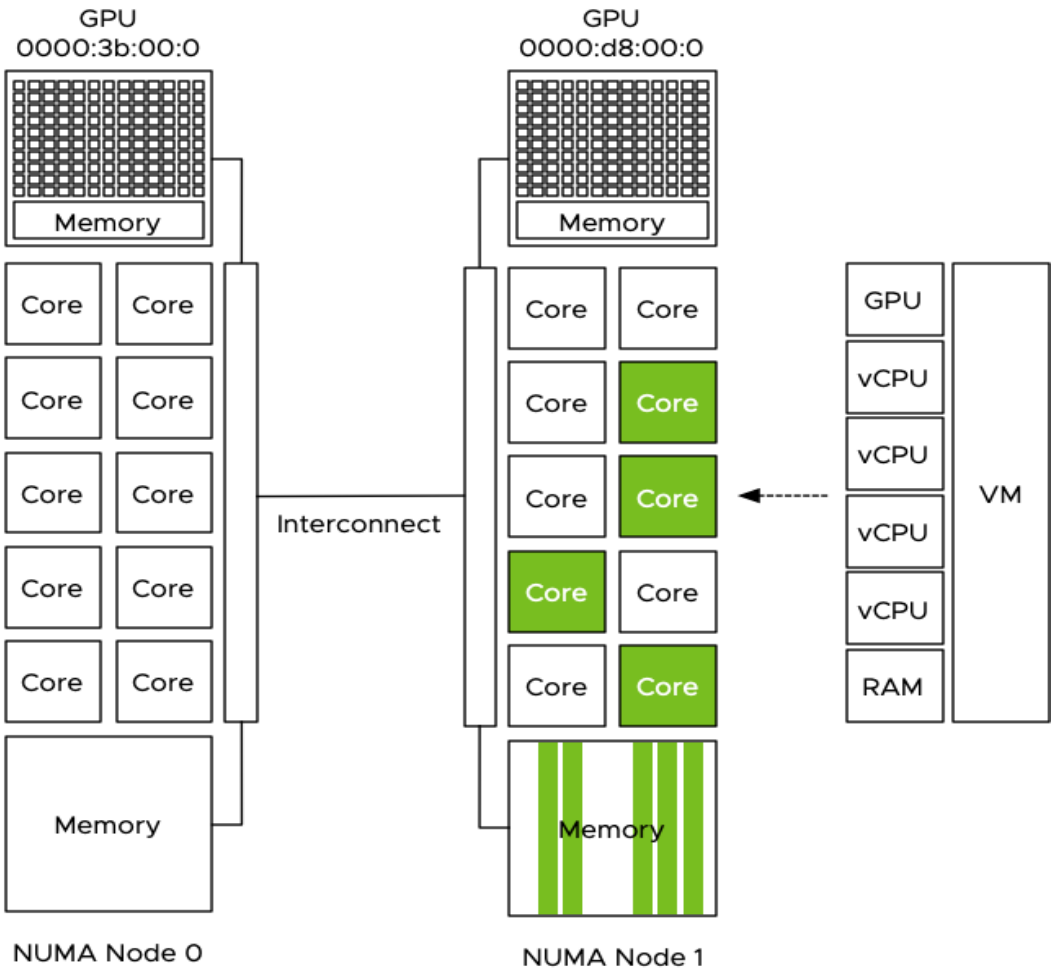

Figure 2. A GPU mismatch occurs because the vCPUs are associated with GPU 000:d8:00:0 instead of 000:3b:00:0

You can use the VMkernel system information shell (vsish) utility on the ESXi host running the Bitfusion server VM to find out which NUMA node the GPU card is associated with. The **vsish** command listed below requires the GPU card's PCI address (bus, device, and function) in decimal format. Use the **lspci** command on an ESXi host to get these values.

```
vsish> cat
/hardware/pci/seg/0/bus/<pci_bdf_address_of_gpucard_in_decimal>/slot/0/func/0/pciConfigHeader #
lists the numa_node.
```

We recommend setting the Bitfusion server VM's **numa.nodeAffinity** to the NUMA node associated with the GPU card, which you can change in Advanced VM options.

The NVIDIA system management interface (nvidia-smi) utility provides monitoring and management capabilities for each of the installed GPUs. The command **nvidia-smi -q** will display all the information about GPUs. The command **nvidia-smi topo -mp** will give information about CPU affinity of each of the GPUs.

```
$ nvidia-smi topo -mp
        GPU0    GPU1    GPU2    GPU3    GPU4    GPU5    GPU6    GPU7    GPU8    GPU9    CPU Affinity
GPU0     X      PHB     PHB     PHB     PHB     PHB     PHB     PHB     PHB     PHB     0-11
GPU1    PHB      X      PHB     PHB     PHB     PHB     PHB     PHB     PHB     PHB     0-11
GPU2    PHB     PHB      X      PHB     PHB     PHB     PHB     PHB     PHB     PHB     0-11
GPU3    PHB     PHB     PHB      X      PHB     PHB     PHB     PHB     PHB     PHB     0-11
GPU4    PHB     PHB     PHB     PHB      X      PHB     PHB     PHB     PHB     PHB     0-11
GPU5    PHB     PHB     PHB     PHB     PHB      X      PHB     PHB     PHB     PHB     0-11
GPU6    PHB     PHB     PHB     PHB     PHB     PHB      X      PHB     PHB     PHB     0-11
GPU7    PHB     PHB     PHB     PHB     PHB     PHB     PHB      X      PHB     PHB     0-11
GPU8    PHB     PHB     PHB     PHB     PHB     PHB     PHB     PHB      X      PHB     0-11
GPU9    PHB     PHB     PHB     PHB     PHB     PHB     PHB     PHB     PHB      X      0-11
```

