

# Current R&D hands on work for pre-Exascale HPC systems

Joshua Mora, PhD

## Agenda

- Heterogeneous HW flexibility on SW request.
  - PRACE 2011 prototype, 1TF/kW efficiency ratio.
- Performance debugging of HW and SW on multi-socket, multi-chipset, multi-GPUs, multi-rail (IB).
  - Affinity/binding.
- In background: [c/nc] HT topologies, I/O performance implications, performance counters, NIC/GPU device interaction with processors, SW stacks.

**If (HPC==customization) {Heterogeneous HW/SW flexibility;}  
else {Enterprise solutions;}**

**On the HW side, customization is all about**

- Multi-socket systems, to provide cpu computing density (ie. aggregated G[FLOP/Byte]/s), and memory capacity.**
- Multi-chipset systems, to hook multiple GPUs (to accelerate computation) and NICs (tightly coupled processors/gpus across computing nodes).**

**But**

- Pay attention, among other things, to [nc/c]HT topologies to avoid running out of bandwidth (ie. resource limitations) between devices when the HPC application starts pushing to the limits in all directions.**

# Example: RAM NFS (RNANETWORKS) over 4 QDR IB cards

## FAT NODE in 4U

- 8xSix-core @ 2.6GH
- 256GB RAM @ DDR2-667
- 4xPClegen2
- 4xQDR IB , 2x IB ports
- RNAcache appliance

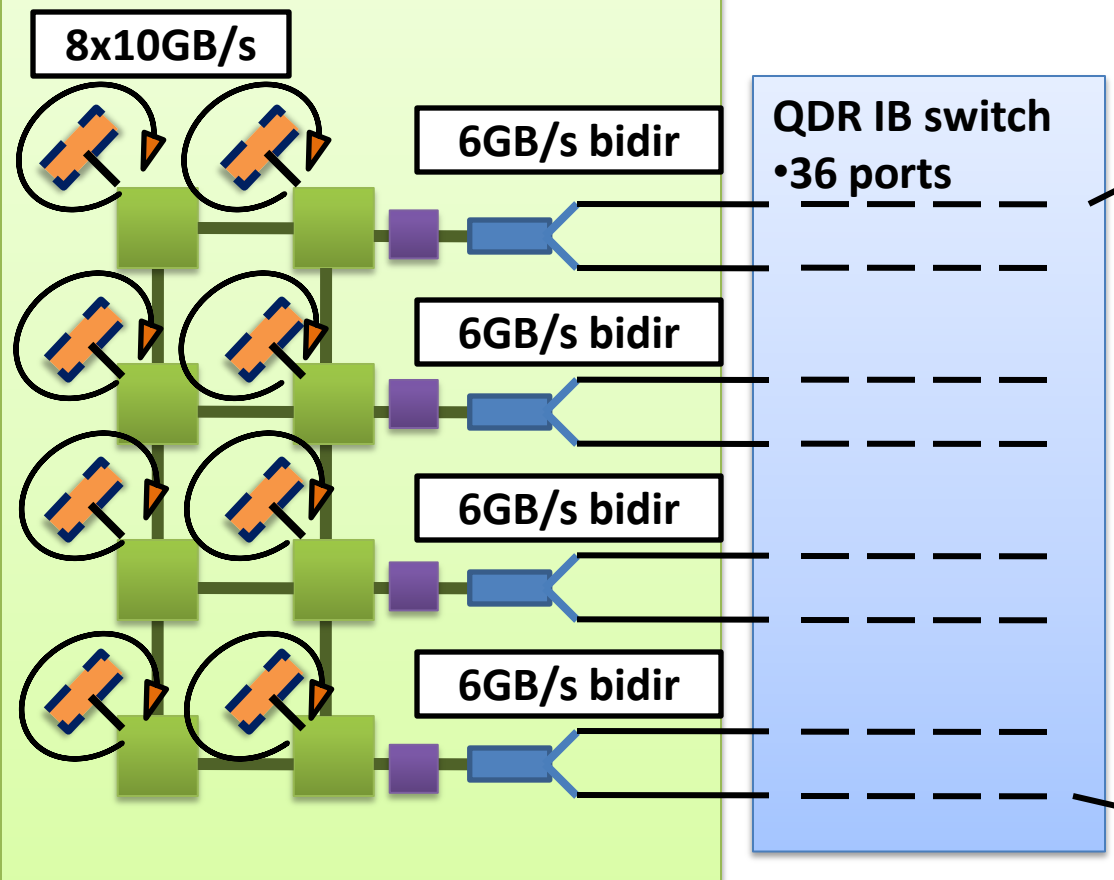
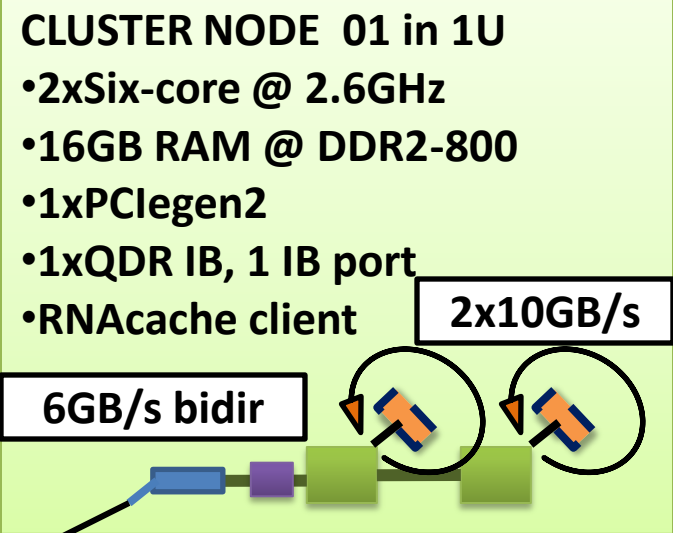
Ultra fast  
local/swap/shared  
cluster file system  
Reads/Writes @  
20GB/sec  
aggregated

