



Considerations for Scalable Heterogeneous Computing with GPUs

Philip C. Roth, Kyle L. Spafford, Jeremy S. Meredith, Jeffrey S. Vetter



Why Heterogeneous Systems?

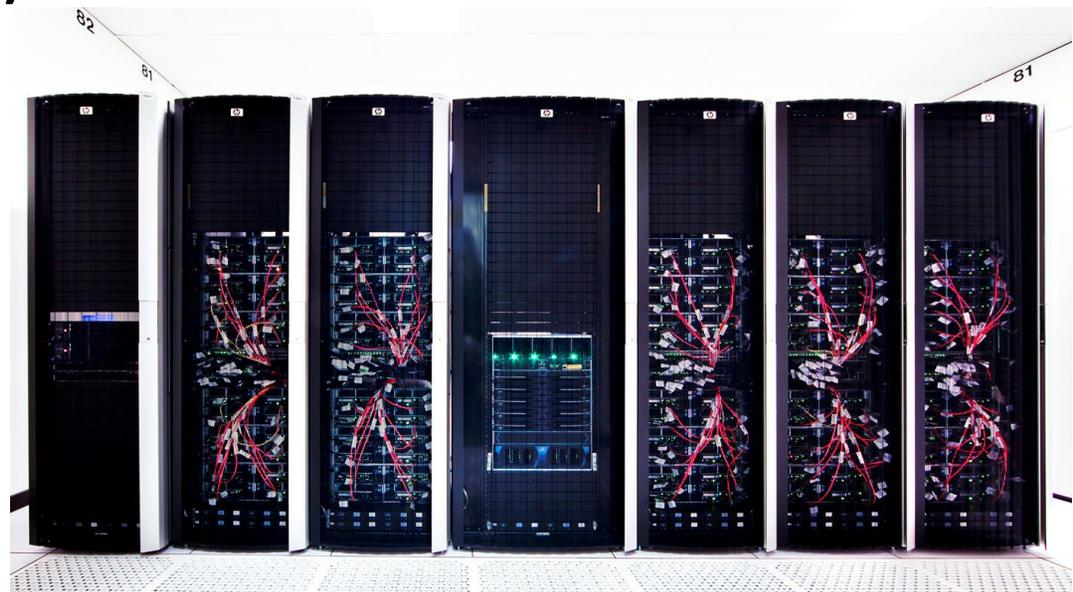
- Traditional single-core CPUs were reaching thermal limits when increasing clock rates
- Multi-core allows continued performance improvements, but only so much
- Specialized devices can give higher performance, better energy efficiency
- Still need general purpose CPUs for parts of the code that cannot run on specialized devices

⇒ Heterogeneous systems – the best of both worlds



Heterogeneous Systems for Scientific Computing

- Several examples of high-profile heterogeneous systems
 - Tianhe-1A
 - Tsubame
 - Dirac
 - Lincoln
 - Roadrunner
 - *Keeneland*



Keeneland ID at ORNL

- Many augment CPUs with GPUs

GPU Advantages

- GPU-based systems have lots of performance potential
 - High computational rate: lots of concurrency
 - High internal memory bandwidth: needed to feed all those thread contexts
 - High energy efficiency: requires less energy per flop than traditional CPUs
 - High spatial density: requires fewer nodes than CPU-based approaches to reach given performance level
- Keeneland: 201 Tflop/s peak in 7 racks

So What's the Problem?

- Not necessarily easy to realize the potential
 - Programs must expose parallelism at many levels
 - Developers must take node and system architecture into account, not just GPU
- GPUs connected to system nodes via a relatively narrow link
 - PCIe Gen 2 theoretical peak unidirectional bandwidth (x16) is 8.0 GB/s
 - In contrast, Tesla M2070 internal memory bandwidth is 148 GB/s (ECC off)
- Need timely, efficient data transfers across PCIe

It's a Multi-Level Problem

- Scalable, high-performance programs require attention to at least three levels:
 - Within a single process
 - Within a single node
 - Across nodes

