WHITE PAPER–OCTOBER 2019

FlexDirect Health Checks

Bitfusion Guides

vmware®

Table of Contents

FlexDirect Health Checks

FlexDirect has a health check command that examines cluster systems and networks for configuration settings, networking error count, library installations and other factors that can help to diagnose if and how well FlexDirect will perform.

This document explains each check. But it begins with a look at the command itself.

Health Command

There are actually two health check commands: local only and cluster. The following performs health checks on the local server only:

Local Health Check

```
flexdirect localhealth
```

You can also use localhealth alias: `flexdirect LH`.

This next command performs health checks on all the GPU servers.conf as well as on the local server. It requires that the local configuration file servers.conf have the IP addresses of the GPU servers and that those servers are running a FlexDirect service (either SRS or FDM), see the *Installation* guide for details on configuration and running the services.

Cluster Health Check

```
flexdirect health

# Or use the 'H' alias
flexdirect H
```

Software Versions Checks

The following checks detect software versions not supported by FlexDirect. Here is some sample output:

Sample Software Versions Checks Output

```
Software Versions checks:
=========================
[PASS    ]      Check library dependency
[PASS    ]      Check OFED Version: 4.3
[PASS    ]      Check CUDA version >= 7050: Current version: 9010
[PASS    ]      Check GPU driver version >= 367.0: Current version: 390.30
```

Now, we will look at the individual checks.

Library Dependency Check

FlexDirect depends on some third-party libraries, such as libprocps.so and libssl.so. This check will list all such dependencies that have not been installed

Please install any missing dependencies before running any further FlexDirect commands.

OFED Version Check

FlexDirect can use various network transports, such as TCP, rsocket and Infinibad. But as high-speed interfaces are required for effective virtualization (≥ 10 Gbps), many customers use interface cards compliant with OpenFabrics Enterprise Distribution (OFED) software overseen by the OpenFabrics Alliance. Mellanox adaptors are a common example. OFED is a set of drivers and software for such interface cards and FlexDirect will work with version 3.3 and higher. This check will tell you if your version of OFED is lower than this

You may check the version of OFED yourself with the `ofed_info` command. This command output is quite lengthy, but the version information is at the top. The formatting of the version information changes from time to time, so it is possible that you may see an error reported by this check, even if you have a version greater than 3.3. If that happens, ignore this error. You can also ignore errors from this check if you are not using OFED.

Please update your OFED software (see your vendor) if necessary.

CUDA Version Check

FlexDirect supports CUDA from version 7.5 and above. This check will tell you if your version of CUDA is supported or not.

Often you can look in the /usr/local directory and tell by the cuda directory name what version you have. Or you can run `ldconfig -p | grep cuda` to see where your CUDA libraries are.

Please update your CUDA installation (often the version will be dictated by the application you are running).

GPU Driver Version Check

FlexDirect supports NVIDIA GPU drivers from version 367.0 and higher. This check will tell you if your driver version is supported or not.

You can see the version of your driver in the output of `nividia-smi`.

Choose and install an up-to-date GPU driver as recommended by the NVIDIA website based on your GPU and the version of CUDA you need.

System Resources Checks

The system resources checks look for adequate memory and at FlexDirect operability itself. Here is some sample output:

Sample System Resources Checks Output

```
System Resources checks:
========================
[PASS    ]      Check flexdirect install
[PASS    ]      Check external connectivity to Bitfusion license server. Ignore for
                on-premise installations.
[PASS    ]      Check shadow memory 191947MB and total gpu mem 32560MB
```

Now, we will look at the individual checks.

FlexDirect Install Check

If you are running the health command, you obviously have installed some of the FlexDirect files. But this check will look for all files that should be there in a complete install. It also looks for a valid license.

If this does not pass, reinstall FlexDirect and check or reinitialize your license.

Checkin FlexDirect License

```
# Check if you have a valid license.
flexdirect license

# If necessary, initialize your license.
flexdirect init  # and follow the prompts
```

External License Server Connectivity Check

Unless otherwise agreed to with BitFusion, FlexDirect licenses are served from an online URL. This check sees if you can connect to that URL.

You can ignore this check if you have made proper arrangements with Bitfusion for an on-premise (offline) installation.