### CLUSTER NODE 01 in 1U

- 2xSix-core @ 2.6GHz
- 16GB RAM @ DDR2-800
- 1xPClegen2
- 1xQDR IB, 1 IB port
- RNAcache client

2x10GB/s

6GB/s bidir

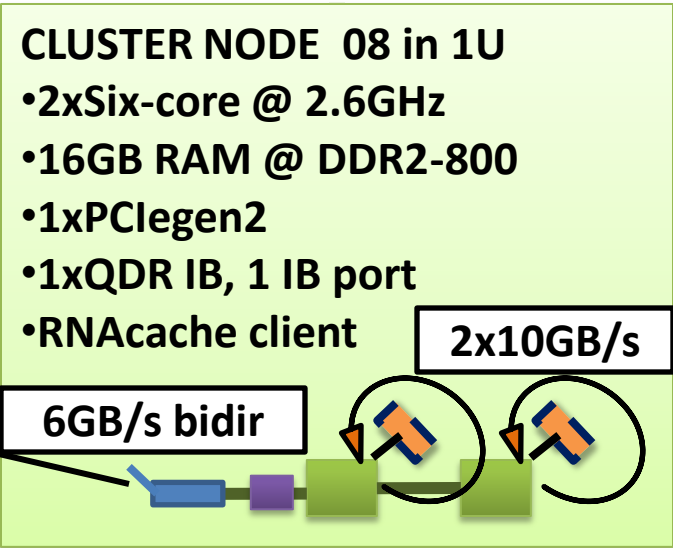


### CLUSTER NODE 08 in 1U

- 2xSix-core @ 2.6GHz
- 16GB RAM @ DDR2-800
- 1xPClegen2
- 1xQDR IB, 1 IB port
- RNAcache client

