GPUs as better MPI Citizens

Author: Dale Southard, NVIDIA
Date: 4/6/2011

GPU Technology Conference 2011
October 11-14 | San Jose, CA

The one event you can't afford to miss

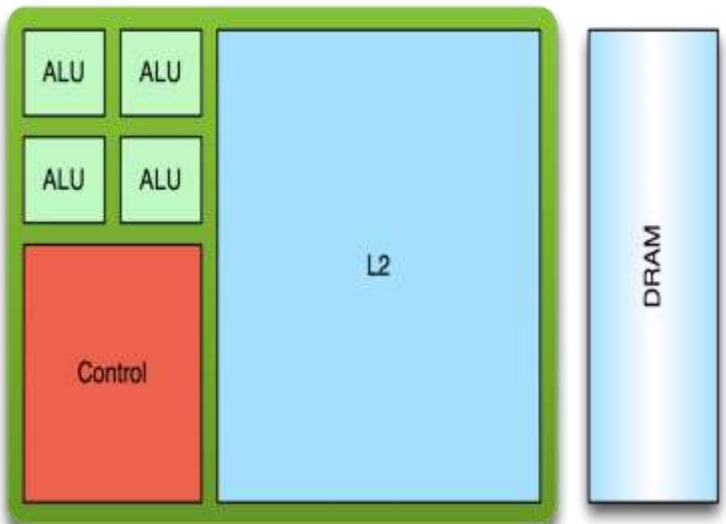
- Learn about leading-edge advances in GPU computing
- Explore the research as well as the commercial applications
- Discover advances in computational visualization
- Take a deep dive into parallel programming

Ways to participate

- Speak – share your work and gain exposure as a thought leader
- Register – learn from the experts and network with your peers
- Exhibit/Sponsor – promote your company as a key player in the GPU ecosystem

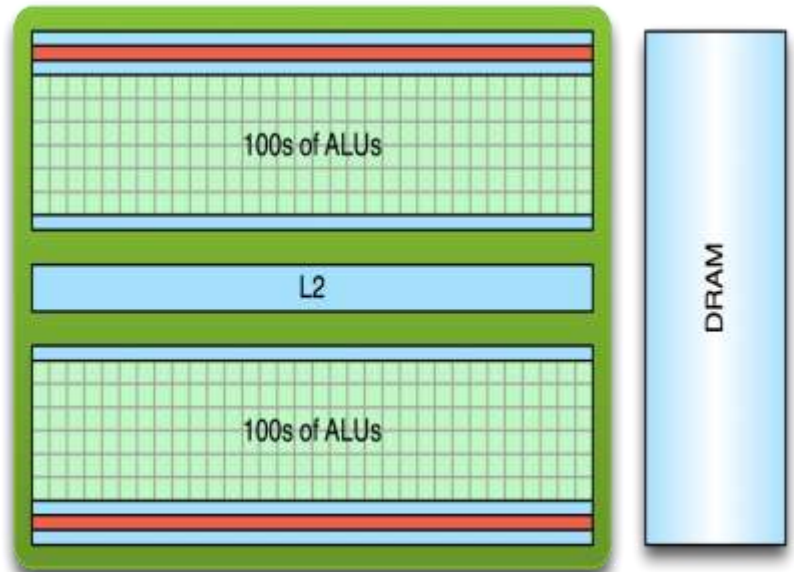
www.gputechconf.com

The Programming Model



CPU

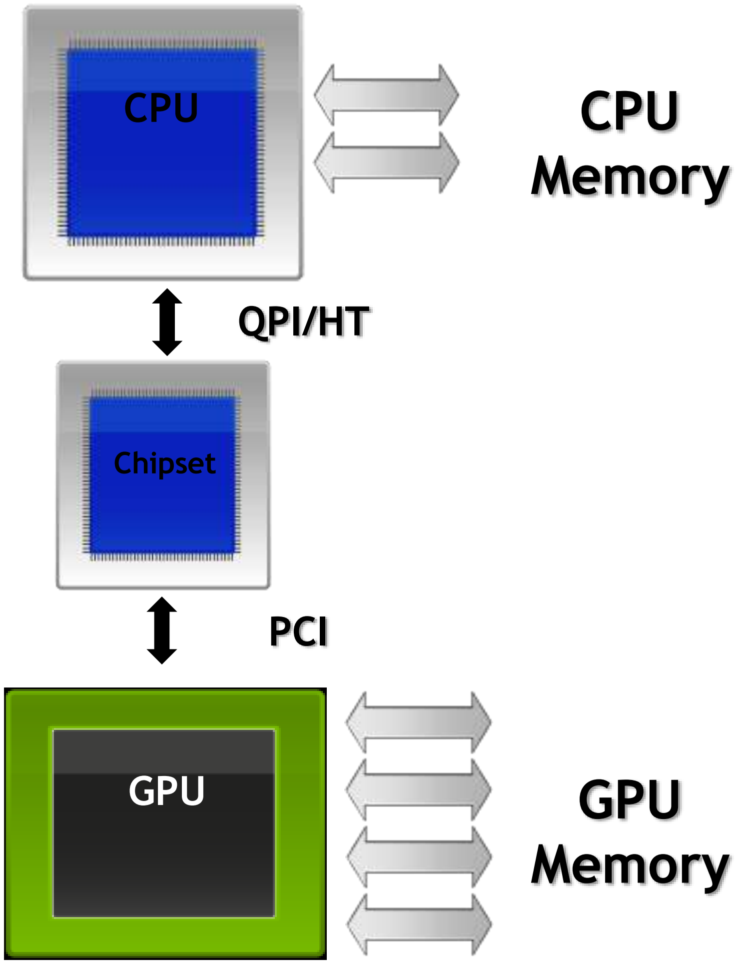
- Optimized for low-latency access (caches)
- Control logic for out-of-order and speculative execution



