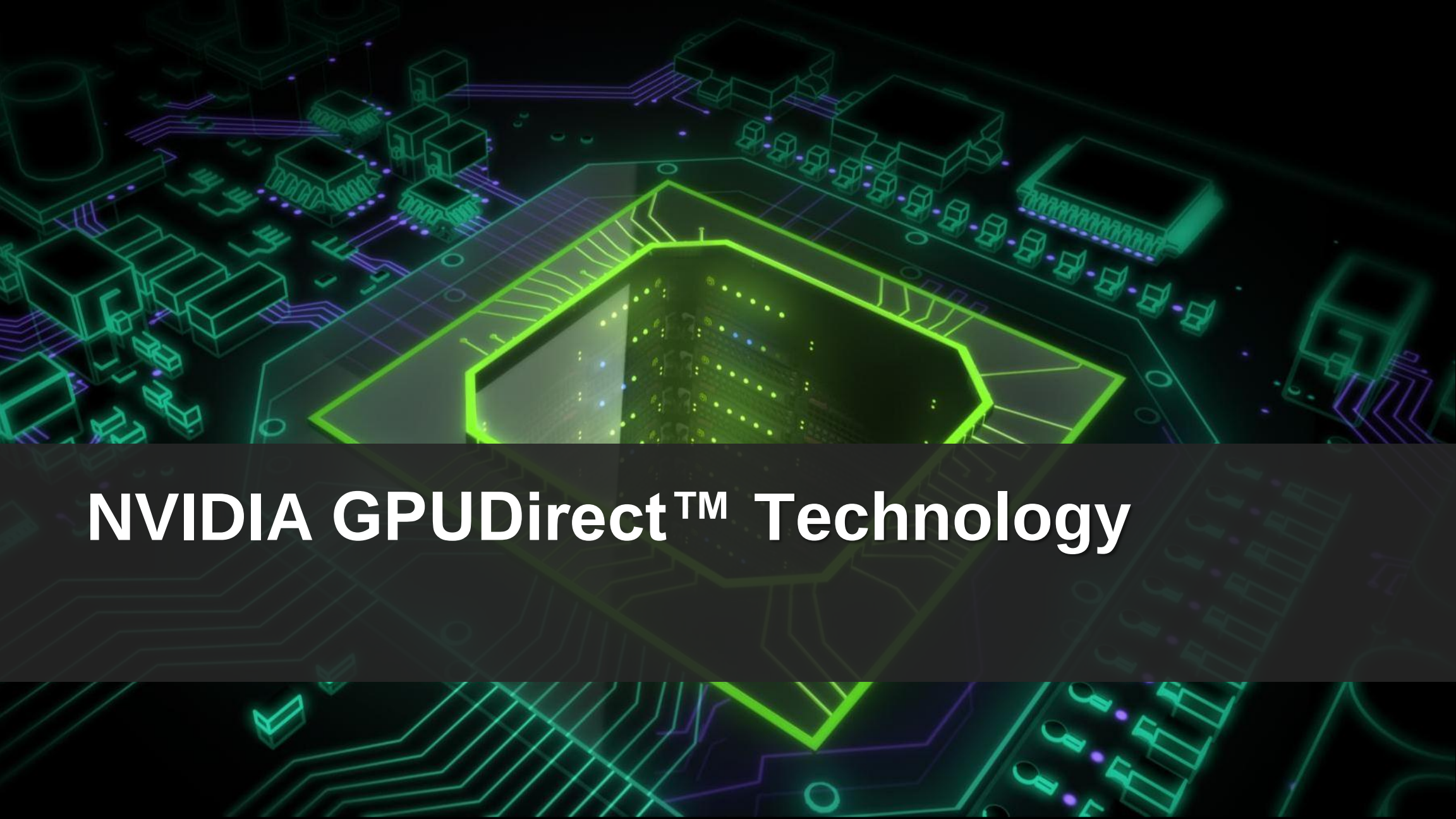


NVIDIA GPUDirect™ Technology

NVIDIA GPUDirect™: *Eliminating CPU Overhead*

High Bandwidth, Low Latency Communication for GPU Accelerated Applications

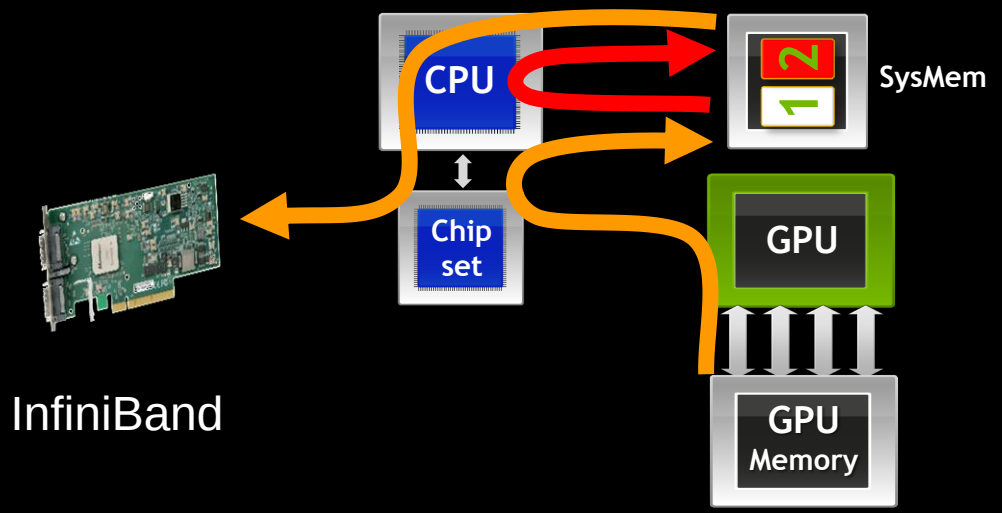
Accelerated Communication with Network & Storage Devices	• Direct access to CUDA memory for 3rd party devices • Eliminates unnecessary memory copies & CPU overhead • CUDA 3.1 and later
Peer-to-Peer Communication between GPUs	• Peer-to-Peer memory access, transfers & synchronization • Less code, higher programmer productivity • CUDA 4.0 and later
GPUDirect for Video	• Optimized pipeline for frame-based video devices • Low-latency communication with OpenGL, DirectX, or CUDA • CUDA 4.2 and later
RDMA	• Direct communication between GPUs across a cluster • Significantly increased MPISendRecv efficiency • CUDA 5.0 and later