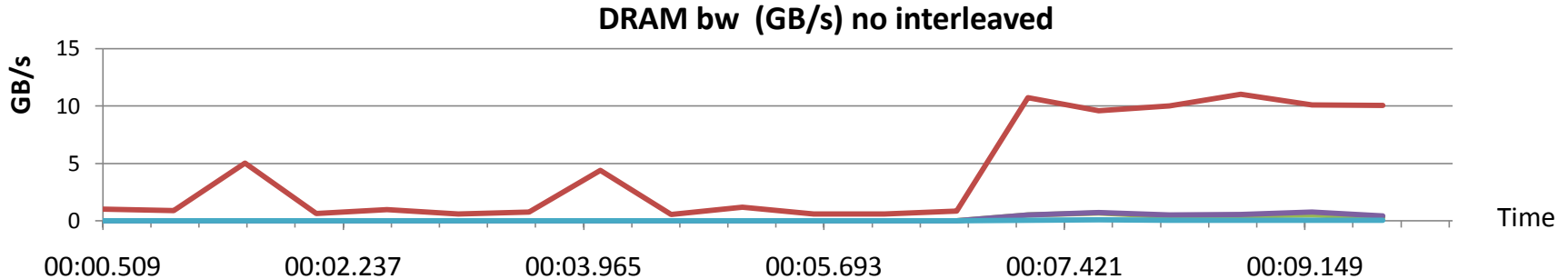
2x10GB/s

6GB/s bidir

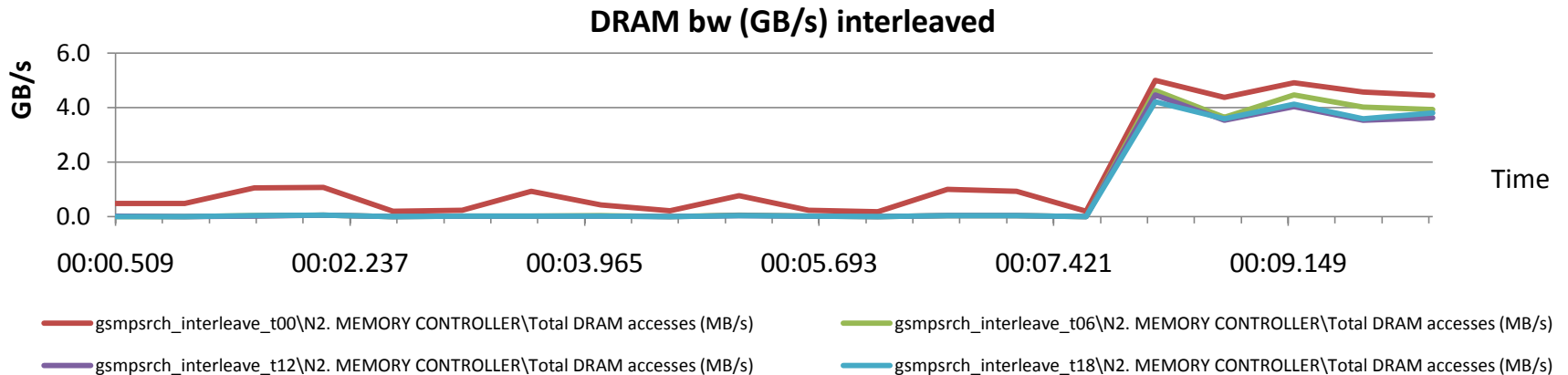


# Example: OpenMP application on NUMA system

Better to configure system as memory node not interleaved ?



Or memory node interleaved enabled ? Huge cHT traffic, limitation.



