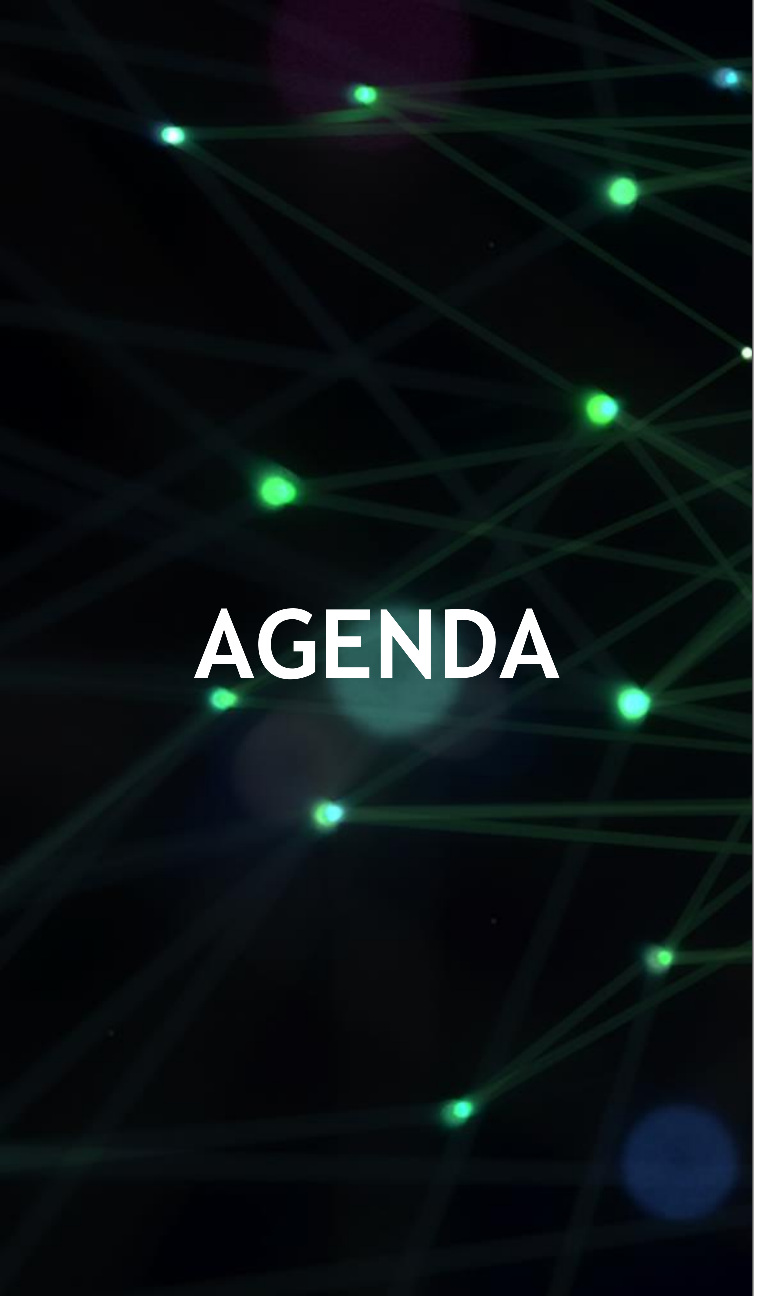
GPUDIRECT, CUDA AWARE MPI, & CUDA IPC

Steve Abbott, OLCF User Conference Call, January 2018



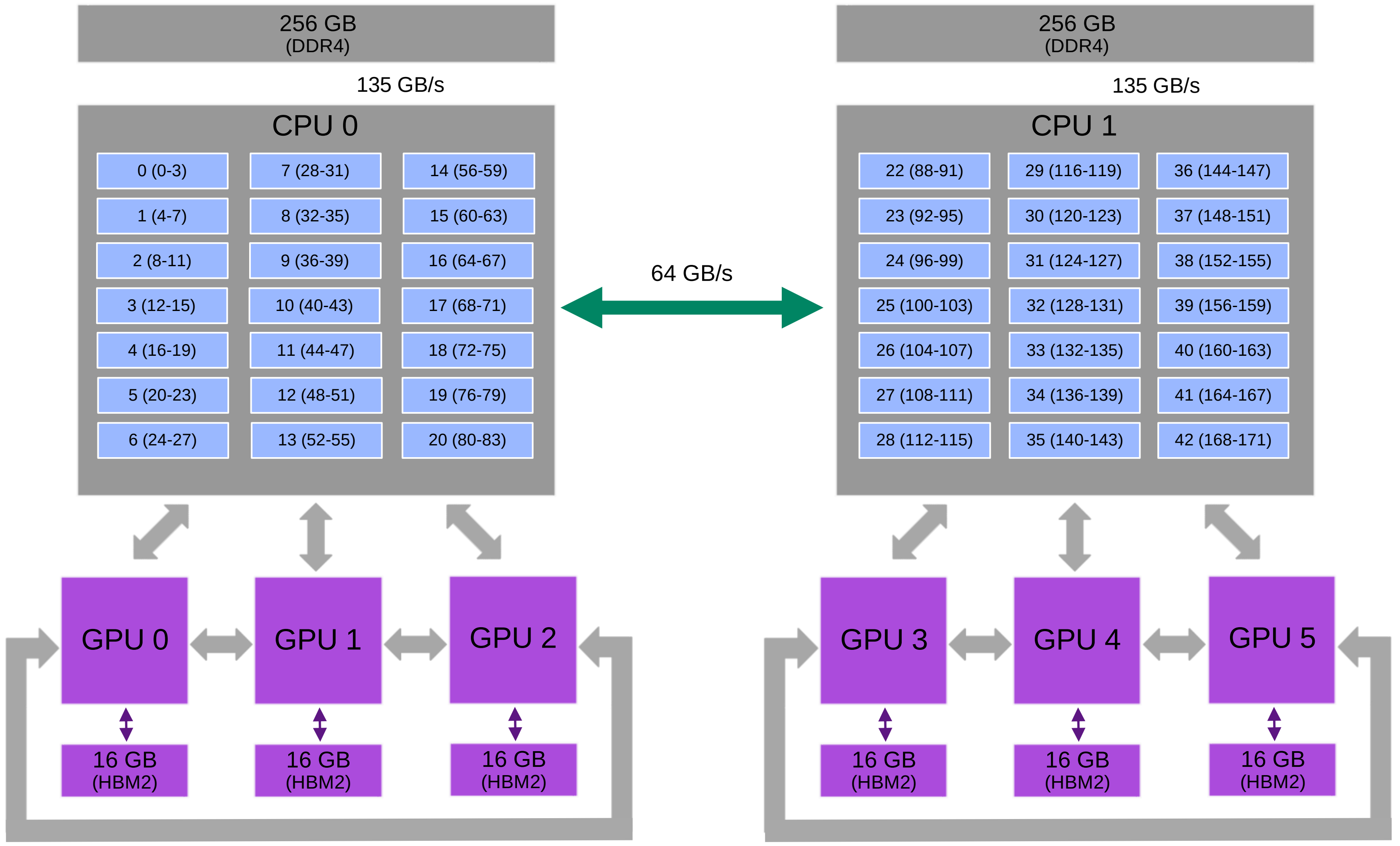
AGENDA

What is GPU Direct?
CUDA Aware MPI
~~Advanced On Node Communication~~

SUMMIT NODE OVERVIEW

SUMMIT NODE

(2) IBM POWER9 + (6) NVIDIA VOLTA V100