NVIDIA GPUDirect™
Accelerated Communication with Network and Storage Devices

Without GPUDirect

Same data copied three times:

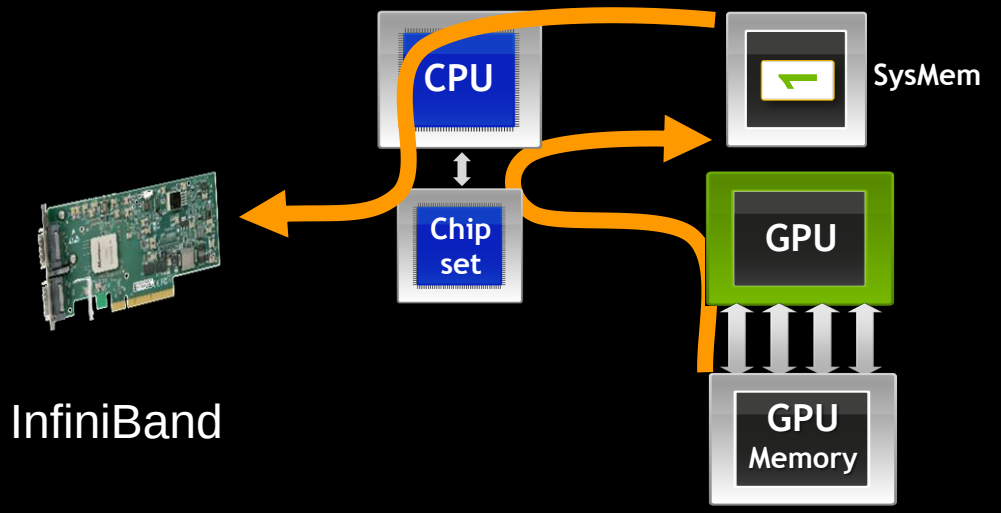
1. GPU writes to pinned sysmem1
2. CPU copies from sysmem1 to sysmem2
3. InfiniBand driver copies from sysmem2