**Answer : modify application to exploit NUMA system by allocating local arrays after threads have started.**

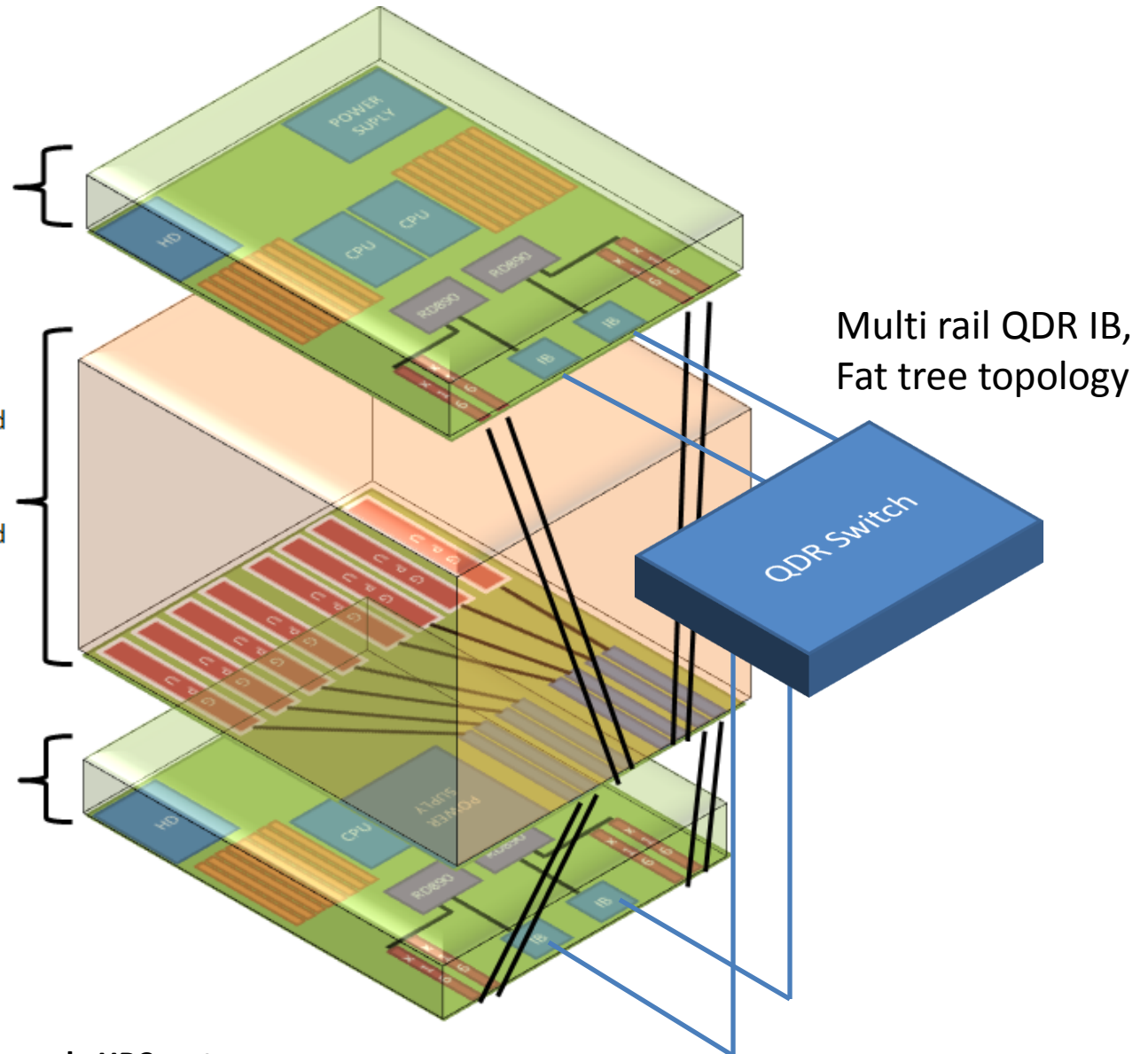
# PRACE 2011 prototype, HW description

1U compute node with  
2P G34 16c 2.3GHz  
+ 2 RD890 + 2 QDR IB

4U Next IO box with  
4 GPUs PCIe x16 connected  
to top compute node

4 GPUs PCIe x16 connected  
to bottom compute node

1U compute node with  
2P G34 16c 2.3GHz  
+ 2 RD890 + 2 QDR IB



# PRACE 2011 prototype, HW description (cont.)

2P G34 16c 2.3GHz + 2 RD890 + 2 QDR IB
Next IO box
4 GPUs PCIe x16 connected to top compute node
4 GPUs PCIe x16 connected to bottom compute node
2P G34 16c 2.3GHz + 2 RD890 + 2 QDR IB

- 6U Sandwich unit replicated 6 times within 42U rack.
- Comp.Nodes connected with Multirail IB.
- Single 36 IB port switch (fat tree).
- Management 1Gb/s