NVLink2 ⟷ (50 GB/s) ↕ (900 GB/s)

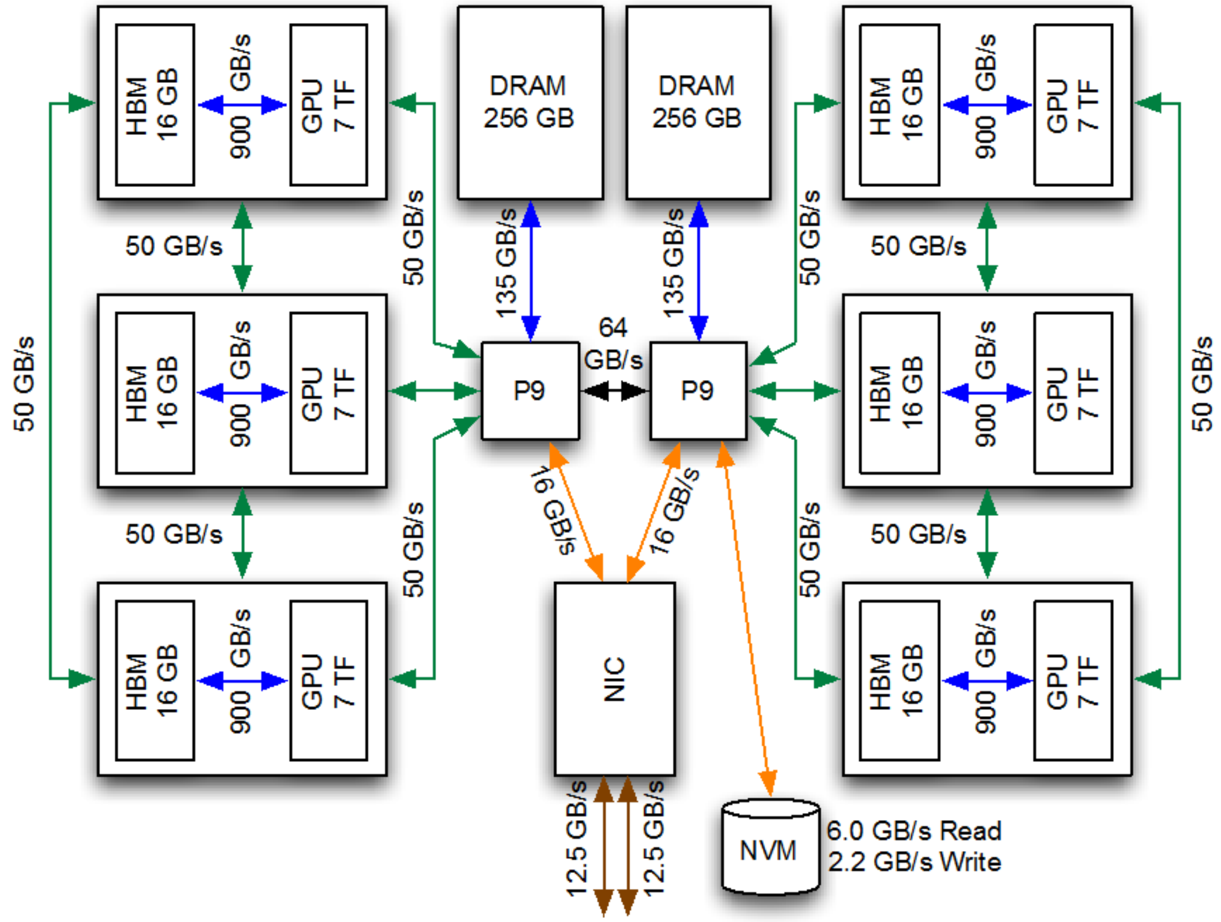
UNDER THE HOOD

Summit has fat nodes!

Many connections

Many devices

Many stacks



TF	42 TF (6x7 TF)
HBM	96 GB (6x16 GB)
DRAM	512 GB (2x16x16 GB)
NET	25 GB/s (2x12.5 GB/s)
MMsg/s	83

↔ HBM/DRAM Bus (aggregate B/W)

↔ NVLINK

↔ X-Bus (SMP)