Legend:

- **X** = Self
- **SYS** = Connection traversing PCIe as well as the SMP interconnect between NUMA nodes (e.g., QPI/UPI)
- **NODE** = Connection traversing PCIe as well as the interconnect between PCIe Host Bridges within a NUMA node
- **PHB** = Connection traversing PCIe as well as a PCIe Host Bridge (typically the CPU)
- **PXB** = Connection traversing multiple PCIe bridges (without traversing the PCIe Host Bridge)
- **PIX** = Connection traversing at most a single PCIe bridge

6.3.3.2. NUMA Settings for Bitfusion Client and Server VMs

The default Bitfusion server and client VMs are configured with 12 vCPUs (numvcpus). The setting of **cpuid.coresPerSocket = 1** (one core per socket) defines the virtual NUMA topology of the virtual machine. All 12 virtual sockets are grouped into a single physical domain, which means that the vCPUs will be scheduled in a single physical CPU package that typically is similar to a single physical NUMA node.

The NUMA scheduler in ESXi autosizes the vNUMA client. For details, refer to https://frankdenneman.nl/2016/08/22/numa-deep-dive-part-5-esxi-vmkernel-numa-constructs/.

During the initial boot, the VMkernel adds two advanced settings to the virtual machine:

```
numa.autosize.vcpu.maxPerVirtualNode=X
numa.autosize.cookie = "XXXXXX"
```

The **autosize** setting reflects the number of vCPUs inside the NUMA node. Do not change this setting unless the number of vCPUs of the VM changes. This is particularly of interest for clusters that contain heterogeneous host configurations. If your cluster contains hosts with different core counts, you could end up with a NUMA misalignment. In this scenario, the following advanced settings can be used:

```
numa.autosize.once = FALSE
numa.autosize = TRUE
```

This forces the NUMA scheduler to reconfigure the NUMA clients at every power cycle.

7. Partial GPUs

The ability to dynamically share GPUs among multiple clients distinguishes Bitfusion from other GPU sharing solutions. Bitfusion allows GPU memory to be subdivided on arbitrary boundaries. For example, **-p 0.23** in the following command will use only 23% of the GPU memory, thus leaving the remaining 77% of the GPU memory to be shared with one or more other Bitfusion clients.

```
% bitfusion run -n 1 -p 0.23 -l 10.196.151.243:56001 -s servers.conf -- time python -B ./main.py
--cuda --epochs 1 --batch_size 512 --save /tmp/model.pt
```

The guidelines for scaling the number of concurrent Bitfusion clients sharing a GPU are empirical:

- Determine the minimum amount of memory needed by each client.
- Determine how many concurrent clients can fit in the GPU memory and still achieve acceptable throughput and latency.
- Divide any remaining memory among the clients; many applications are elastic in their memory use and might achieve higher performance, or work with larger batch sizes, if more memory is available.

8. Bitfusion Health Check Utility

Bitfusion has a health check command that checks and reports on Bitfusion cluster systems, networks for configuration settings, etc. The report has a **Performance Check** section of which we'll discuss three tests. For other questions on health checks, please ask for the latest health check documentation.

```
Performance checks:
=======================
[PASS ] Check Interface/Subnet Compatibility: network interfaces and subnets configured correctly
[PASS ] Check ulimit -n >= 4096: 4096
[MARGINAL] Check MTU Size: 10000Mbps interface ens160 has low MTU: 1500 < 4K
```

- **Interface/Subnet Compatibility Check –** Bitfusion performance relies heavily on a healthy, low-latency, high-speed network. This check determines if the network interfaces sharing a subnet have compatible settings. Specifically, it will:
  - Issue a FATAL condition if Ethernet and InfiniBand interfaces share the same subnet
  - Issue a MARGINAL condition if interfaces with different speeds share the same subnet
- **Check ulimit –** Bitfusion may require many open file descriptors to perform well. This check looks at your Linux user limit for open descriptors and warns you if it is less than 4096.
- **MTU Size Check –** Bitfusion performance relies heavily on a healthy, low-latency, high-speed network. Because you pay a latency penalty with every packet sent over the network, you should send a few large packets instead of many small packets. This check determines if you have a large (≥4K) MTU (maximum transfer unit) setting for all high-speed (≥10 Gbps) interfaces. You can ignore this check for interfaces that you won't use with Bitfusion.