Keeneland ID System Architecture

- Traditional cluster
- 120 compute nodes
- 4 login and service nodes
- 4x QDR Infiniband Interconnect
 - Program communication
 - Parallel file system access
- 1Gb Ethernet management network



Multi-Node Programs

- Some problems are too large to fit in memory on one node
- Some problems require more computing power than one node provides
- Systems like Keeneland support multi-node programs, e.g., using MPI

CUDA/OpenCL and MPI

- Neither CUDA nor OpenCL provide any particular support for multi-node programs...
- ...Nor do they get in the way
- CUDA/OpenCL are orthogonal to MPI
- “Easy” to take an MPI program and convert some of it to run on a GPU

It's a Multi-Level Problem

- Scalable, high-performance programs require attention to at least three levels:
 - *Within a single process*
 - Within a single node
 - Across nodes

Single Process Performance

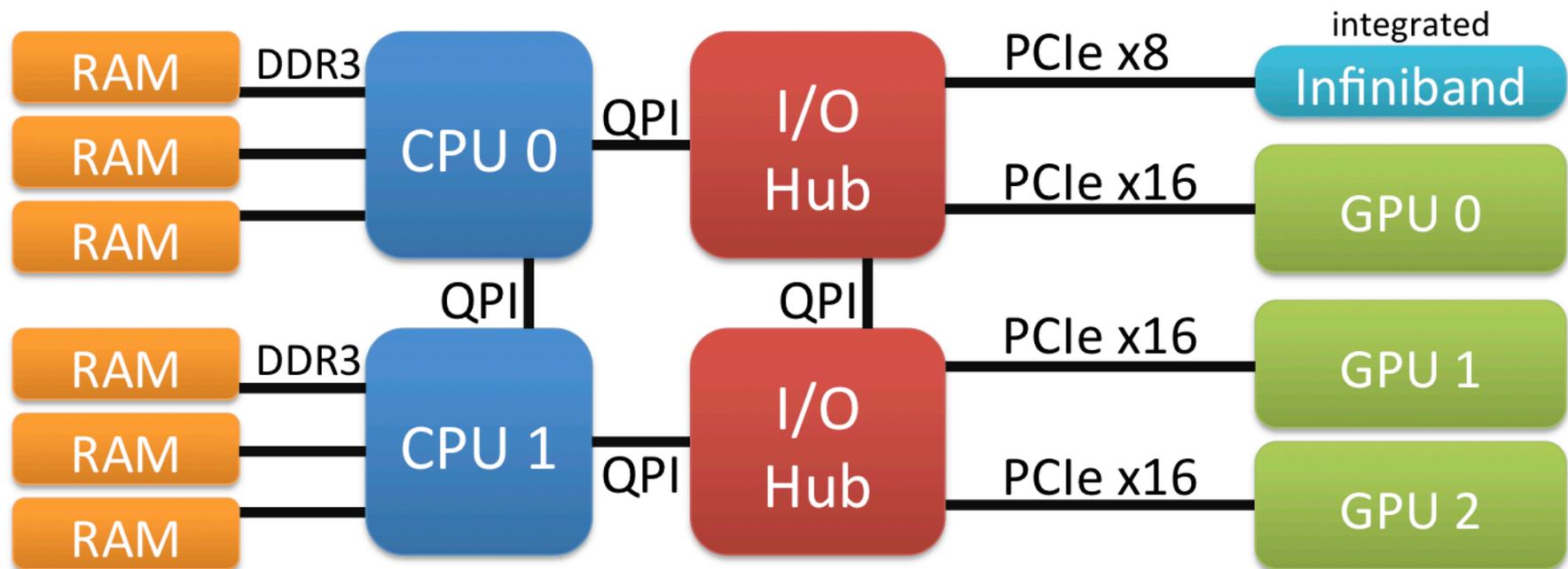
- Lots said elsewhere in the workshop about support for good single process performance
 - Compilers/translators
 - Libraries
 - CUBLAS, CUFFT, NVIDIA Performance Primitives
 - MAGMA
 - Jacket/libJacket
 - Thrust