Shadow Memory Check

For GPU virtualization to work, the client machine makes use of buffers in its own memory to shadow the memory on the physical, but remote, GPUs. The precise amount of memory required on the client host can vary from application to application, but a reasonable rule-of-thumb is to have at least as much host memory as there is GPU memory. This check sees if that is the case.

You can see the amount of memory on your client server from the MemTotal line of the pseudo file /proc/meminfo. To calculate the GPU memory, from a GPU server run nvidia-smi, and add up the memory sizes from all the GPUs.

You may need to add more memory to your client to pass this test, or when you run applications make sure you do not allocate more GPUs than you can shadow on the client.

Performance Checks

The performance checks look for settings and library installations that can effect the performance of virtualized GPUs. Here is some sample output:

Sample Performance Checks Output

```
Performance checks:
=====================
[PASS    ]      Check Interface/Subnet Compatibility: network interfaces and subnets
                configured correctly
[PASS    ]      Check MTU Size: hi-speed interfaces MTU >= 4K
[PASS    ]      Check ulimit -n >= 4096: 4096
[MARGINAL]      Check PCIe Width:Bus 01:00.0 Ethernet controller: Intel Corporation I350
                Gigabit Network Connection (rev 01) has 4 lanes, only using 2

                Bus 01:00.1 Ethernet controller: Intel Corporation I350 Gigabit Network
                Connection (rev 01) has 4 lanes, only using 2

[MARGINAL]      Check multinode support: GPU direct not supported

[MARGINAL]      Check nv_peer_mem: nv_peer_mem module not loaded
```

Now, we will look at the individual checks.

Interface/Subnet Compatibility Check

FlexDirect 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 type interfaces and IB type interfaces share the same subnet
- Issue a MARGINAL condition if interfaces with different speeds share the same subnet

You see these settings yourself:

Network Settings: Type, Speed, and Subnets

```
$ # Each interface has its own subdirectory here:
$ ls /sys/class/net
eth0   enp175s0 enp175s0d1

$ # Speed in Mbits per second - here we look at a 10Gbps interface:
$ cat /sys/class/net/enp175s0
10000

# Type - 1 is ethernet; 32 is IB - here we look at an ethernet interface:
$ cat /sys/class/net/enp175s0
1

# Perform a logical AND on the IP address and netmask to see the subnet
$ ifconfig
...
enp175s0  Link encap:Ethernet  HWaddr 00:02:c9:20:cc:00
          inet addr:10.10.10.42  Bcast:10.10.10.255  Mask:255.255.255.0
...
# subnet 10.10.10.0  <=  10.10.10.42 AND 255.255.255.0   (in binary)
```

You should fix any incompatibilities found.

MTU Size Check

FlexDirect performance relies heavily on a healthy, low-latency, high-speed network. Because you pay a latency penalty with every packet you send over the network, you should send a few large packets instead of many small packets. This checks determines if you have a large (≥4K) MTU setting for all high-speed (≥10 Gbps) interfaces. MTU stands for Maximum Transfer Unit. You can ignore this check for interfaces you will not use with FlexDirect.

You can both get and set the MTU as shown here:

Network Settings: Type, Speed and Subnets

```
$ # Check the MTU size of my interfaces
$ ifconfig
...
enp175s0  Link encap:Ethernet  HWaddr 00:02:c9:20:cc:70
          inet addr:10.10.10.43  Bcast:10.10.10.255  Mask:255.255.255.0
          inet6 addr: fe80::202:c9ff:fe20:cc70/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9000  Metric:1
...

$ # Change MTU on interface enp175s from 1500 to 9000 (bytes in an ethernet frame)
$ sudo ifconfig enp175s0 mtu 9000
```

Set the MTU to a value between 4096 and 9000, inclusive, for all relevant interfaces.

ulimit Check

FlexDirect may require a large number of open file descriptors to perform well. This check looks at at your Linux user limit for open descriptors and warns you if it is less than 4096.

You can check and set the limit as shown here:

Open File Descriptors User Limit

```
$ # Check the MTU size of my interfaces
$ ifconfig
...
enp175s0  Link encap:Ethernet  HWaddr 00:02:c9:20:cc:70
          inet addr:10.10.10.43  Bcast:10.10.10.255  Mask:255.255.255.0
          inet6 addr: fe80::202:c9ff:fe20:cc70/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9000  Metric:1
...

$ # Change MTU on interface enp175s from 1500 to 9000 (bytes in an ethernet frame)
$ sudo ifconfig enp175s0 mtu 9000
```

FlexDirect may change this limit for itself when it launches, avoiding the need for you to worry about it.