With GPUDirect

Data only copied twice

Sharing pinned system memory makes sysmem-to-sysmem copy unnecessary

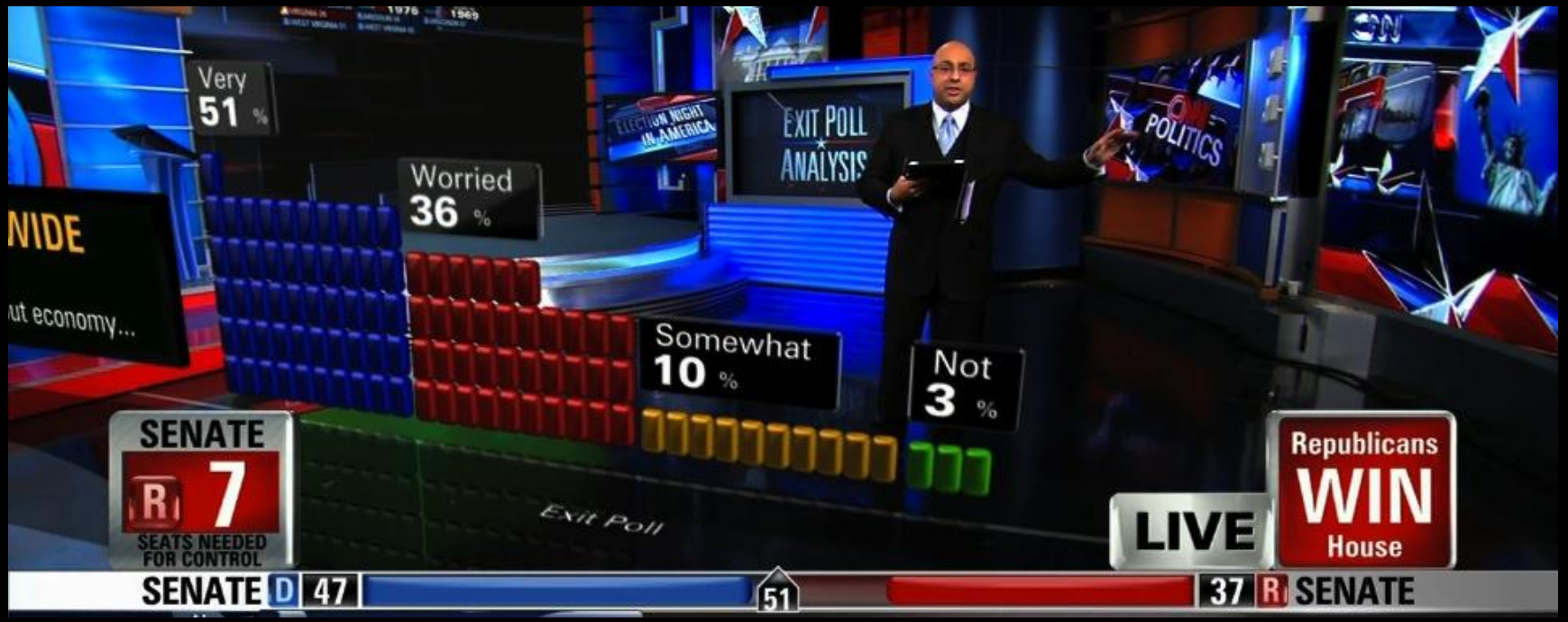


Using GPUDirect
Accelerated Communication with Network and Storage Devices

- CUDA 4.0 and later:
  - Set the environment variable CUDA_NIC_INTEROP=1
    - Ensures access to CUDA pinned memory by third party drivers
    - All CUDA pinned memory will be allocated first in user-mode as pageable memory
    - CUDA and third party driver pin and share the pages via get_user_pages()
  - Requires NVIDIA Drivers v270.41.19 or later
  - Requires Linux kernel 2.6.15 or later (no Linux kernel patch required)

- Earlier releases:
  - Only necessary when using NVIDIA Drivers older than v270.41.19
  - Developed jointly by NVIDIA and Mellanox
    - New interfaces in the CUDA and Mellanox drivers + Linux kernel patch
  - Installation instructions at http://developer.nvidia.com/gpudirect
    - Supported for Tesla M and S datacenter products on RHEL only

NVIDIA GPUDirect™ For Video
Accelerating Communication with Video I/O Devices



AJA VIDEO SYSTEMS · Blackmagicdesign · bluefish444 · DVS A Rohde & Schwarz Company · DELTACAST · matrox Digital Video Solutions

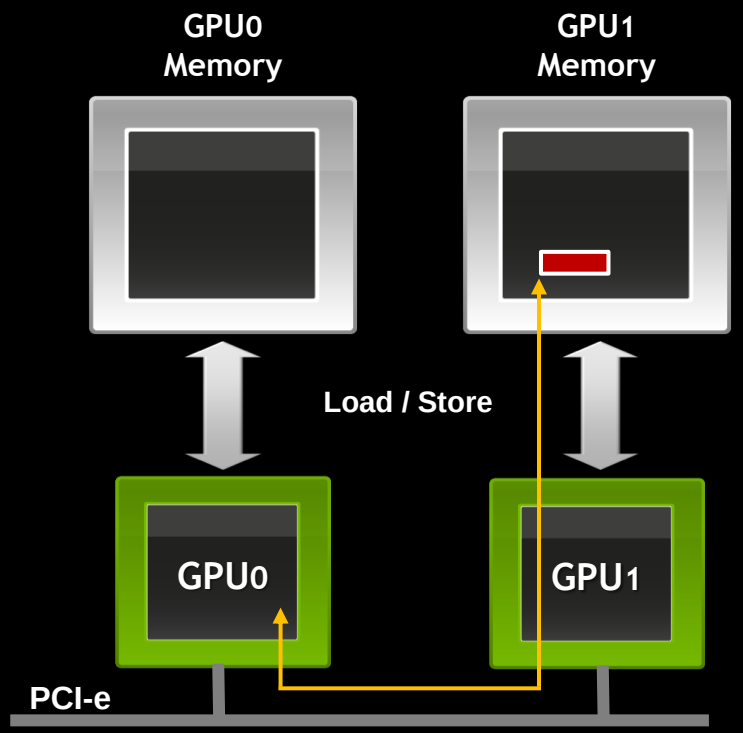
NVIDIA GPUDirect™ For Video
Accelerating Communication with Video I/O Devices

- Low Latency I/O with OpenGL, DirectX or CUDA
- Shared system memory model with synchronization for data streaming
- Support for asynchronous data transfers to maximize GPU processing time
- Minimized CPU overhead

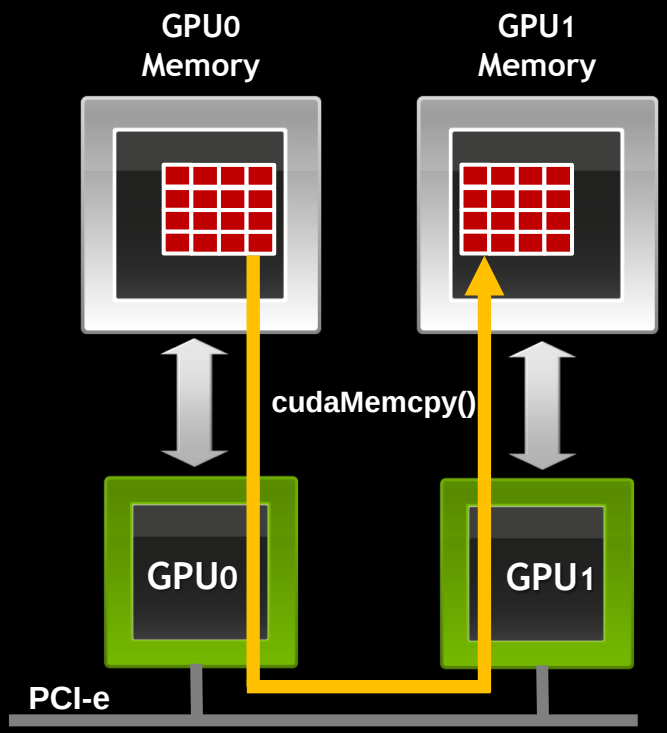
- Windows 7, Linux
- OpenGL, DirectX or CUDA
- Quadro 4000, 5000, 6000 Tesla C2075, M-Series
- Industry Leading I/O Boards