It's a Multi-Level Problem

- Scalable, high-performance programs require attention to at least three levels:
 - Within a single process
 - *Within a single node*
 - Across nodes

Keeneland ID Node Architecture

- Hewlett Packard ProLiant SL390s G7

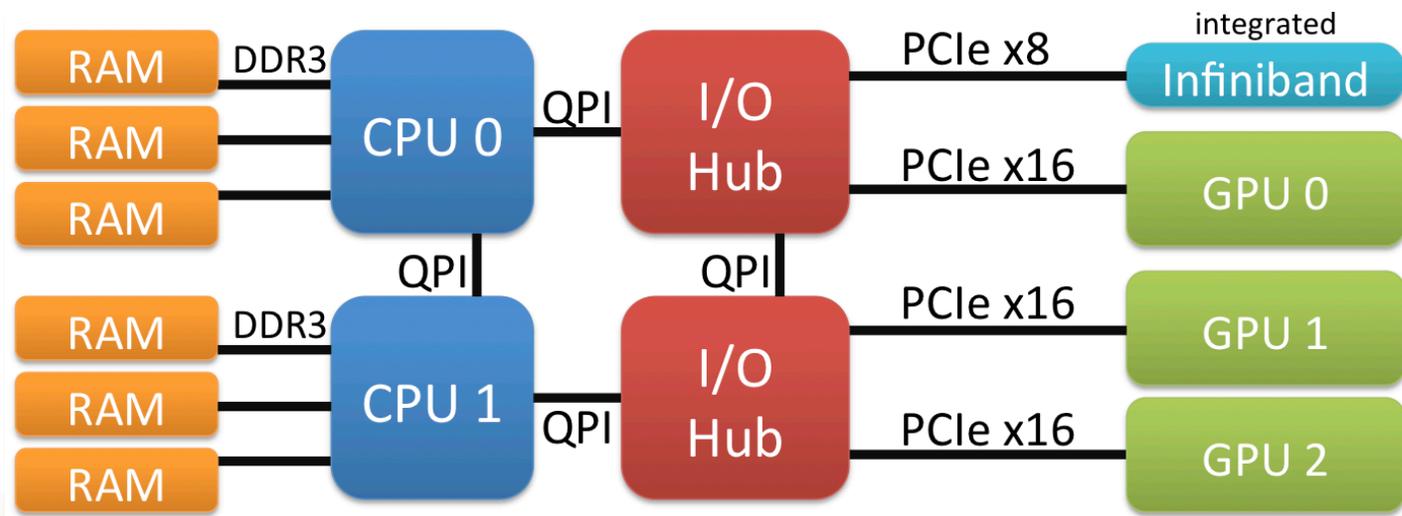


Within a Node

- Avoid data transfers across PCIe whenever possible
 - Transfer data to device and leave it there
 - Don't transfer redundant/excess data (e.g., entire matrix when boundary values are all that is needed for halo exchange)
- Ensure timely, efficient data transfers when necessary

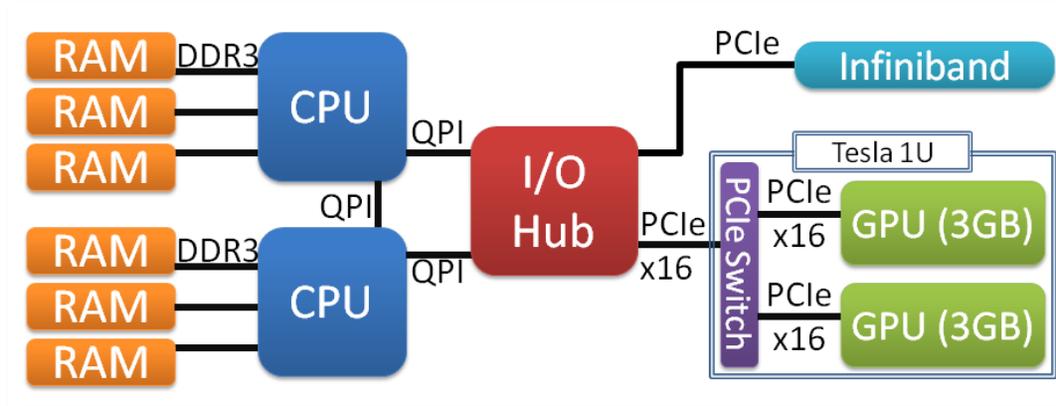
Non-Uniform Memory Access

- Node architectures result in Non-Uniform Memory Access (NUMA)
 - Point-to-point connections between devices
 - Not fully-connected topologies
 - Host memory connected to sockets instead of across a bus