↔ PCIe Gen4

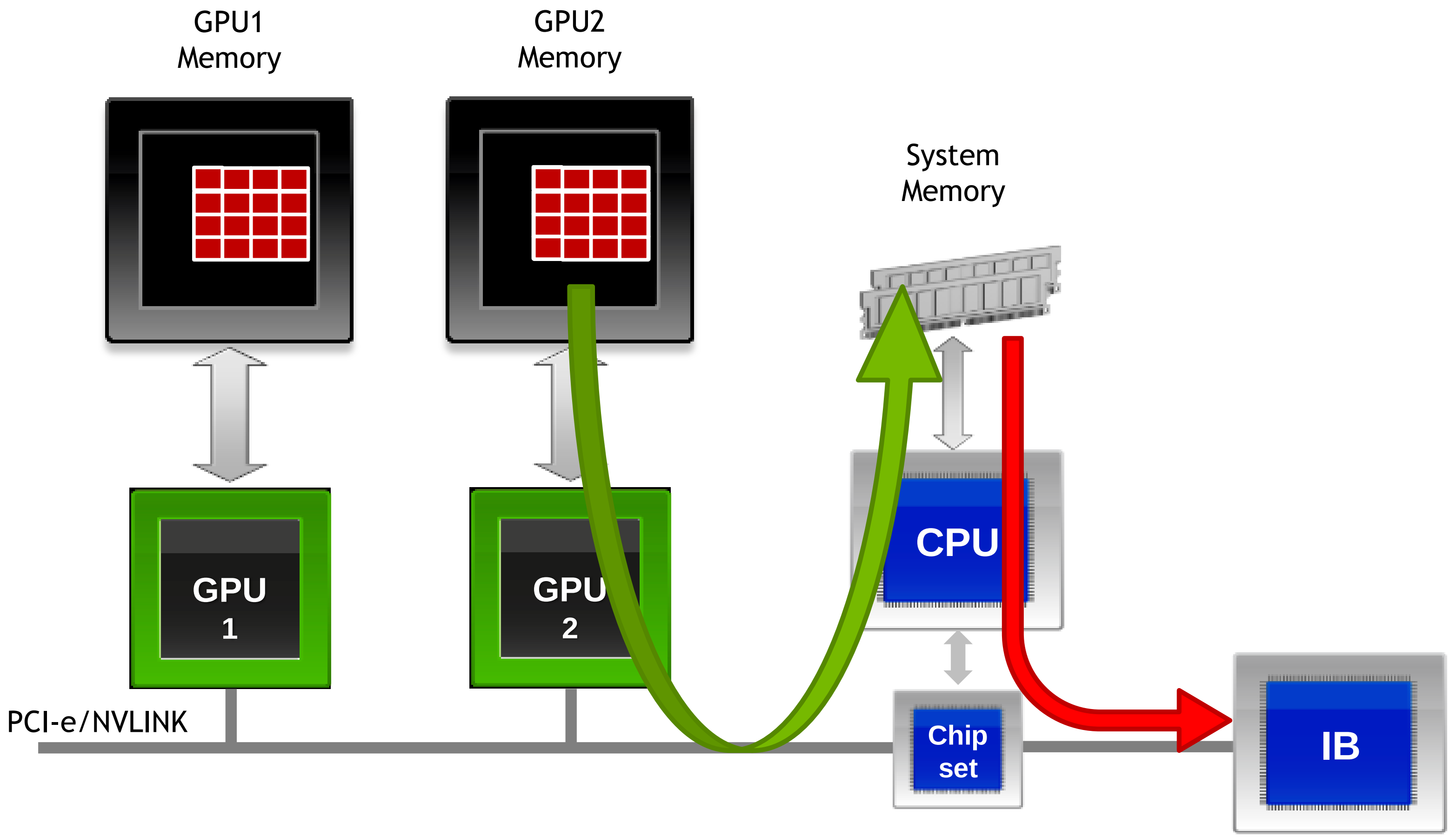
↔ EDR IB

HBM & DRAM speeds are aggregate (Read+Write).
All other speeds (X-Bus, NVLink, PCIe, IB) are bi-directional.