PCIe Width Check

FlexDirect and the applications that use it for GPU virtualization will access several devices on the PCIe bus: interfaces, GPUs, and IB host bus adaptors. The bandwidth to these devices can affect performance. This check tells you if these devices are using fewer PCIe bus lanes than capacity permits.

You can see the width capacity and current width as shown below:

PCIe Width

```
$ # This command will produce a lot of output, but it will include what you want.
$ # We'll extract examples of the lines you want, preceded by a comment
$ sudo lspci -vvv
...
# Look for blocks with the words, "nvidia", "graphics", "network", or "mellanox" (case
# insensitive).
# Here is the start of a block for a GPU.
d8:00.0 3D controller: NVIDIA Corporation Device 1db4 (rev a1)
...
# Here is the line telling you the width capacity of the GPU - 16 lanes.
                LnkCap:  Port #0, Speed 8GT/s, Width x16, ASPM not supported, Exit
Latency L0s <1us, L1 <4us
                        ClockPM+ Surprise- LLActRep- BwNot- ASPMOptComp+
...
# Here is the line telling you the current width - 16 lanes.
                LnkSta:  Speed 8GT/s, Width x16, TrErr- Train- SlotClk+ DLActive-
BWMgmt- ABWMgmt-
...
# This device is using all 16 lanes in its capacity, so the check will pass.
```

Change your device settings to use their full capacity.

Multinode Support Check

This is a conglomeration of several checks which will issue multiple errors if any does not pass, but will issue a single PASS message if all pass.

- Is RDMA supported
  - Do RDMA devices exist - Is `/sys/class/infiniband`
  - Are the error counts on RDMA devices all zero (e.g., some of the files here) `/sys/class/infiniband/mlxX_X/ports/*/counters`
  - Does the device have any valid GIDs -(any files in) `/sys/class/infiniband/mlxX_X/ports/*/gids` have values that are NOT `0000:0000:0000:0000:0000:0000:0000:0000` or `fe80:0000:0000:0000:0000:0000:0000:0000`
- Are the Mellanox devices near enough to the GPU devices in the PCIe tree to allow GPU-to-GPU communication across the servers in a virtual environment? The command `nvidia-smi topo -m` prints a matrix of GPU and Mellanox adaptors. If any values in the Mellanox rows do not belong to the set {'X', 'PIX', PXB', 'NV'}, they are not near enough.

NV Peer Memory Check

Fast communication between GPUs in different on separate servers is extremely critical for applications using such a configuration (multinode). You should use Mellanox's nv_peer_mem kernel module, which relies in turn on NVIDIA's GPUDirect functionality. GPUDirect allows the Mellanox adaptors to directly access GPU memory (over PCIe), and then extends that access over an RDMA connection to the Mellanox adapter on a remote server. That remote adaptor can again use GPUDirect to access its local GPU. This check tests if the nv_peer_mem library is installed and usable (if the Mellanox adaptors can register host and device memory).

You can ignore this check until you have consulted with Bitfusion about the viability and necessity of using multinode on your cluster.

Test for nv_peer_mem and Installation

```
# Check for the nv_peer_mem kernel module.
lsmod | grep nv_peer_mem

# To install the module - the Mellanox driver is a prerequisite.
git clone https://github.com/Mellanox/nv_peer_memory.git
cd nv_peer_memory
make
sudo insmod nv_peer_mem.ko

# Follow the instructions in the above repo's README.md file to have this module.
# install at boot time
```

Stream Mem Ops Check

The CUDA Memory Operations API is an NVIDIA low-latency CPU-GPU memory synchronization library that is necessary for best performance. This check tests if the API is enabled. The check is only made on GPU servers.

To check for yourself, see that `EnableStreamMemOPs` is set to `1` in pseudo file, `/proc/driver/nvidia/params`.

The error message is shown below along with steps to enable the API.