GPU

- Optimized for data-parallel, high throughput
- Latency tolerant
- More ALUs

Two Memory Spaces



- CPU has deep caches
- GPU has more parallelism

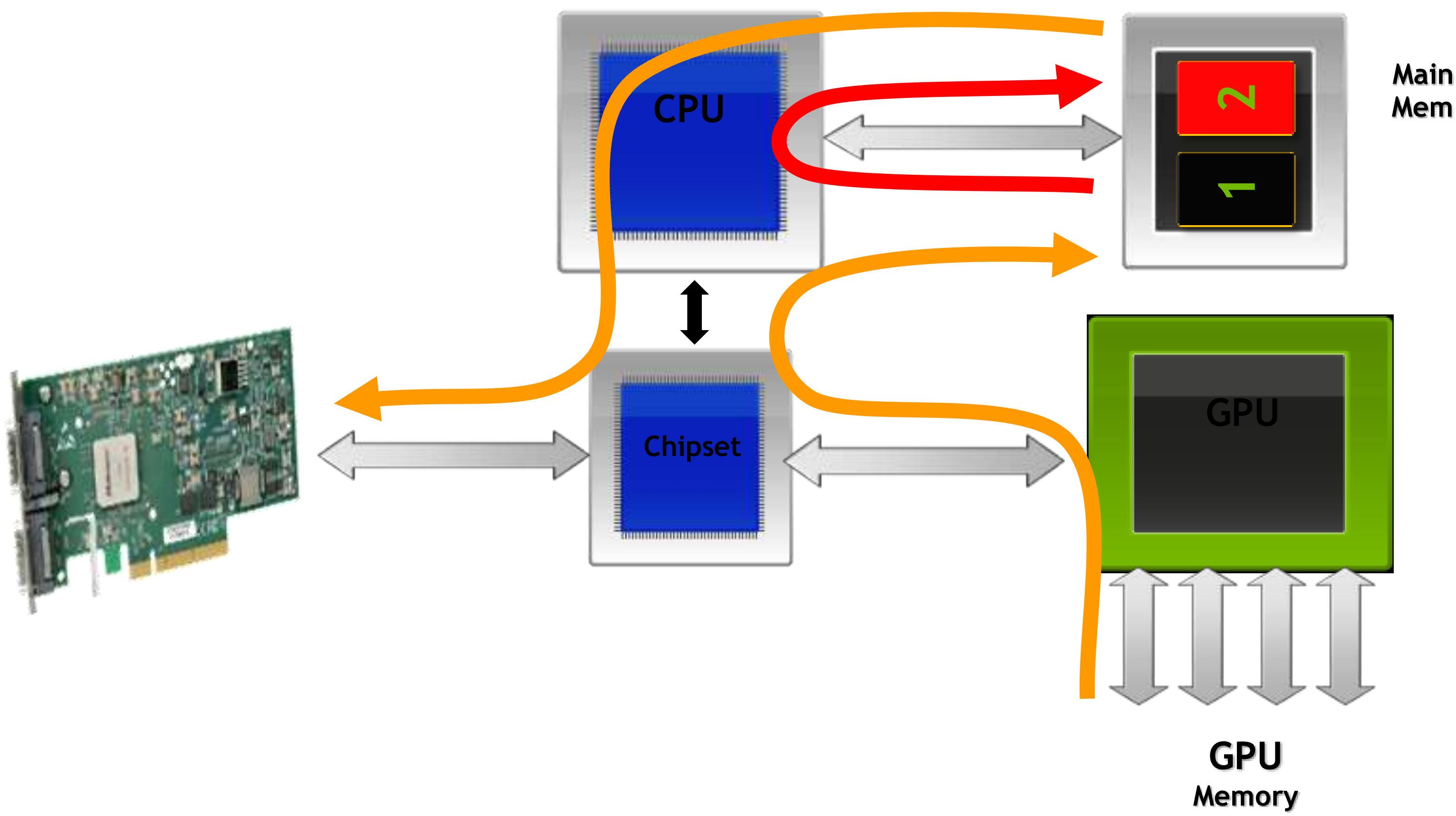
But:

- Message Passing happens in the CPU
- GPUs have most of the FLOPs

The DMA/RDMA Problem

- CUDA driver allocates its own pinned memory region for DMA transfers to/from GPU
- IB driver allocates its own pinned memory region for RDMA transfers to/from IB card
- GPU can only access system memory
- IB can only access system memory
- MPI stack has no knowledge of GPU

MPI and CUDA Before GPUDirect



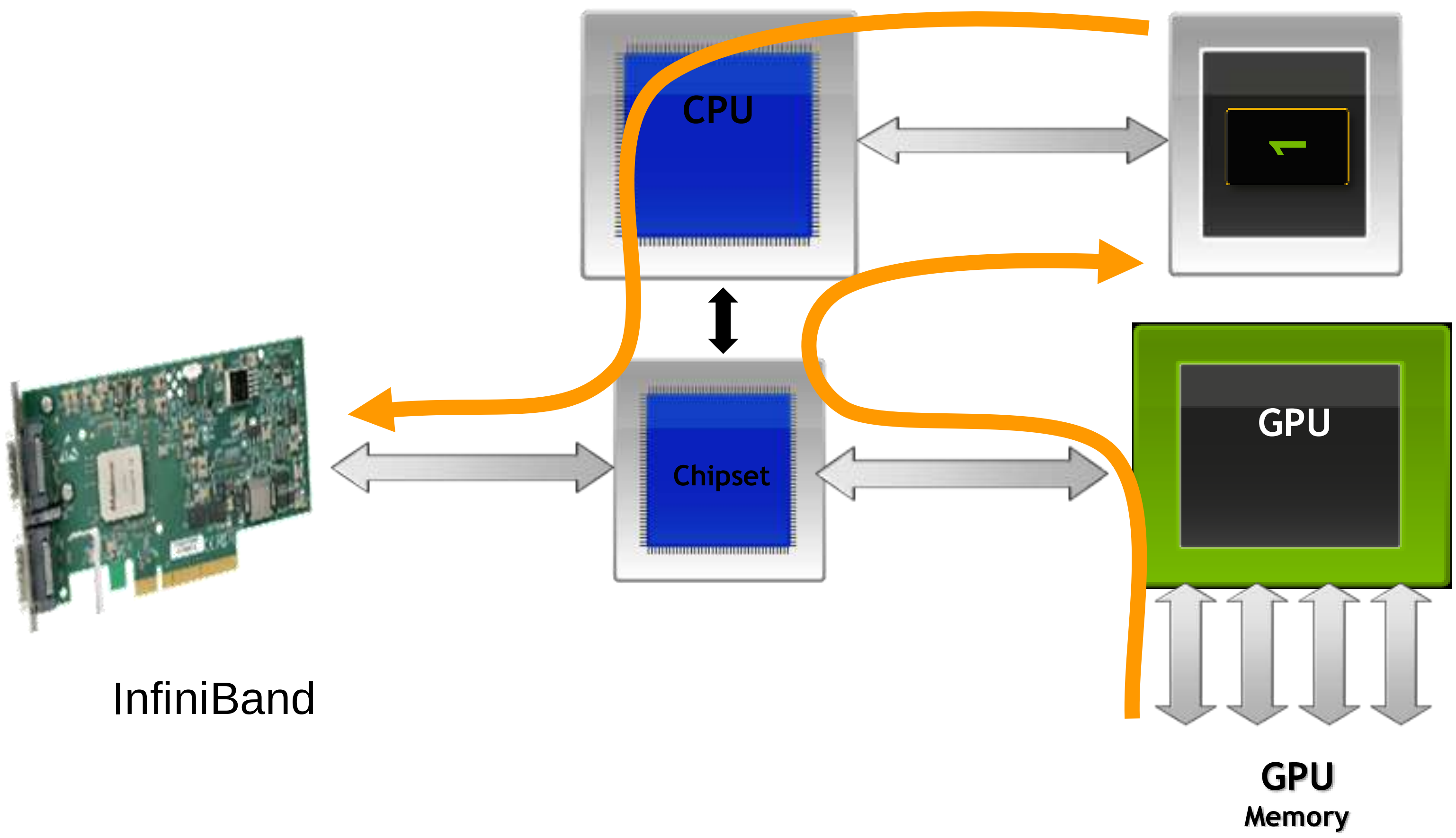
What is GPUDirect?