NVIDIA GPUDirect™
Peer-to-Peer Communication

Direct Access



Direct Transfers



Eliminates system memory allocation & copy overhead

More convenient multi-GPU programming

Using GPUDirect
Peer-to-Peer Communication Between GPUs

- Direct Access
  - GPU_0 reads or writes GPU_1 memory (load/store)
  - Data cached in L2 of the target GPU
- Direct Transfers
  - cudaMemcpy() initiates DMA copy from GPU_0 memory to GPU_1 memory
  - Works transparently with CUDA Unified Virtual Addressing (UVA)
- Examples in the CUDA C Programming Guide and simpleP2P code sample in the GPU Computing SDK
  - Requires CUDA 4.0 and NVIDIA Drivers v270.41.19 or later
  - Supported on Tesla 20-series and other Fermi GPUs
    - 64-bit applications on Linux and Windows TCC

GPUDirect P2P Communication on Dual-IOH Systems (1/2)

PCI-e P2P Communication Not Supported Between Intel IOH Chipsets

- NVIDIA GPUs are designed to take full advantage of the PCI-e Gen2 standard, including the Peer-to-Peer communication
- The IOH chipset does not support the full PCI-e Gen2 specification for P2P communication with other IOH chipsets
  - *"The IOH does not support non-contiguous byte enables from PCI Express for remote peer-to-peer MMIO transactions. This is an additional restriction over the PCI Express standard requirements to prevent incompatibility with Intel QuickPath Interconnect."**
- See slides #14-18 for diagrams explaining supported system configurations

This limitation has minimal impact on developers (see next slide…)

* http://www.intel.com/Assets/PDF/datasheet/321328.pdf

GPUDirect P2P Communication on Dual-IOH Systems (2/2)

(…continued from previous slide)

Dual-IOH Limitation has Minimal Impact on Developers

- GPUDirect P2P Transfers code path is always supported
  - cudaMemcopy() automatically falls back to Device-to-Host-to-Device when P2P is unavailable
- GPUDirect P2P Access is a single-node optimization technique
  - load/store in device code is an optimization when the 2 GPUs that need to communicate are in the same node, but many applications also need a non-P2P code path to support communication between GPUs in different nodes which can be used when communicating with GPUs separated by a QPI bus as well

NVIDIA is investigating whether GPU P2P across QPI* can be supported by adding functionality to future GPU architectures

* http://www.intel.com/Assets/PDF/datasheet/321328.pdf

Unified Virtual Addressing

- New in CUDA 4.0
- One address space for all CPU and GPU memory
  - Determine physical memory location from pointer value
  - Enables libraries to simplify their interfaces (e.g. cudaMemcpy)

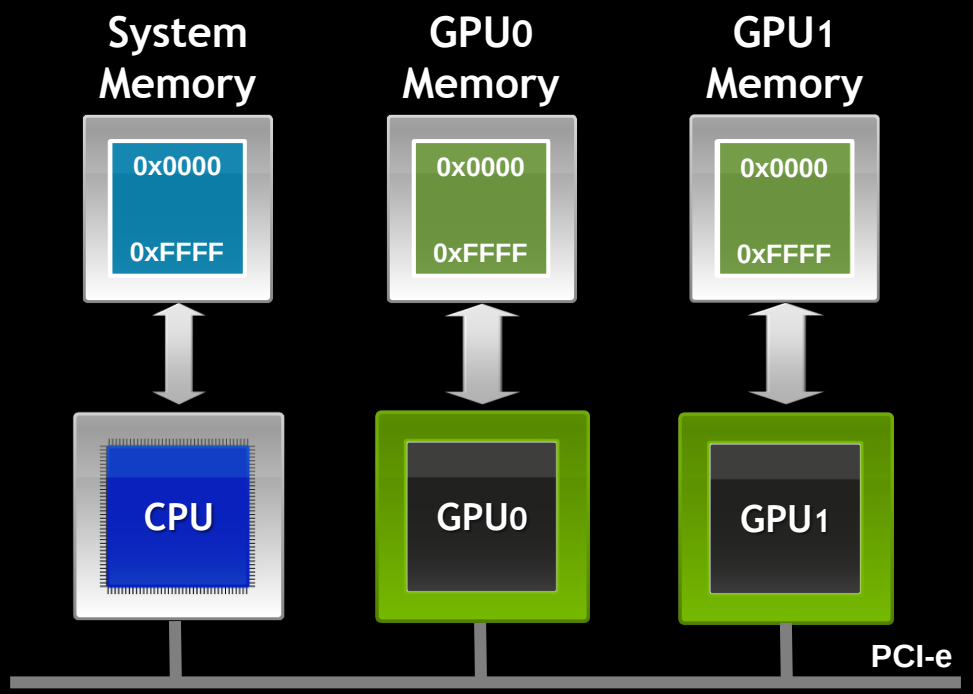
Before UVA	With UVA
Separate options for each permutation	One function handles all cases
cudaMemcpyHostToHost cudaMemcpyHostToDevice cudaMemcpyDeviceToHost cudaMemcpyDeviceToDevice	cudaMemcpyDefault (data location becomes an implementation detail)

- Supported on Tesla 20-series and other Fermi GPUs
  - 64-bit applications on Linux and Windows TCC