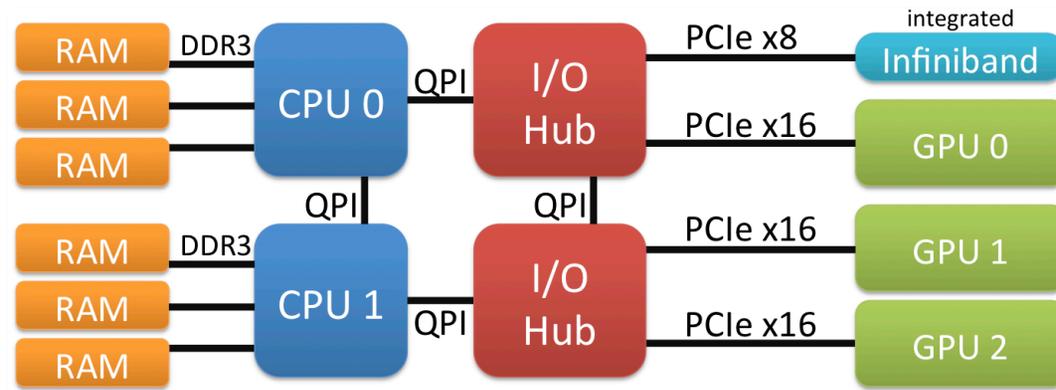
NUMA Can Affect GPUs and Network Too

Older node architecture with single I/O hub but no NUMA effects between CPU and GPU/HCA