GPUDIRECT

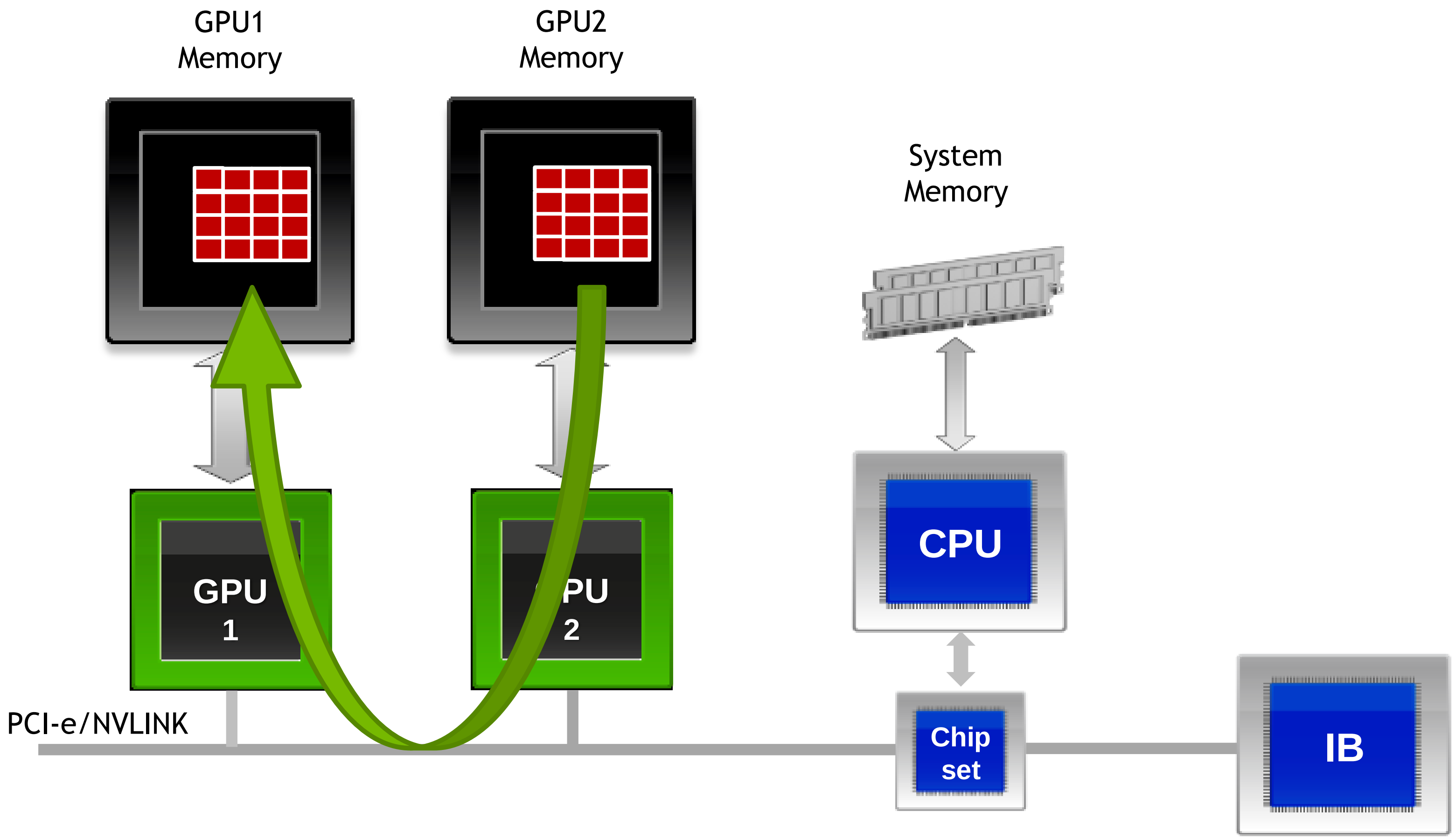
NVIDIA GPUDIRECT™

Accelerated Communication with Network & Storage Devices



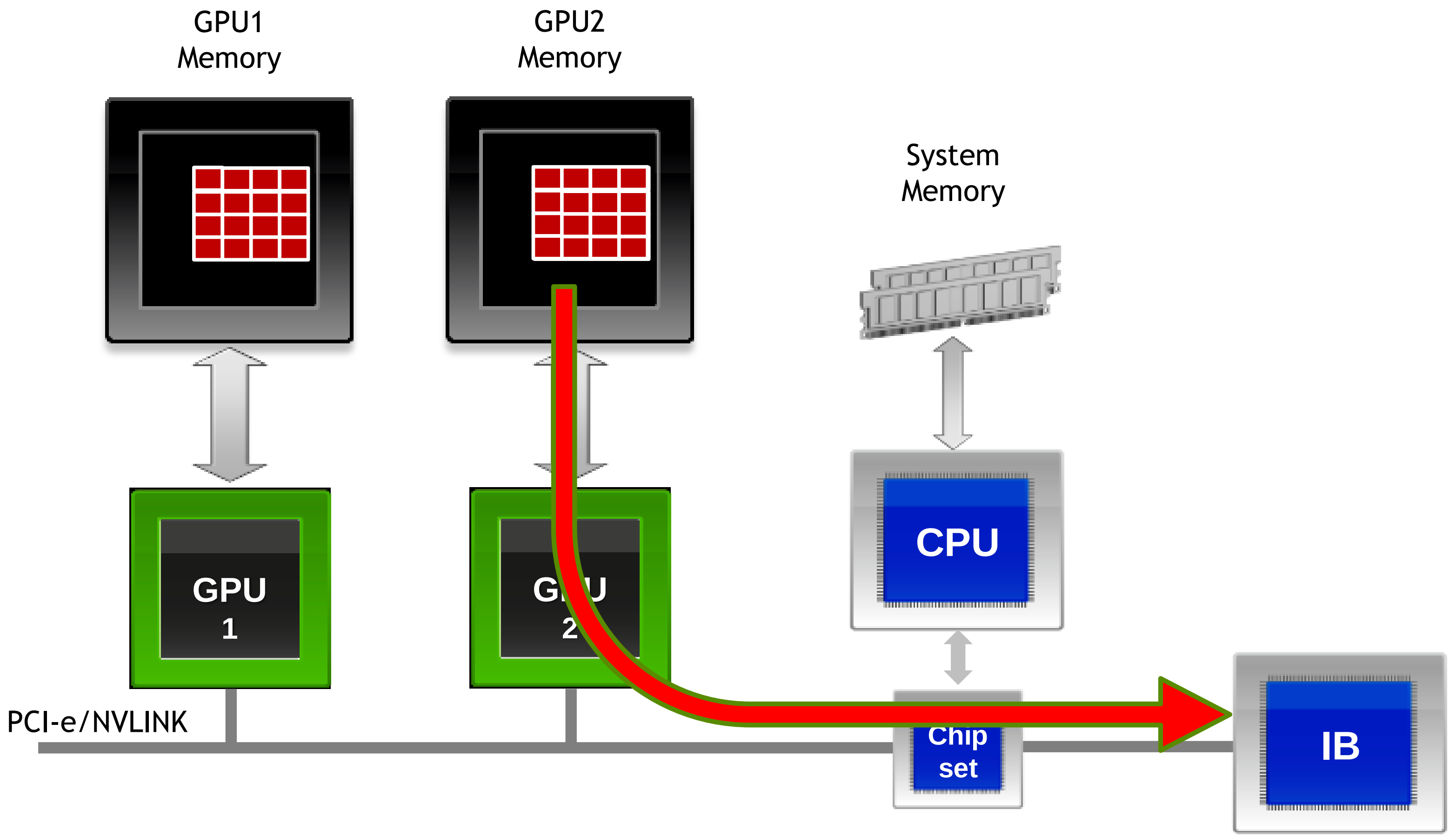
NVIDIA GPUDIRECT™