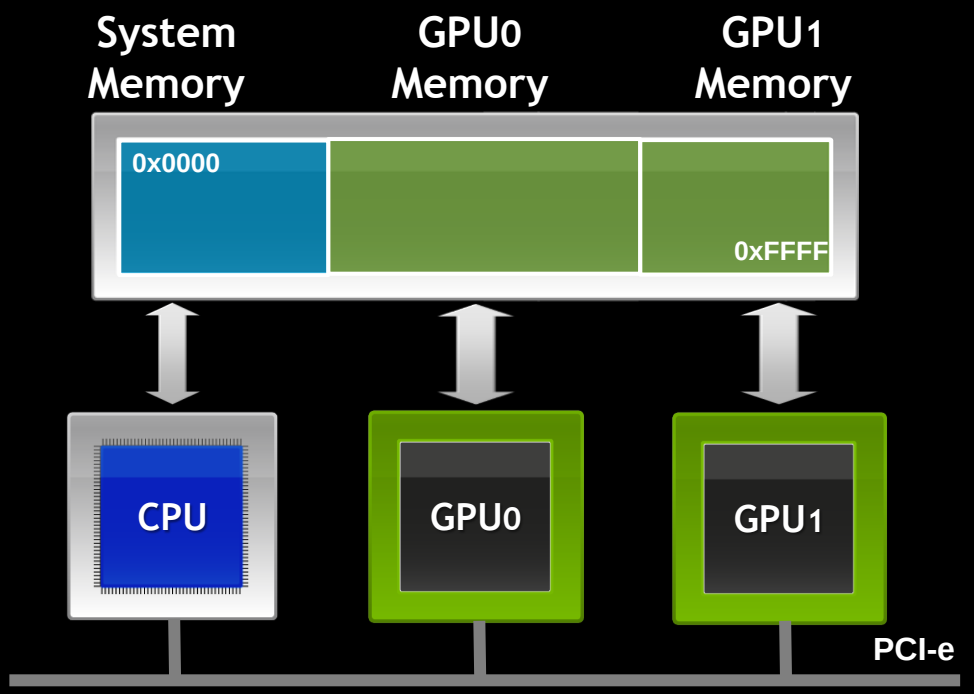
Unified Virtual Addressing

Easier to Program with Single Address Space

No UVA: Multiple Memory Spaces



UVA: Single Address Space



MPI Integration of NVIDIA GPUDirect™

- **MPI libraries with support for NVIDIA GPUDirect and Unified Virtual Addressing (UVA) enables:**
  - **MPI transfer primitives copy data directly to/from GPU memory**
  - **MPI library can differentiate between device memory and host memory without any hints from the user**
  - **Programmer productivity: less application code for data transfers**

Code without MPI integration

At Sender:

```
cudaMemcpy(s_buf, s_device, size, cudaMemcpyDeviceToHost);
MPI_Send(s_buf, size, MPI_CHAR, 1, 1, MPI_COMM_WORLD);
```

At Receiver:

```
MPI_Recv(r_buf, size, MPI_CHAR, 0, 1, MPI_COMM_WORLD, &req);
cudaMemcpy(r_device, r_buf, size, cudaMemcpyHostToDevice);
```

Code with MPI integration

At Sender:

```
MPI_Send(s_device, size, …);
```

At Receiver:

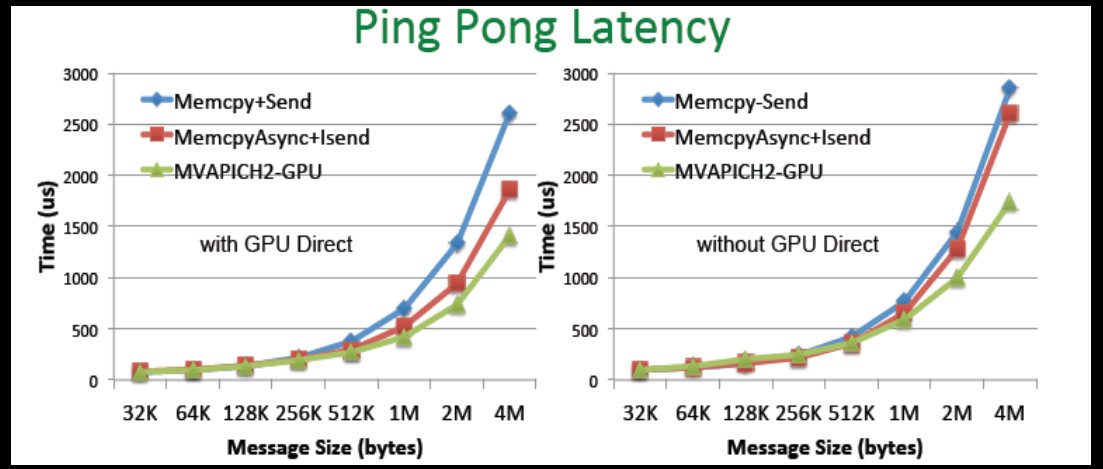
```
MPI_Recv(r_device, size, …);
```

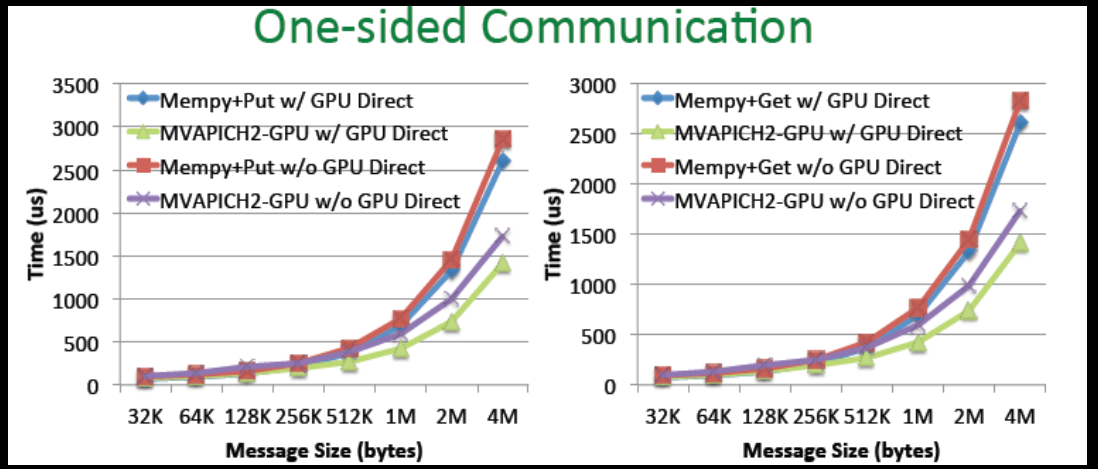
Open MPI

- Transfer data directly to/from CUDA device memory via MPI calls
- Code is currently available in the Open MPI trunk, available at:
  http://www.open-mpi.org/nightly/trunk (contributed by NVIDIA)
- More details in the Open MPI FAQ
  - Features: http://www.open-mpi.org/faq/?category=running#mpi-cuda-support
  - Build Instructions: http://www.open-mpi.org/faq/?category=building#build-cuda

MVAPICH2-GPU

- **Upcoming MVAPICH2 support for GPU-GPU communication with**
  - Memory detection and overlap CUDA copy and RDMA transfer





With GPUDirect
- 45% improvement compared to Memcpy+Send (4MB)
- 24% improvement compared to MemcpyAsync+Isend (4MB)

Without GPUDirect
- 38% improvement compared to Memcpy+send (4MB)
- 33% improvement compared to MemcpyAsync+Isend (4MB)

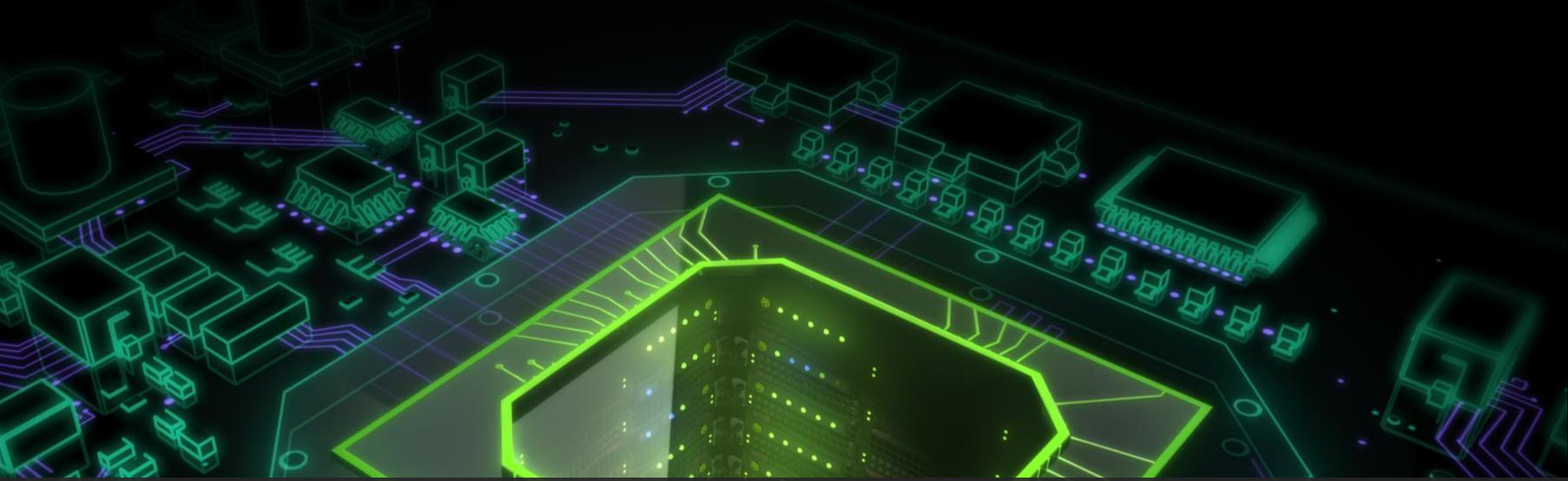
With GPUDirect
- 45% improvement compared to Memcpy+Put

Without GPUDirect
- 39% improvement compared with Memcpy+Put

Similar improvement for Get operation
Major improvement in programming

Measurements from:
H. Wang, S. Potluri, M. Luo, A. Singh, S. Sur and D. K. Panda, *"MVAPICH2-GPU: Optimized GPU to GPU Communication for InfiniBand Clusters"*, Int'l Supercomputing Conference 2011 (ISC), Hamburg