CUDA Memory Operations Enabled

```
# Example Stream Mem Ops check output
[MARGINAL]  Check mem ops: Device 0 (Tesla V100-SXM2-16GB) does not support stream
memory operations.
  please try reloading nvidia driver with `modprobe nvidia NVreg_EnableStreamMemOPs=1`

# To enable the API immediately (but not configuring for boot time enablement):

sudo lsmod | grep nvidia  # to list all the nvidia modules
sudo rmmod nvidia_uvm nvidia_drm nvidia_modeset nvidia # stopping all nvidia modules
sudo modprobe nvidia NVreg_EnableStreamMemOPs=1

# Example to enable API at boot (you will need to know the
# full name of the NVIDIA module, not just the 'NVIDIA' alias.
# The full name is based on the driver major release number.
# The example name we use here is nvidia_390. At some higher
# release numbers, the full name is just "NVIDIA". You will likely
# see this full name already used in some *.conf file).
# You need to add a line to a module configuration file.

# On Ubuntu:
# The file /etc/modprobe.d/nvidia-graphics-drivers.conf is
# a reasonable choice.

# On RHEL/CentOS
# The file /lib/modprobe.d/nvidia.conf is a reasonable choice.

# You can also create your own file, e.g., nvidia.conf.
# The options in this file take the format:
# options <module name> <option-name>=<option-val>
#
# Add this option and value at the end of the file:
# NVreg_EnableStreamMemOPs=1
#
cat /etc/modprobe.d/nvidia-graphics-drivers.conf
...snip...
options  nvidia_390 NVreg_EnableStreamMemOPs=1

# Now update the initramfs (maybe) and reboot:

# On Ubuntu
sudo update-initramfs -u
reboot

# On RHEL and CentOS, you may need to update the initramfs with dracut
# sudo dracut /boot/initramfs-$(uname -r).img $(uname -r)
# But skip the dracut command and see if a simple reboot alone is enough.
reboot
```

Stability Checks

The stability checks look for conditions, errors and settings that cause your system to fail. Here is some sample output:

Sample Stability Checks Output

```
Stability checks:
=================
[PASS    ]  Check PCIe P2P support: PCIe p2p supported
[PASS    ]  Check IB physically up
[PASS    ]  Check IB SM up
[PASS    ]  Check MAD agent registration errors
[PASS    ]  Check GPU API mismatch
[PASS    ]  Check GPU Xid errors
[PASS    ]  Check temperature <= 100.00: current temp: 30.00
[PASS    ]  Check ECC errors
[MARGINAL]  Check Network Errors/Drops:     drops reported in file: /sys/class/net/eno1/
                                            statistics/rx_dropped

                                       :    drops reported in file: /sys/class/net/
                                            enp94s0/statistics/rx_dropped

[MARGINAL]  Check RDMA: TX pause parameter is off
                : RX pause parameter is off
                : TX pause parameter is off
                : RX pause parameter is off
```

Now, we will look at the individual checks.

PCIe P2P Support Check

If you are going to run applications using multiple GPUs you will require fast PCIe peer-to-peer traffic. Fast peer-to-peer traffic requires you to disable ACS (Access Control Services) on PCIe switches that support it. This check issues a FATAL message if ACS is enabled.

You can check for enabled ACS (FATAL setting) and disable it yourself as shown here:

Is ACS Disabled for fast P2P

```
# Run lscpi and search output for ACS capability and if it is set.
# (If is is set then fast peer-to-peer communication is not possible. This
# is considered a FATAL condition for FlexDirect.)
sudo lspci -vv | grep ACSCtl | grep SrcValid+

# If you get output with 'SrcValid+' ('SrcValid-' means the feature is off),
# e.g., "ACSCtl: SrcValid+" for the bridges or other devices, then turn the feature
# off (you'll have to printout the whole output find the lines, scroll up to get
# the bus#, slot#, and func# of each device) by running:
setpci -s bus#:slot#.func# f2a.w=0000

# Then rerun the lspci command above to make sure  to ACSCtl on the devices is now off.
```

IB Physically Up Check

This check sees if the IB links are physically up.

You can check this youself as shown below:

IB Physically Up

```
$ # Check on the state of the IB devices.
$ # Have filtered the output to show Physical state is "LinkUp"
$ ibstat
...
CA 'mlx4_0'
...
        Port 1:
...
                Physical state: LinkUp
```

IB SM Up Check

In clusters using Infiniband (IB), its subnet manager (SM) must be running. This check sees if the IB devices are active.