Peer to Peer Transfers



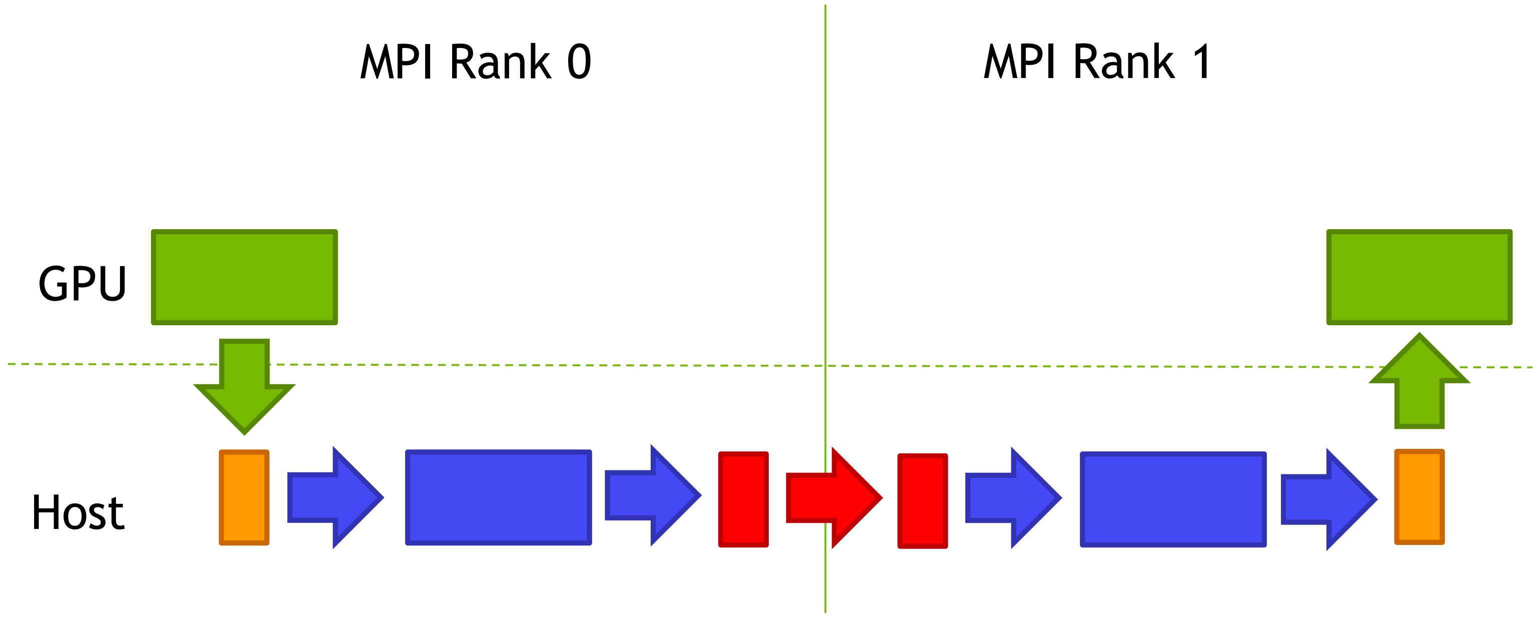
NVIDIA GPUDIRECT™

Support for RDMA



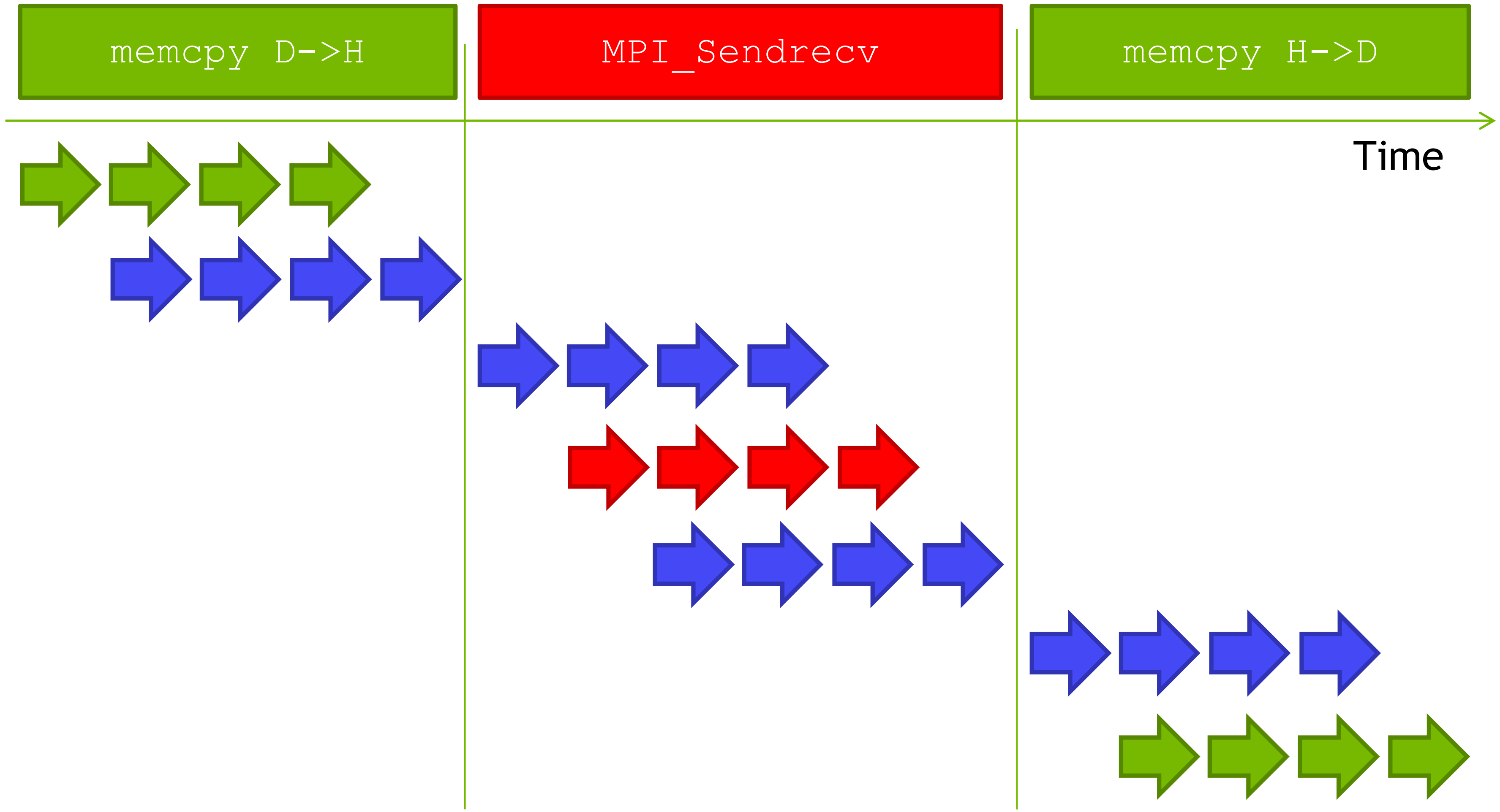
CUDA AWARE MPI FOR ON AND OFF NODE TRANSFERS