- GPUDirect is an umbrella term for improving interoperability with third-party devices
  - Especially cluster fabric hardware
- Long-term goal is to reduce dependence on CPU for managing transfers
- Contains both programming model and system software enhancements
- Linux only (for now)

GPUDirect v1

- Jointly developed with Mellanox
- Enables IB driver and CUDA driver to share the same pinned memory
- Eliminates CPU memcpy()s
- Kernel patch for additional kernel mode callback
- Guarantees proper cleanup of shared physical memory at process shutdown
- Currently shipping

GPUDirect v1



CUDA 4.0 Enhancements

No-copy Pinning of System Memory

- Reduce system memory usage and CPU memcpy() overhead
  - Easier to add CUDA acceleration to existing applications
  - Just register malloc'd system memory for async operations and then call cudaMemcpy() as usual

Before No-copy Pinning	With No-copy Pinning
Extra allocation and extra copy required	Just register and go!
malloc(a)	
cudaMallocHost(b) memcpy(b, a)	cudaHostRegister(a)
cudaMemcpy() to GPU, launch kernels, cudaMemcpy() from GPU	
memcpy(a, b) cudaFreeHost(b)	cudaHostUnregister(a)

- All CUDA-capable GPUs on Linux or Windows
  - Requires Linux kernel 2.6.15+ (RHEL 5)

Unified Virtual Addressing

- 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)

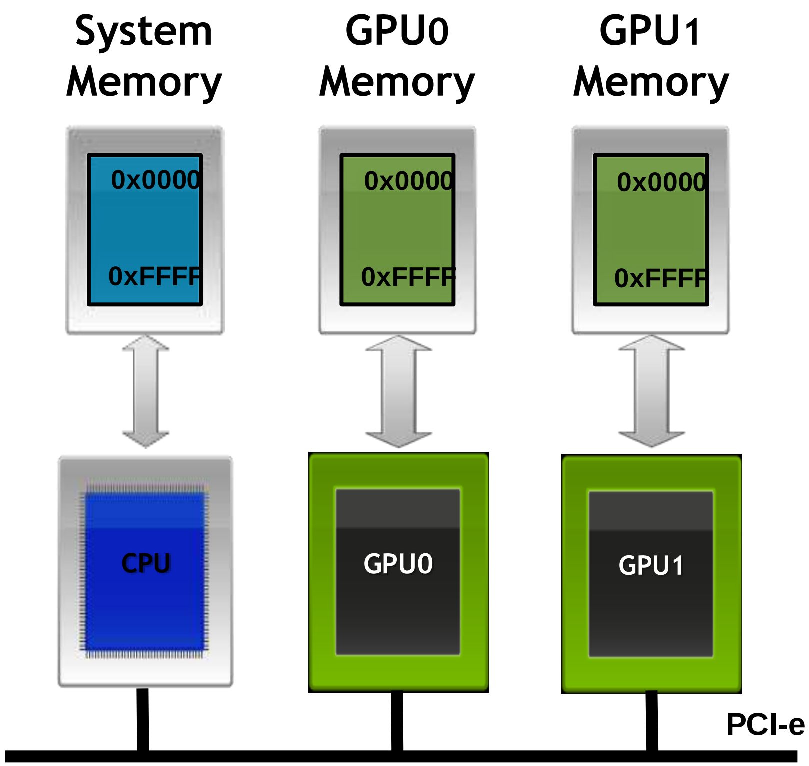
Before UVA	With UVA
Separate options for each permutation	One function handles all cases
cudaMemcpyHostToHost cudaMemcpyHostToDevice cudaMemcpyDeviceToHost cudaMemcpyDeviceToDevice	cudaMemcpyDefault (data location becomes an implementation detail)