http://mvapich.cse.ohio-state.edu/



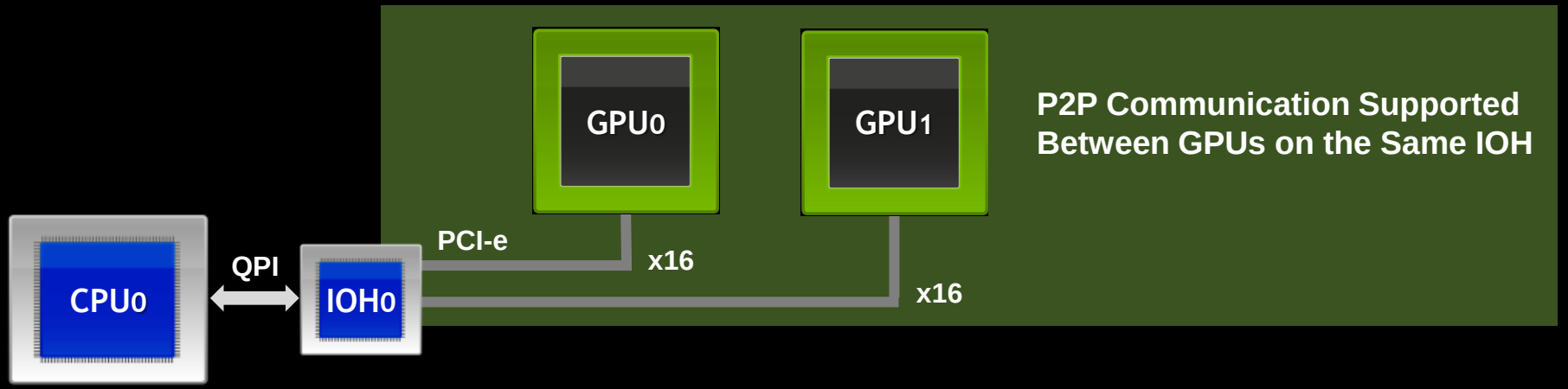
NVIDIA GPUDirect™ Technology

System Design Guidance

http://developer.nvidia.com/gpudirect

Designing Systems for P2P Communication

Single IOH Systems: Fully Supported

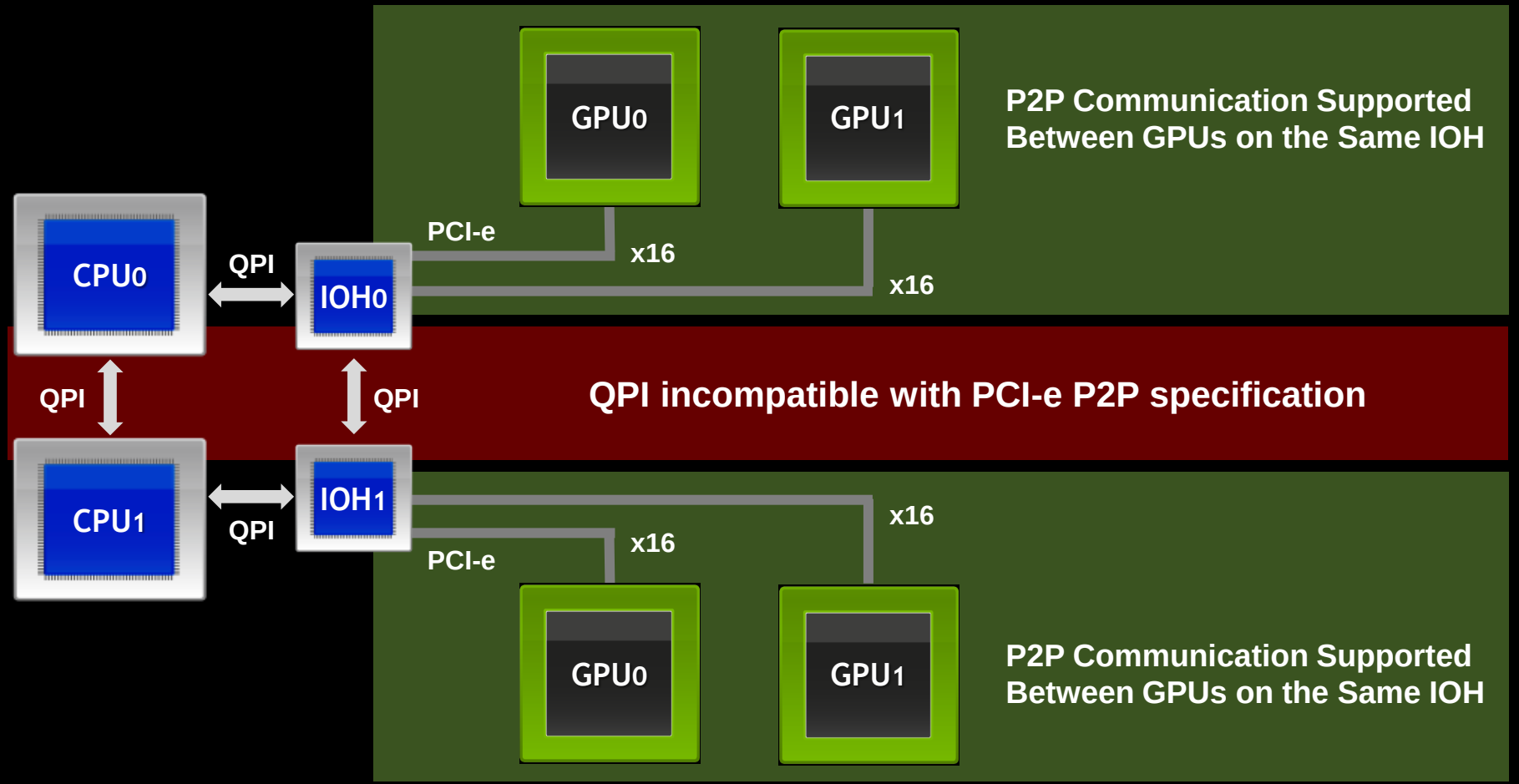


- A single IO Hub (IOH) can support up to 36 PCI-e lanes
- When connecting GPUs directly to an IOH, NVIDIA recommends using 16 lanes for each GPU