REGULAR MPI GPU TO REMOTE GPU



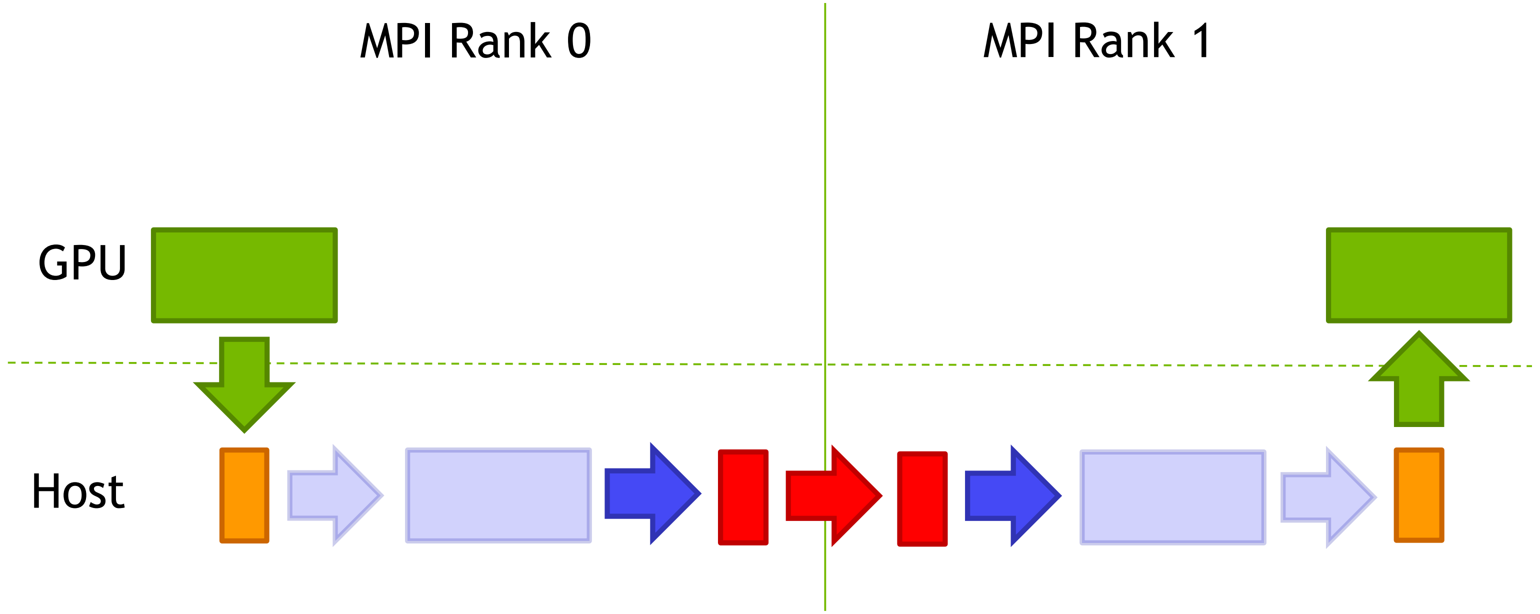
```
cudaMemcpy(s_buf_h,s_buf_d,size,cudaMemcpyDeviceToHost);
MPI_Send(s_buf_h,size,MPI_CHAR,1,tag,MPI_COMM_WORLD);

MPI_Recv(r_buf_h,size,MPI_CHAR,0,tag,MPI_COMM_WORLD,&stat);
cudaMemcpy(r_buf_d,r_buf_h,size,cudaMemcpyHostToDevice);
```

REGULAR MPI GPU TO REMOTE GPU



MPI GPU TO REMOTE GPU

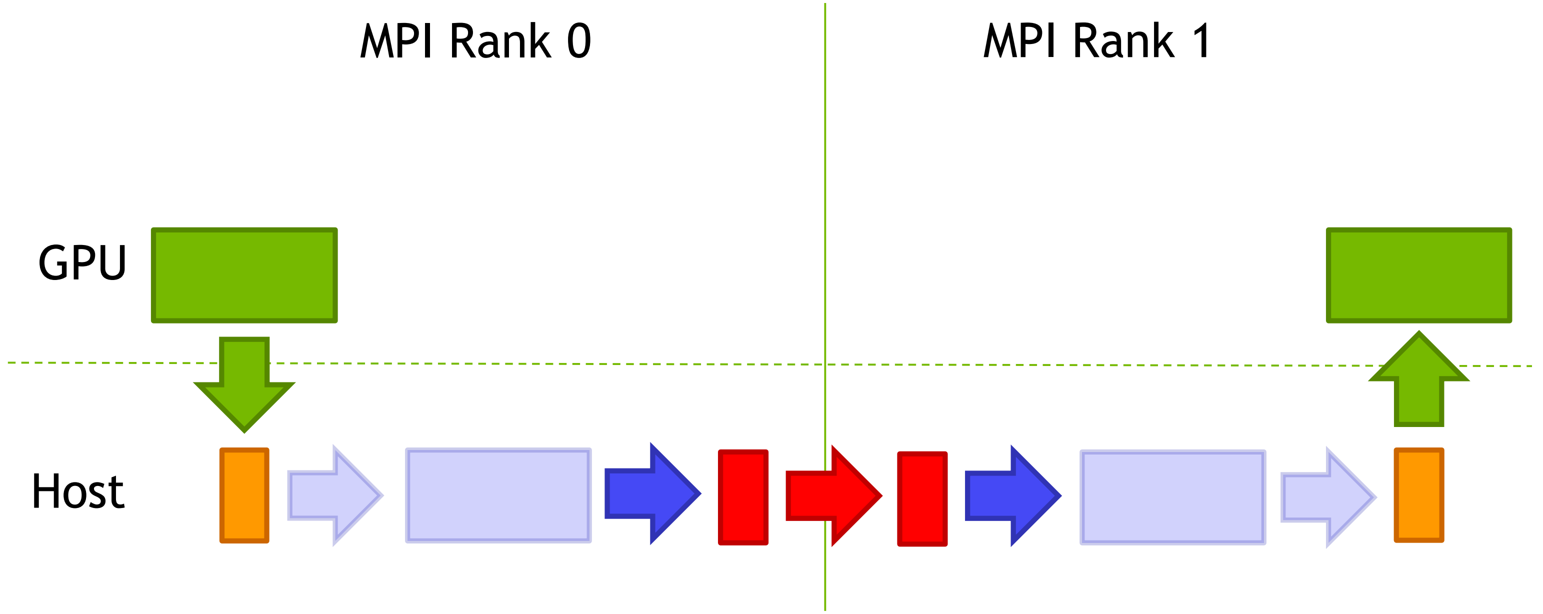
without GPUDirect



```
MPI_Send(s_buf_d,size,MPI_CHAR,1,tag,MPI_COMM_WORLD);

MPI_Recv(r_buf_d,size,MPI_CHAR,0,tag,MPI_COMM_WORLD,&stat);
```