- Supported on Tesla 20-series and other Fermi GPUs

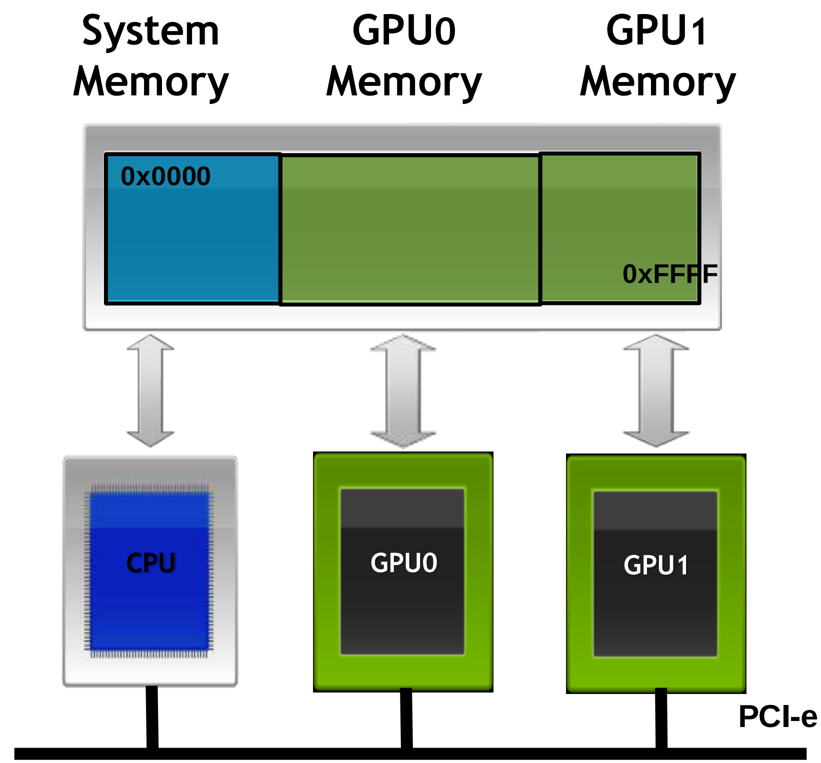
Unified Virtual Addressing

Easier to Program with Single Address Space

No UVA: Multiple Memory Spaces



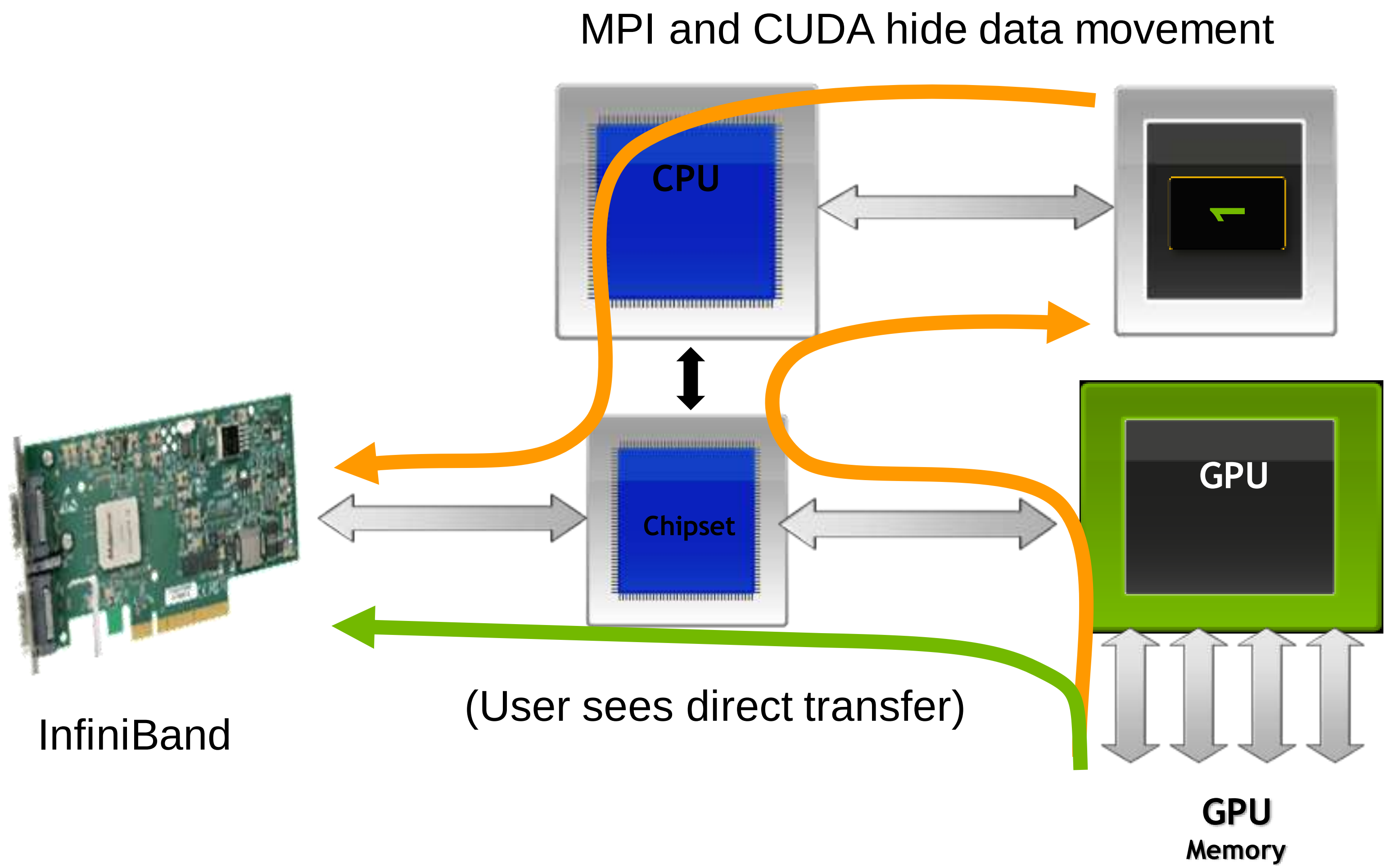
UVA : Single Address Space



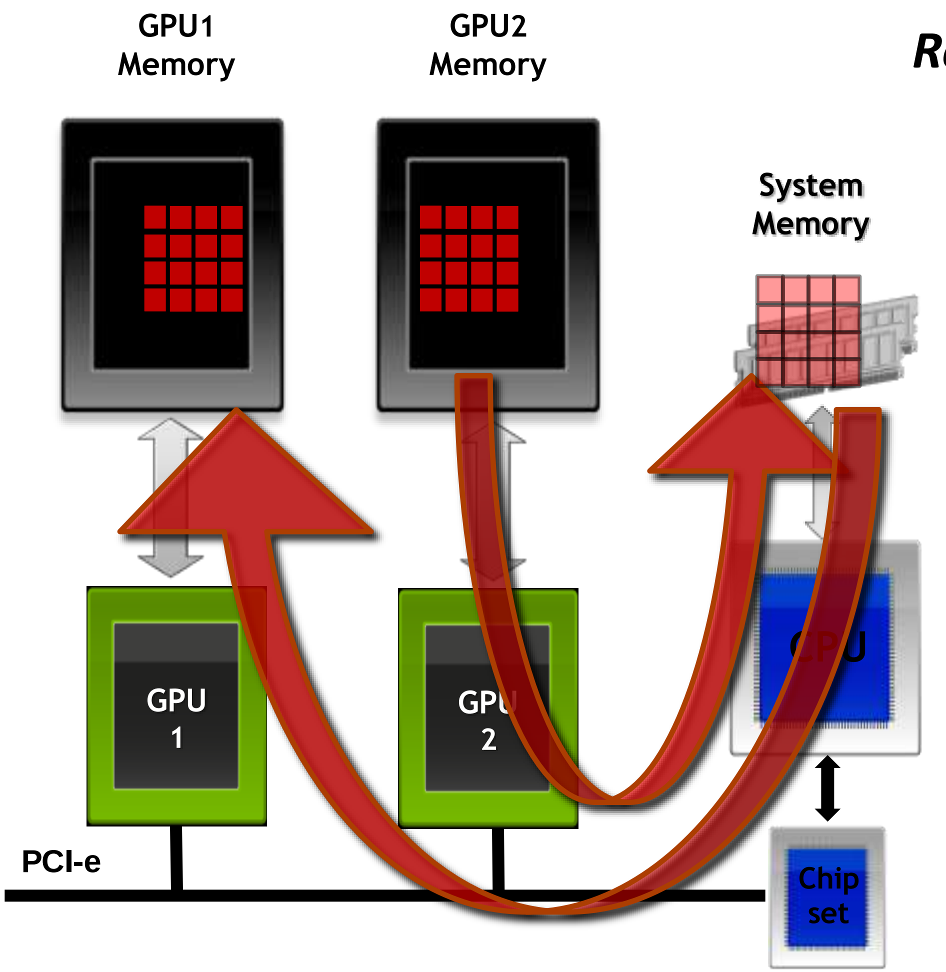
GPUDirect v2

- Uses UVA
- GPU Aware MPI
  - MPI calls handle both GPU and CPU pointers
- Improves programmer productivity
  - Data movement done in SW
  - Same performance as v1
- Requires
  - CUDA 4.0 and unified address space support
  - 64-bit host app and GF100+ only