http://developer.nvidia.com/gpudirect

Designing Systems for P2P Communication

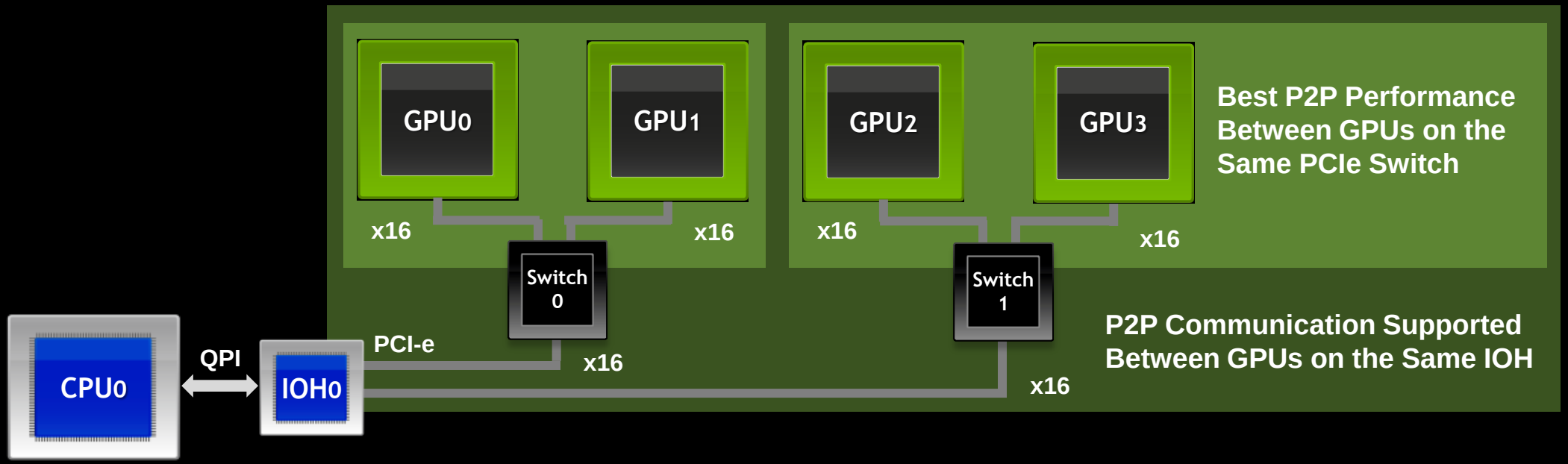
P2P Communication Across QPI: Not Supported



http://developer.nvidia.com/gpudirect

Designing Systems for P2P Communication

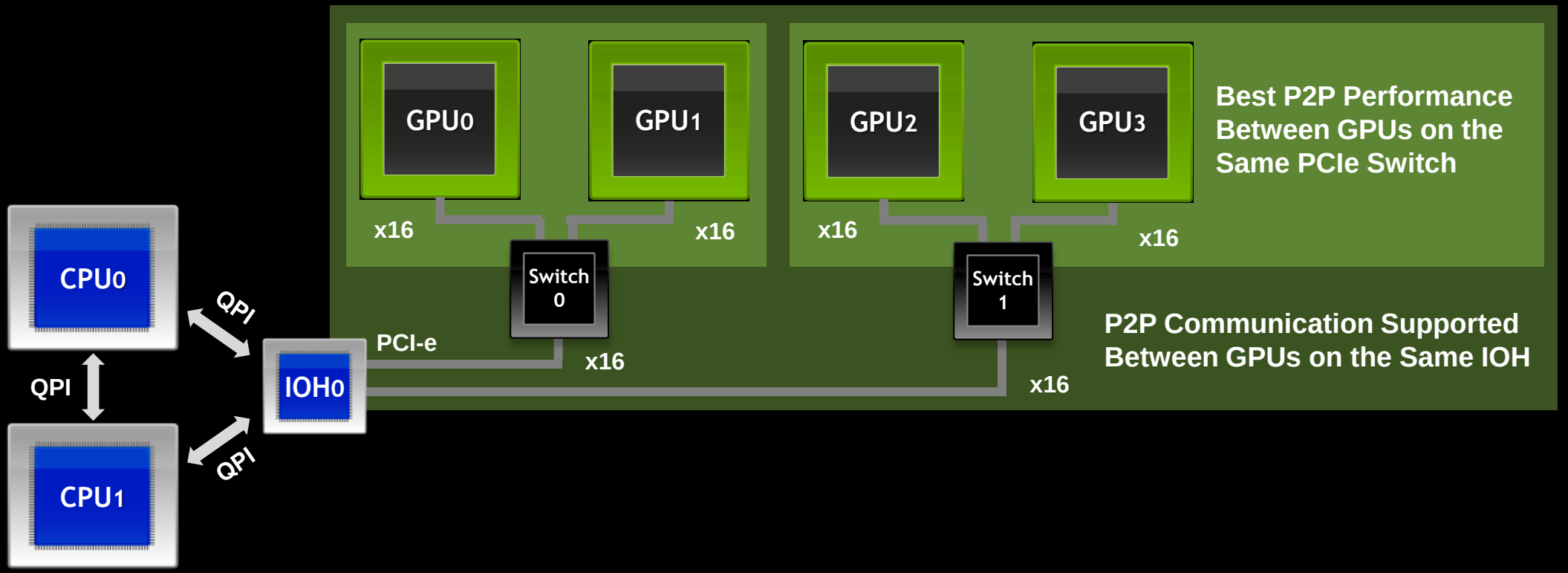
Single IOH Systems with PCI-e Switches: Fully Supported



http://developer.nvidia.com/gpudirect

Designing Systems for P2P Communication

Dual-CPU Systems with Single IOH: Fully Supported



http://developer.nvidia.com/gpudirect