Design for CSC PRACE project 2011, by Joshua.Mora@amd.com		DP GFLOPs	Wats	USD	
42 U rack components for > 20TF peak /sqm, 200K USD		total	22700	13000	200000
42	Note: CPUs are Interlagos, GPUs are Kestrel	Linpack estimate	10000	9000	
41					
40	2P G34 16c 2.3GHz + 2 RD890 + 2 QDR IB		294.4	350	5860
39	Next IO box				
38	4 GPUs PCIe x16 connected to top compute node				
37	4 GPUs PCIe x16 connected to bottom compute node		3200	1300	20000
36					
35	2P G34 16c 2.3GHz + 2 RD890 + 2 QDR IB		294.4	350	5860
34	2P G34 16c 2.3GHz + 2 RD890 + 2 QDR IB		294.4	350	5860
33					
32	Next IO box				
31	4 GPUs PCIe x16 connected to top compute node				
30	4 GPUs PCIe x16 connected to bottom compute node		3200	1400	20000
29	2P G34 16c 2.3GHz + 2 RD890 + 2 QDR IB		294.4	350	5860
28	2P G34 16c 2.3GHz + 2 RD890 + 2 QDR IB		294.4	350	5860
27					
26	Next IO box				
25	4 GPUs PCIe x16 connected to top compute node				
24	4 GPUs PCIe x16 connected to bottom compute node		3200	1400	20000
23	2P G34 16c 2.3GHz + 2 RD890 + 2 QDR IB		294.4	350	5860
22	QDR 36p			100	5000
21	1Gb 24p			50	400
20	KVM			100	200
19					
18	2P G34 16c 2.3GHz + 2 RD890 + 2 QDR IB		294.4	350	5860
17	Next IO box				
16	4 GPUs PCIe x16 connected to top compute node				
15	4 GPUs PCIe x16 connected to bottom compute node		3200	1400	20000
14					
13	2P G34 16c 2.3GHz + 2 RD890 + 2 QDR IB		294.4	350	5860
12	2P G34 16c 2.3GHz + 2 RD890 + 2 QDR IB		294.4	350	5860
11					
10	Next IO box				
9	4 GPUs PCIe x16 connected to top compute node				
8	4 GPUs PCIe x16 connected to bottom compute node		3200	1400	20000
7	2P G34 16c 2.3GHz + 2 RD890 + 2 QDR IB		294.4	350	5860
6	2P G34 16c 2.3GHz + 2 RD890 + 2 QDR IB		294.4	350	5860
5					
4	Next IO box				
3	4 GPUs PCIe x16 connected to top compute node				
2	4 GPUs PCIe x16 connected to bottom compute node		3200	1400	20000
1	2P G34 16c 2.3GHz + 2 RD890 + 2 QDR IB		294.4	350	5860

# PRACE 2011 prototype, SW description

- “Plain” C/C++/Fortran (application)
- Pragma directives for OpenMP (multi threading)
- Pragma directives for HMPP (CAPS enterprise) (GPUs)
- MPI (OpenMPI,MVAPICH) (communications)
- GNU/Open64/CAPS (compilers)
- GNU debuggers
- ACML, LAPACK,.. (math libraries)
- “Plain” OS (eg. SLES11sp1, support for Multi-chip Module)
- No need to develop kernels in
  - OpenCL (for ATI cards)
  - CUDA (for NVIDIA cards).

# PRACE 2011 prototype : exceeding all requirements

## Requirements and Metrics

- Budget of 400k EUR
- 20 TFLOP/s peak
- Contained in 1 square meter (ie. single rack)
- 0.6 TFLOP/kW

Metric	Requirement	Proposal	Exceeded
peak TF/s	20	22	yes
m <sup>2</sup>	1	1 (42U rack)	met
TF/kW	0.6	1.1	yes
EUR	400k	250K	yes

- This system delivers real double precision 10TFLOP/s in rack within 10kW, assuming multi-cores only pump in and out data to GPUs.
- 100 racks can deliver 1PFLOP within 1MW, >>20Million EUR.
- Reaching affordability point in private sector (eg. O&G).