You can check this yourself as shown below:

IB SM: Are Devices Active

```
$ # Check on the state of the IB devices.
$ # Have filtered the output to show active device
$ ibstat
...
CA 'mlx4_0'
...
        Port 1:
                State: Active
...
```

You can check this yourself as shown here:

MAD Agent Registration Errors Check

Infiniband has a subnet manager that communicates with an agent running on every node. The format of these communciations is the Management Datagram (MAD). This check tells you if there have been been errors with agent registration (and that the subnet is not healthy).

You can search for these errors yourself with `dmesg`:

MAD Agent Registration Errors

```
# Check for MAD agent registration error messages (-T for human-readable timestamps).
dmesg -T | grep “ib_register_mad_agent: QP [0-9]+ not supported”
```

GPU API Mismatch Check

This check tells if there are version mismatches between your NVIDIA kernel module and your NVIDIA APIs.

You can search for these errors yourself with `dmesg`:

GPU API Mismatch

```
# Check for GPU API mismatch error messages (-T for human-readable timestamps).
dmesg -T | grep “NVRM: API mismatch”
```

Examine the entire set of NVIDIA (NVRM) output from `dmesg`, if this check finds a problem and address all issues.

GPU Xid Check

The NVIDIA driver will report several types of errors to the system log. This check will alert you if any such messages exist.

You can search for these errors yourself with `dmesg`:

Text

```
# Check for GPU Xid error messages (-T for human-readable timestamps)
dmesg -T | grep “NVRM: Xid”
```

Examine the NVRM: Xid”output from dmesg, using the NVIDIA’s *website* to explain the meaning and resolution of any issues.

Temperature Check

This check tells you if any of the GPUs are running too hot (over 100 degrees Celsius).

You can see this temperature yourself as shown below:

Check GPU Termperature

```
# Print out the temperature (Celsius) of all GPUs.
nvidia-smi --query-gpu=temperature.gpu --format=csv,noheader
```

ECC Errors Check

The GPUs count ECC errors. This checks tells you if any counts are non-zero.

You can see these counts as shown below:

Shell

```
# Print out the ECC counts of your GPUs.
$ nvidia-smi --query-gpu=ecc.errors.corrected.volatile.total,ecc.errors.corrected.
aggregate.total,ecc.errors.uncorrected.volatile.total,ecc.errors.uncorrected.aggregate.
total --format=csv,noheader

# The output is shown here from a system with two GPUs. The first of these GPUs has
# ECC disabled.  The second has ECC error counts of 0.
[Not Supported], [Not Supported], [Not Supported], [Not Supported]
0, 0, 0, 0
```

You can also run `nvidia-smi -q` to see a more verbose report that includes the ECC error counts.

Network Errors Check

Network interfaces collect error statistics and dropped packet counts. These files can be found here:

- `/sys/class/net/<interface>/statistics/*errors`
- `/sys/class/net/<interface>/statistics/*dropped`

This check will tell if there are errors or dropped packets.

Pause Parameter Check

RDMA over Converged Ethernet (RoCE) requires specific network settings so that no packets are lost or retransmitted. If you are using RDMA transports, you should set the TX and RX pause parameter on. You can ignore this health check for interfaces you will not use at all or will not use with RoCE.

To turn on pause parameters on, first get the list of your interfaces with `ifconfig` and then use `ethtool` on relevant interfaces, as shown below, to view and set the parameters.

Setting Pause Parameters

```
ifconfig  # list the interfaces so you can use them by name with ethtool - <interface>

ethtool -a <interface>
ethtool -A <interface> <tx|rx> on
  # The following example shows an interface (en0) with
  # TX pause parameters off and then how to use ethtool
  # to set them on. The same can be done for RX:
$ ethtool -a eno1
Pause parameters for eno1:
Autonegotiate:  off
RX:             on
TX:             off
$ ethtool -A eno1 tx on
$ ethtool -a eno1
Pause parameters for eno1:
Autonegotiate:  off
RX:             on
TX:             on
```

Consistency Checks

This section will give the total checks in the PASS, MARGINAL and FATAL categories. It also prints out the version information of several libraries. Here is some sample output:

Text

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

PASS: 18
MARGINAL: 5
FATAL: 0
nvmlSystemGetNVMLVersion: 9.390.30
nvmlSystemGetDriverVersion: 390.30
CUDA device capability Major/Minor version number: 6.0
cudriverVersion: 9010
cudaDriverGetVersion: 9010
cudaRuntimeGetVersion: 8000
cudnnGetVersion: 6021
cudnnGetCudartVersion: 8000
```