GPUDirect v2



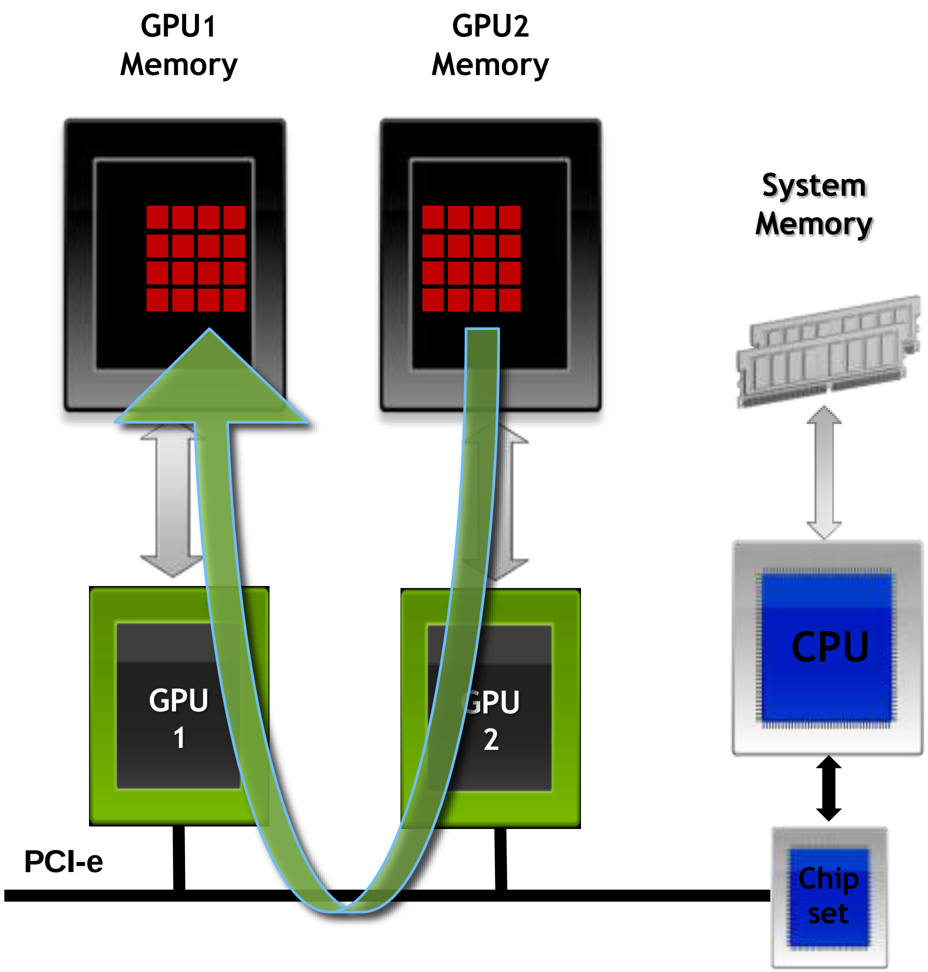
Before NVIDIA GPUDirect™ v2.0



Required Copy into Main Memory

1. cudaMemcpy(sysmem, GPU2)
2. cudaMemcpy(GPU1,sysmem)

NVIDIA GPUDirect™ v2.0:
Peer-to-Peer Communication



Direct Transfers between GPUs

1. cudaMemcpy(GPU1, GPU2)

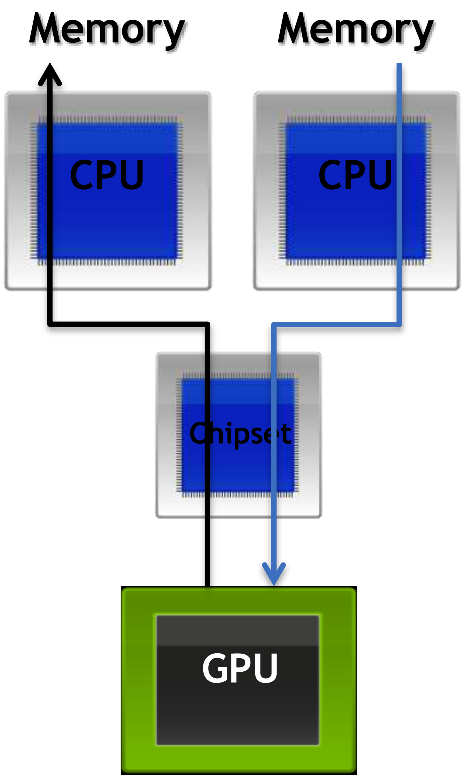
GPUDirect v2.0: Peer-to-Peer Communication

- Direct communication between GPUs
  - Faster - no system memory copy overhead
  - More convenient multi-GPU programming
- Direct Transfers
  - Copy from GPU_0 memory to GPU_1 memory
  - Works transparently with UVA
- Direct Access
  - GPU_0 reads or writes GPU_1 memory (load/store)
- Supported only on Tesla 20-series (Fermi)
  - 64-bit applications on Linux and Windows TCC