- DL160
- Single I/O Hub
- PCIe switch connects GPUs

Keeneland node architecture with dual I/O hub but NUMA effects



- SL390
- Dual I/O Hub
- No PCIe switch

NUMA Control Mechanisms

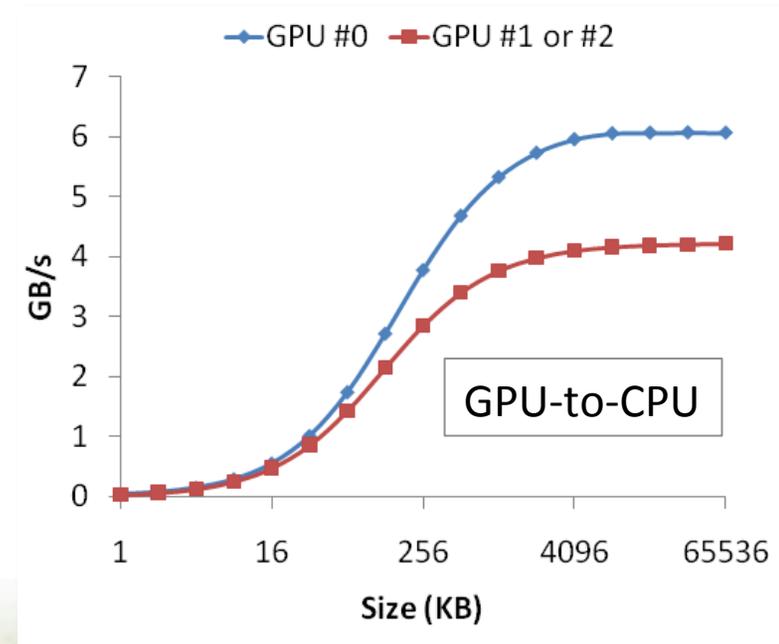
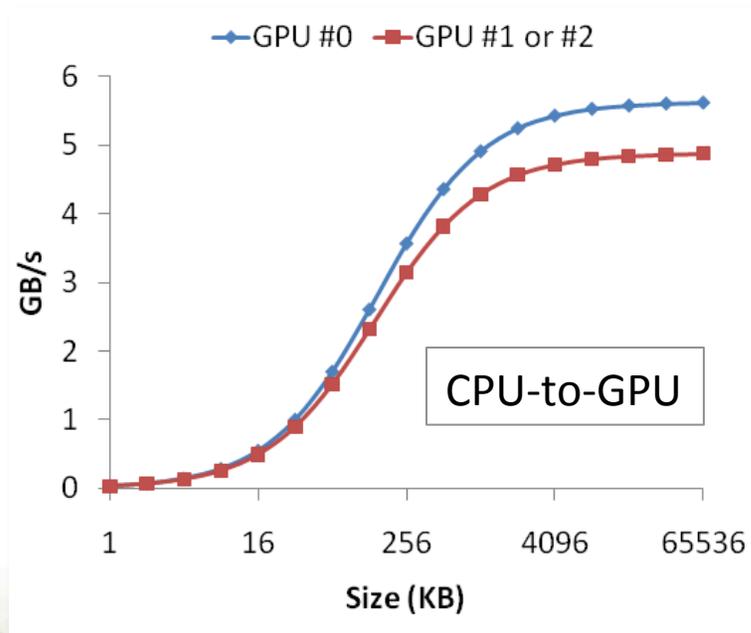
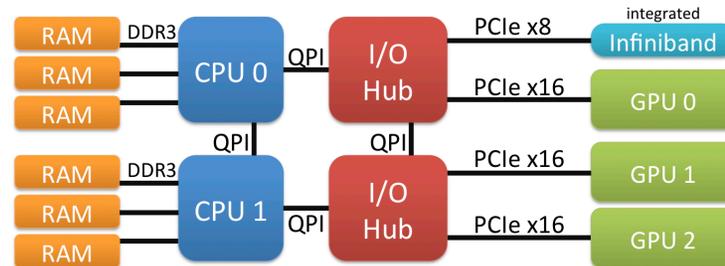
- Process, data placement tools:
 - Tools like libnuma and numactl
 - Some MPI implementations have NUMA controls built in (e.g., Intel MPI, OpenMPI)

How much Does NUMA Impact Performance?

- Microbenchmarks to focus on individual node components
- Macrobenchmarks to focus on individual operations and program kernels
- Full applications to gauge end-user impact

Data Transfer Bandwidth

- Measured bandwidth of data transfers between CPU socket 0 and the GPUs



SHOC Benchmark Suite

- What penalty for “incorrect” mapping?
- Rough inverse correlation to computational intensity