# PRACE 2011 prototype : HW flexibility on SW request

## NextIO box:

- PClegen2 switch box/appliance (GPUs, IB, FusionIO,..)
- Connection of x8 and x16 PClegen1/2 devices
- Hot swappable PCI devices
- Dynamically reconfigurable through SW without having to reboot system
- Virtualization capabilities through IOMMU
- Applications can be “tuned” to use more or less GPUs on request:

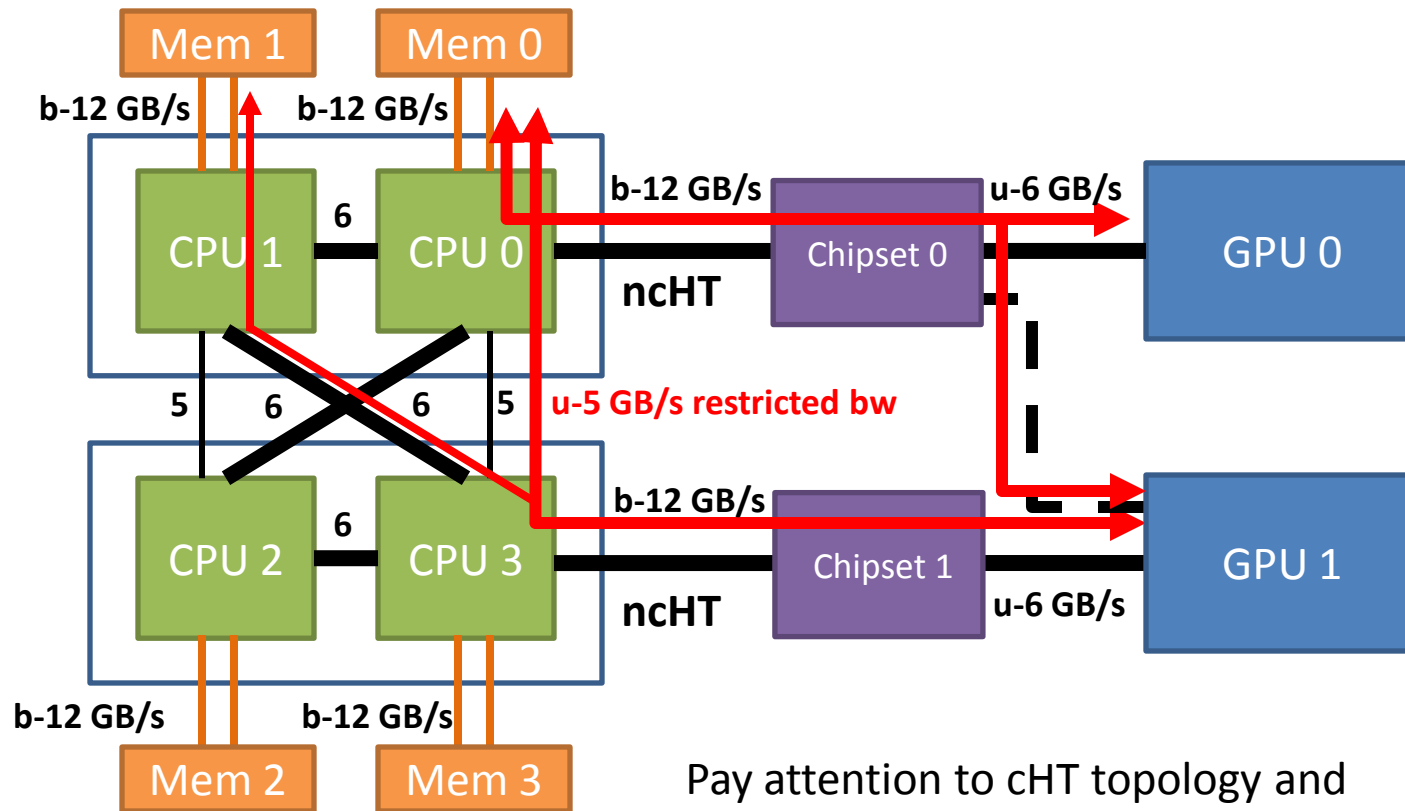
Some GPU kernels are very efficient and the PCI bw requirements are low → Can dynamically allocate more GPUs to fewer compute nodes increasing Performance/Watt.