MPI GPU TO REMOTE GPU

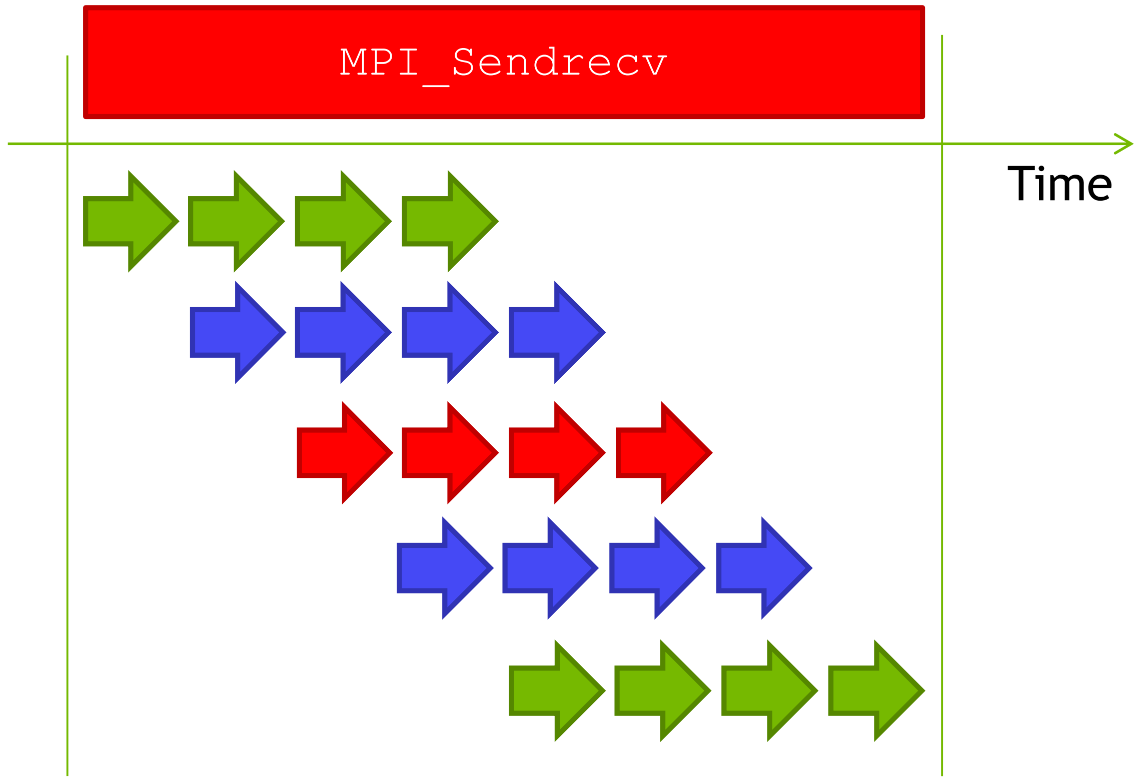
without GPUDirect



```
#pragma acc host_data use_device (s_buf, r_buf)
MPI_Send(s_buf,size,MPI_CHAR,1,tag,MPI_COMM_WORLD);

MPI_Recv(r_buf,size,MPI_CHAR,0,tag,MPI_COMM_WORLD,&stat);
```