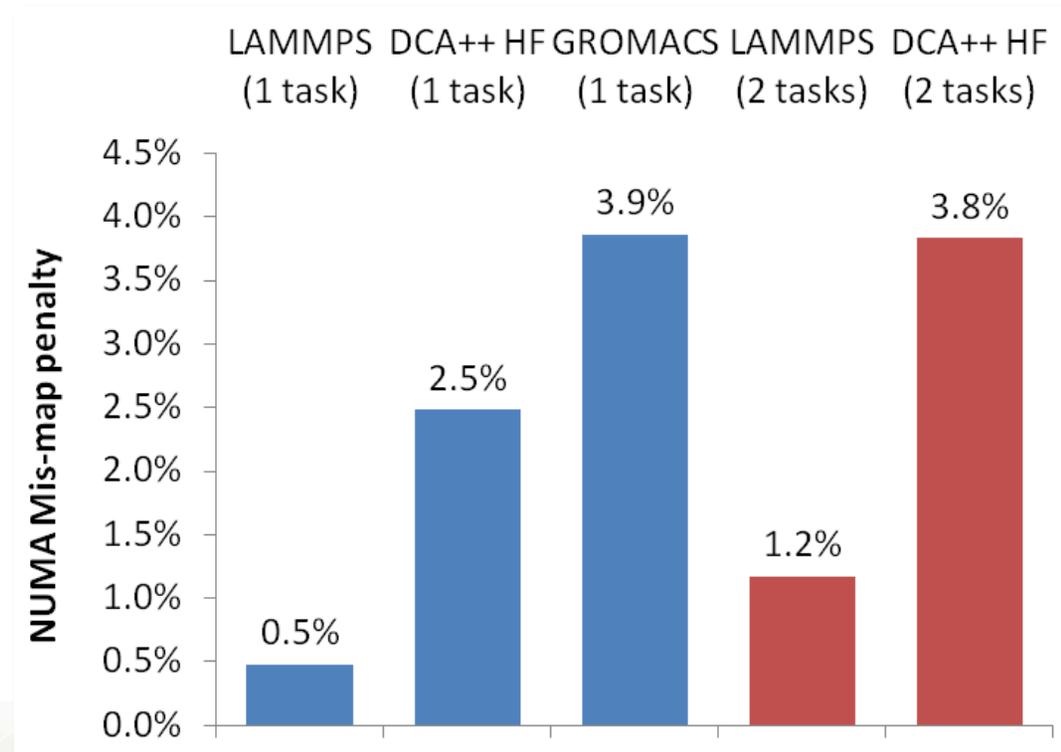
Test	Units	Correct NUMA	Incorrect NUMA	% Penalty
SGEMM	GFLOPS	535.640	519.581	3%
DGEMM	GFLOPS	239.962	230.809	4%
FFT	GFLOPS	30.501	26.843	12%
FFT-DP	GFLOPS	15.181	13.352	12%
MD	GB/s	12.519	11.450	9%
MD-DP	GB/s	19.063	17.654	7%
Reduction	GB/s	5.631	4.942	12%
Scan	GB/s	0.007	0.005	31%
Sort	GB/s	1.081	0.983	9%
Stencil	seconds	8.749	11.895	36%