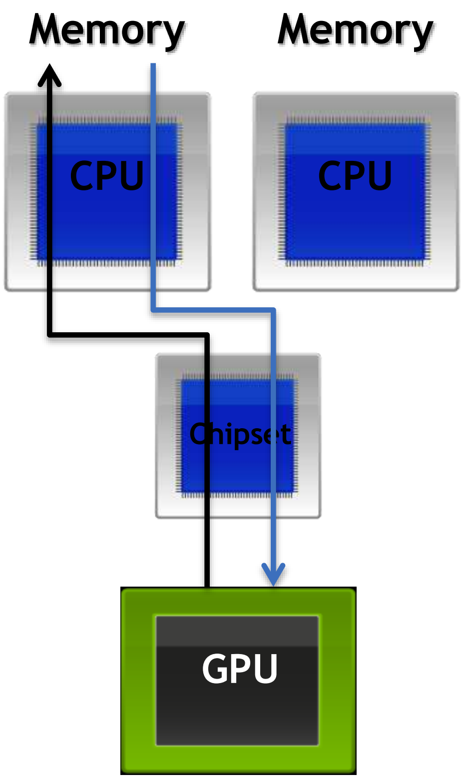
GPUDirect Future Directions

- P2P protocol could be extended to other devices
  - Network cards
  - Storage devices (flash?)
  - Other?
- Extended PCI topologies
- More GPU autonomy
- Better NUMA topology discovery/exposure

Topology

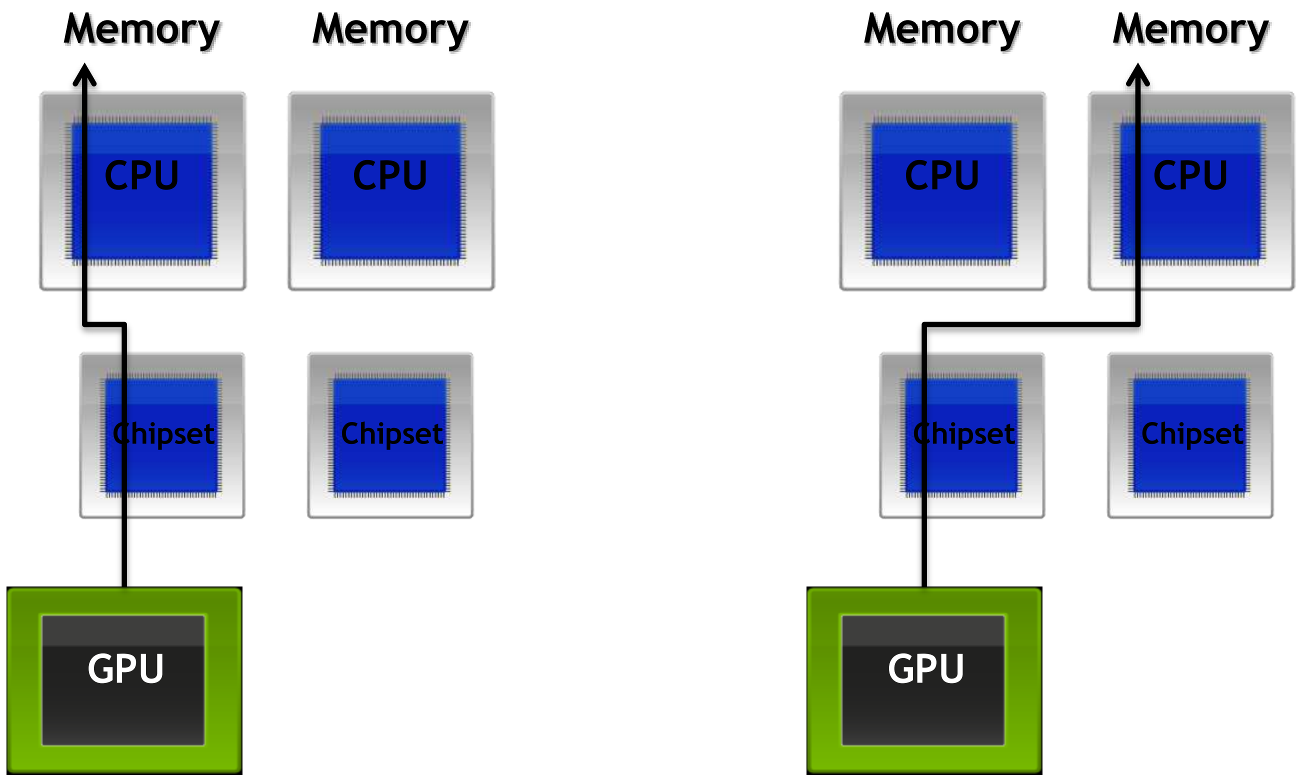


11.0 GB/s



7.4 GB/s

And More Topology



6.5 GB/s

4.34 GB/s

GPU Technology Conference 2011

October 11-14 | San Jose, CA

OPENFABRICS ALLIANCE

The one event you can't afford to miss

- Learn about leading-edge advances in GPU computing
- Explore the research as well as the commercial applications
- Discover advances in computational visualization
- Take a deep dive into parallel programming

Ways to participate

- Speak – share your work and gain exposure as a thought leader
- Register – learn from the experts and network with your peers
- Exhibit/Sponsor – promote your company as a key player in the GPU ecosystem

www.gputechconf.com