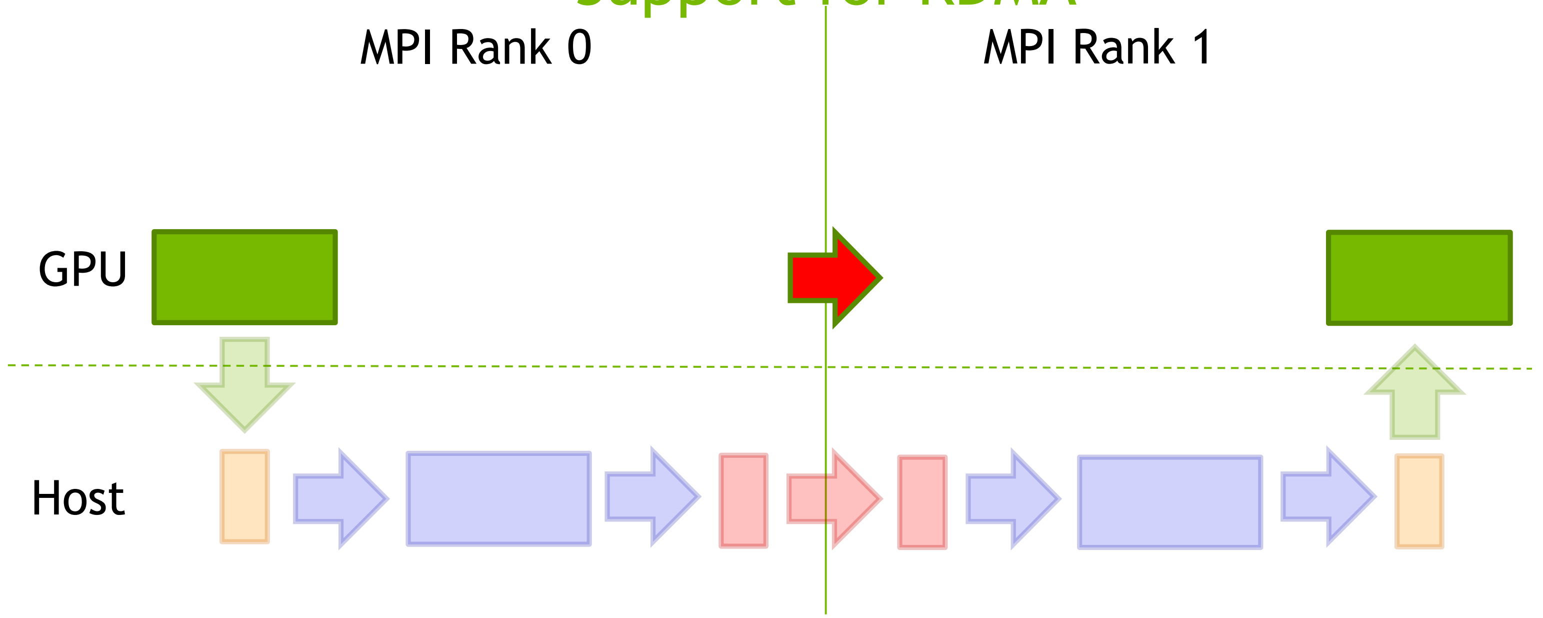
MPI GPU TO REMOTE GPU

without GPUDirect



MPI GPU TO REMOTE GPU

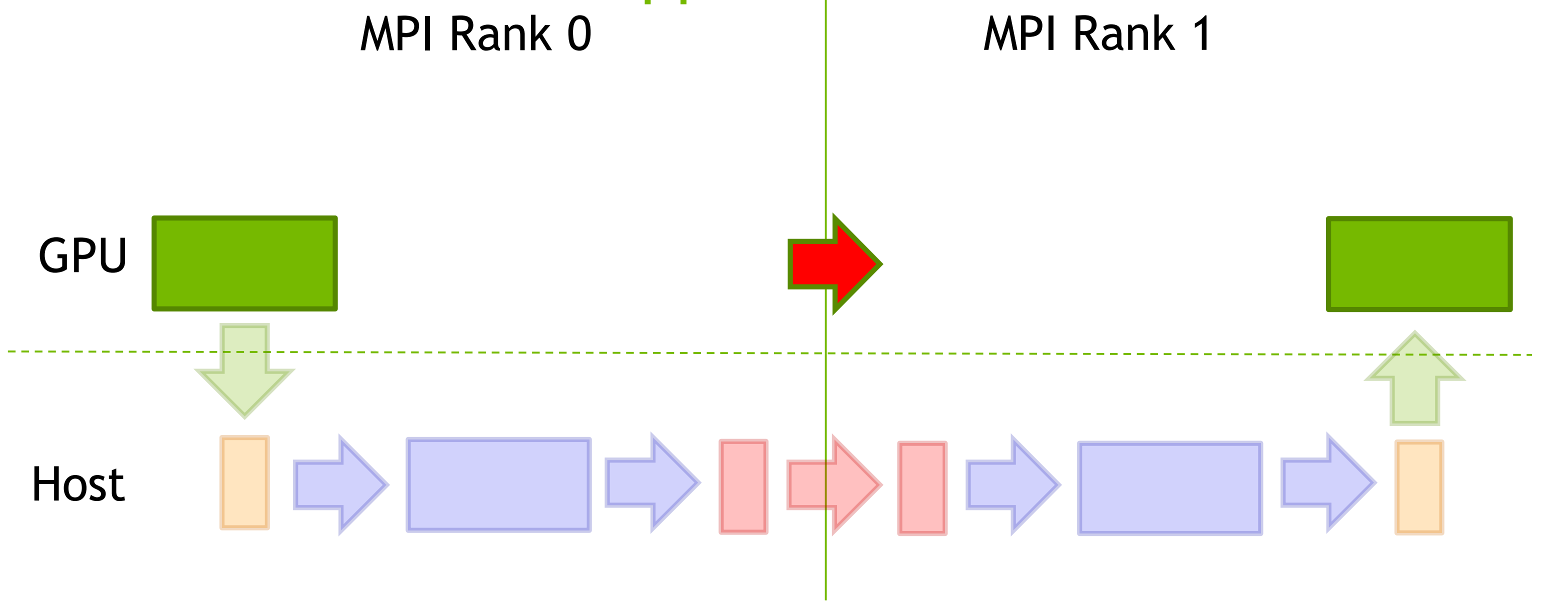
Support for RDMA



```
MPI_Send(s_buf_d,size,MPI_CHAR,1,tag,MPI_COMM_WORLD);

MPI_Recv(r_buf_d,size,MPI_CHAR,0,tag,MPI_COMM_WORLD,&stat);
```

MPI GPU TO REMOTE GPU

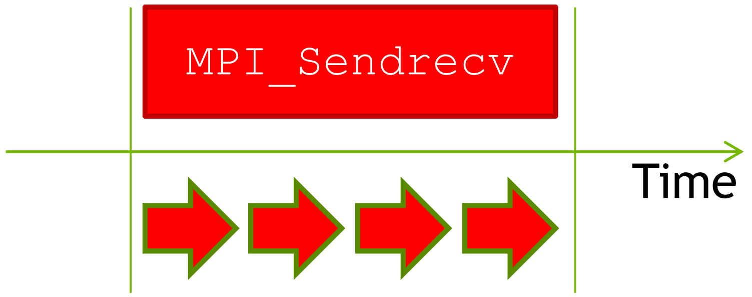
Support for RDMA



```
#pragma acc host_data use_device (s_buf, r_buf)
MPI_Send(s_buf,size,MPI_CHAR,1,tag,MPI_COMM_WORLD);

MPI_Recv(r_buf,size,MPI_CHAR,0,tag,MPI_COMM_WORLD,&stat);
```

MPI GPU TO REMOTE GPU

Support for RDMA



JSRUN/SMPI GPU OPTIONS

Running On Summit

To enable CUDA aware MPI, use **jsrun --smpiargs="-gpu"**

KNOWN ISSUES ON SUMMIT

Things to watch out for (as of January)

Problems with Multiple resource sets per node:

```
$> jsrun –g 1 –a 1 --smpiargs=“-gpu” ….

[1]Error opening IPC Memhandle from peer:0, invalid argument
```

- One workaround: set PAMI_DISABLE_IPC=1
  - Expect poor performance, but a good functionality check

Will be resolved by software updates later this year

PERFORMANT WORKAROUNDS

Running On Summit

Option 1: Run in one resource set and set GPU affinity in your code

(do NOT restrict CUDA_VISIBLE_DEVICES, but you can permute it)

Option 2: Use a wrapper script

- Add “#BSUB -step_cgroup n” to your LSF options
- Run with `jsrun <your-jsrun-options> --smpiargs=”-gpu” ./gpu_setter.sh <your app>`
  - (script on next slide)
- Will need to be careful about your CPU bindings!

```
#! /bin/bash
# gpu_setter.sh
# Rudimentary GPU affinity setter for Summit
# >$ jsrun -rs_per_host 1 -gpu_per_rs 6 <task/cpu option> ./gpu_setter.sh <your app>

# This script assumes your code does not attempt to set its own
# GPU affinity (e.g. with cudaSetDevice). Using this affinity script
# with a code that does its own internal GPU selection probably won't work!

# Compute device number from OpenMPI local rank environment variable
# Keeping in mind Summit has 6 GPUs per node

mydevice=$((${OMPI_COMM_WORLD_LOCAL_RANK} % 6))

# CUDA_VISIBLE_DEVICES controls both what GPUs are visible to your process
# and the order they appear in. By putting "mydevice" first the in list, we
# make sure it shows up as device "0" to the process so it's automatically selected.
# The order of the other devices doesn't matter, only that all devices (0-5) are
present.

CUDA_VISIBLE_DEVICES="${mydevice},0,1,2,3,4,5"

# Process with sed to remove the duplicate and reform the list, keeping the order we
set

CUDA_VISIBLE_DEVICES=$(sed -r ':a; s/\b([[:alnum:]]+)\b(.*)\b\1\b/\1\2/g; ta;
s/(,,)+/,/g; s/, *$//' <<< $CUDA_VISIBLE_DEVICES)

export CUDA_VISIBLE_DEVICES

# Launch the application we were given
exec "$@"
```

GPU TO GPU COMMUNICATION

- CUDA aware MPI functionally portable
    - OpenACC/MP interoperable
    - Performance may vary between on/off node, socket, HW support for GPU Direct
    - Unified memory support varies between implementations, but it becoming common
- For more information on the following advanced on-node communication models, see the OLCF Training Archive or Summit Training Workshops
    - Single-process, multi-GPU
    - Multi-process, single-gpu

NVIDIA®