Table 3: SHOC Benchmark Results



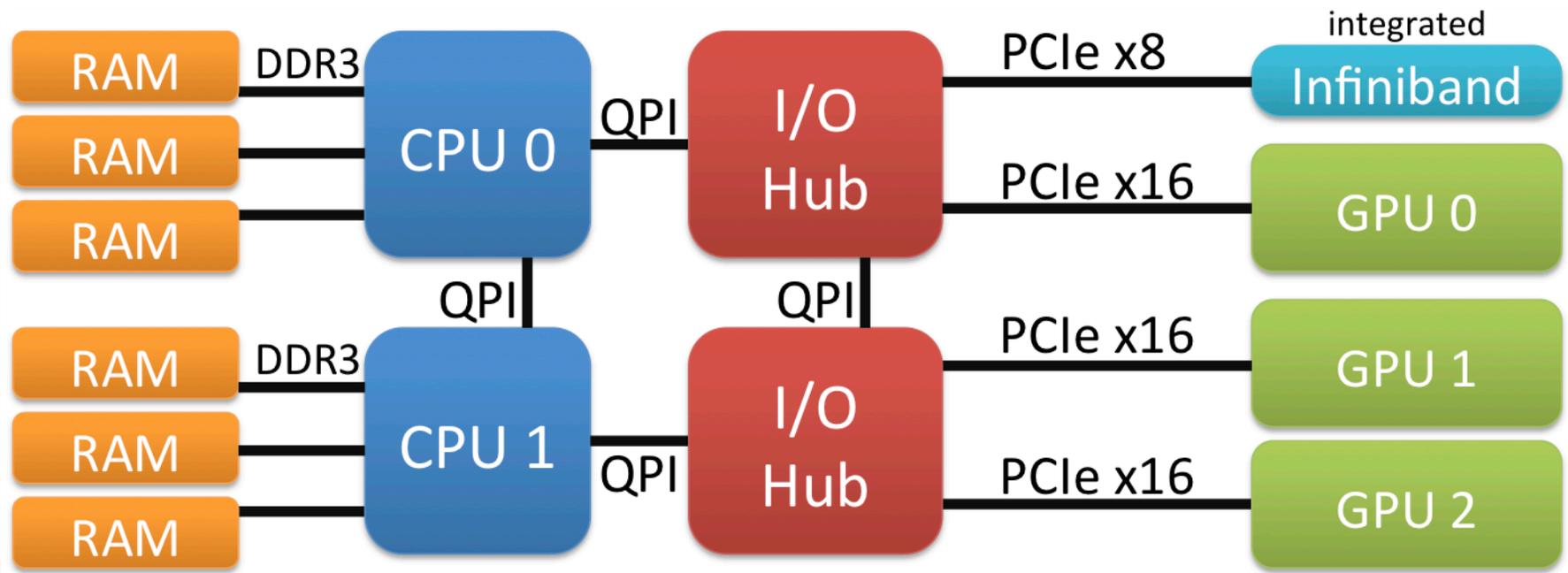
Full Applications

- With one application task, performance penalty for using incorrect mapping (e.g., CPU socket 0 with GPU 1)
- With two application tasks, performance penalty for using mapping that uses “long” paths for both (e.g., CPU socket 0 with GPU 1 and CPU socket 1 with GPU 0)



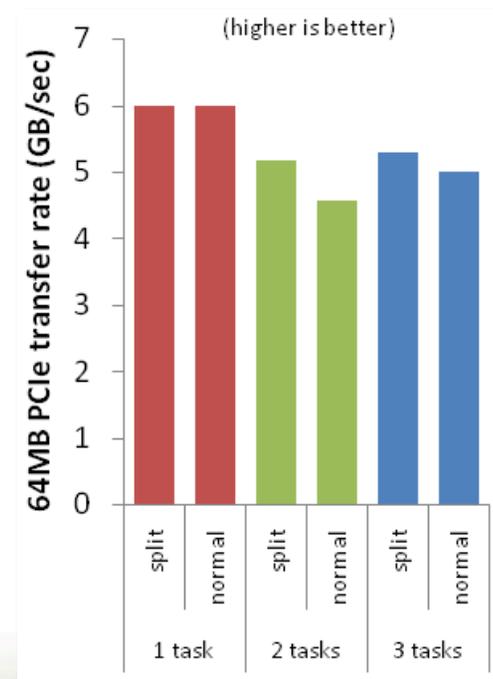
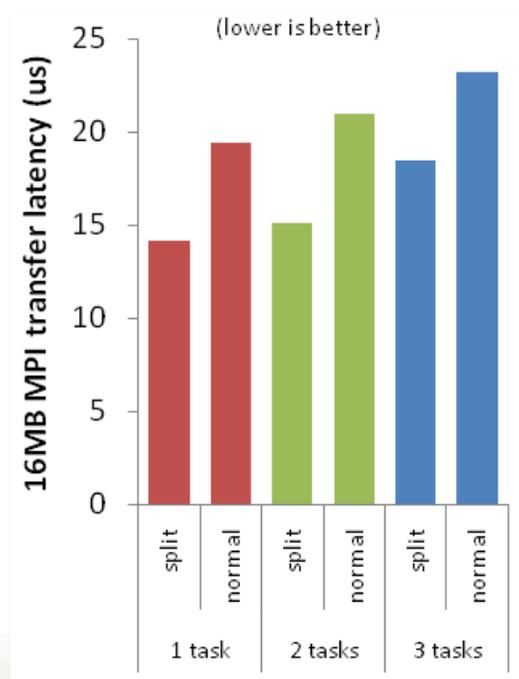
NUMA and Network Traffic

- Have to worry about not only process/data placement for CPU and GPU, but also about CPU and Infiniband HCA



Thread Splitting

- Instead of 1 thread that controls a GPU and issues MPI calls, split into two threads and bind to appropriate CPU sockets



It's a Multi-Level Problem

- Scalable, high-performance programs require attention to at least three levels:
 - Within a single process
 - Within a single node
 - *Across nodes*

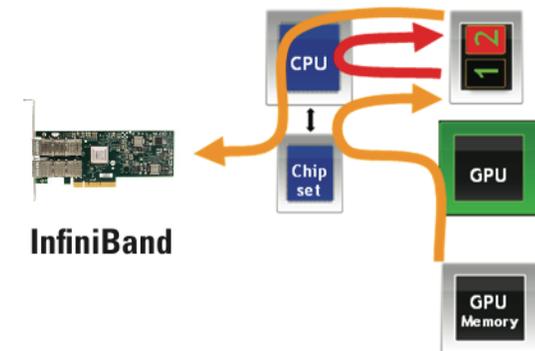
Interconnection Network

- System nodes are connected by an interconnection network, or interconnect
- Infiniband is a common interconnect for mid-to high-end systems
- Keeneland ID uses 4x QDR Infiniband, with 40Gb/s maximum bandwidth



The Problem

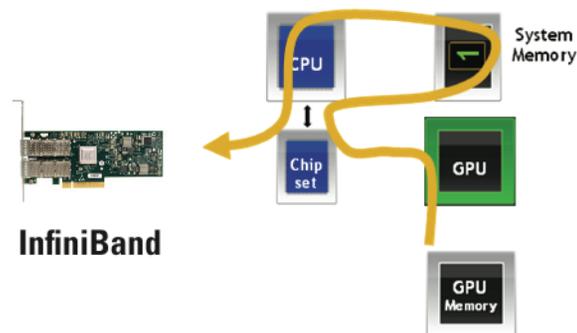
- Data is in device memory of GPU on one node, needs to be transferred to device memory of GPU on another node
- Several hops:
 - Data transferred from GPU memory to GPU buffer in host memory
 - Data copied from GPU buffer to IB buffer in host memory
 - Data read by IB HCA using RDMA transfer
 - Repeat in reverse on other end



http://www.mellanox.com/pdf/whitepapers/TB_GPU_Direct.pdf

GPUDirect

- NVIDIA and Mellanox developed an approach for allowing others to access the GPU buffer in host memory
- Eliminates the data copy from GPU buffer to IB buffer
 - Eliminates two system memory data copy operations (one on each end)
 - Keeps host CPU out of the data path
 - Up to 30% performance improvement (according to NVIDIA)



http://www.mellanox.com/pdf/whitepapers/TB_GPU_Direct.pdf

GPUDirect Status

- Initially, GPUDirect packaging made deploying it unattractive
 - Required running a specific kernel version, with RPMs provided by Mellanox
 - Kernel version was quite old
 - Concerns about keeping updated to avoid security risks
- Mellanox now makes kernel patches available for some Enterprise distributions
- Mellanox working on getting GPUDirect support included as part of stock Linux kernel

GPUDirect on Keeneland

- GPUDirect not (yet) deployed on Keeneland
 - Keeneland runs CentOS 5.5
 - That distribution requires patched kernel for PAPI
 - Have not yet tested patches for PAPI with GPUDirect patches

Back to the Node: GPUDirect 2.0

- GPUDirect 1.0 improves performance for data transfers over an interconnect
- GPUDirect 2.0 improves performance for data transfers between two GPUs in the same node
- Coming in CUDA 4.0

GPUDirect 2.0

- Old way:
 - Copy data from GPU 1 to host memory
 - Copy data from host memory to GPU 2
- New way:
 - Copy data from GPU 1 to GPU2 without host CPU involvement
- Integrates well with Unified Virtual Addressing feature (single address space for CPU and 1+ GPUs)

Summary

- Scalable, high performance computing with GPU-based heterogeneous systems is possible
- Requires attention to node-level and system level
- NUMA controls and GPUDirect (both 1 and 2) are keys to good, scalable performance

Useful information

- Tutorial slides at http://ft.ornl.gov/doku/keeneland/keeneland_tutorial_14_april_2011/start
- Log into Keeneland ID system with `ssh -Y username@kidlogin.nics.utk.edu` (replace 'username' with your username)
- Information about Keeneland and getting a Keeneland account is available at <http://keeneland.gatech.edu>

For more information

- <http://keeneland.gatech.edu>