- Targeting SC10 to do a demo/cluster challenge.

# Performance debugging of HW/SW on multi...you name it.

Example: 2 processor + 2 chipsets + 2 GPUs

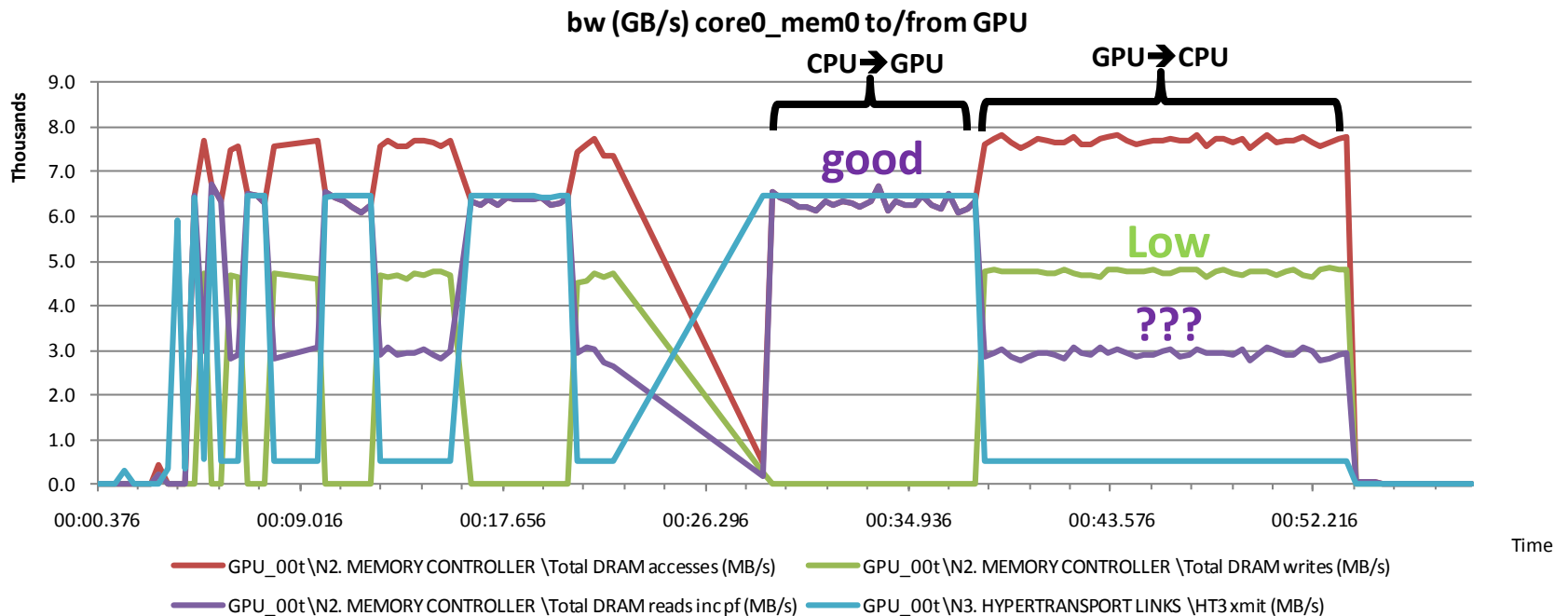
PCISpeedTest available at [developer.amd.com](http://developer.amd.com)



Pay attention to cHT topology and I/O balancing since cHT links can restrict CPU  $\leftrightarrow$  GPU bandwidth

# Performance debugging of HW on multi...you name it.

Problem: CPU → GPU good, GPU → CPU bad, why ?



First phase: CPU → GPU at ~6.4GB/s, memory controller does reads.

Second phase: GPU → CPU at 3.5GB/s, something is wrong.

On second phase: memory controller is doing writes and reads. Reads do not make sense. Memory controller should only do writes.

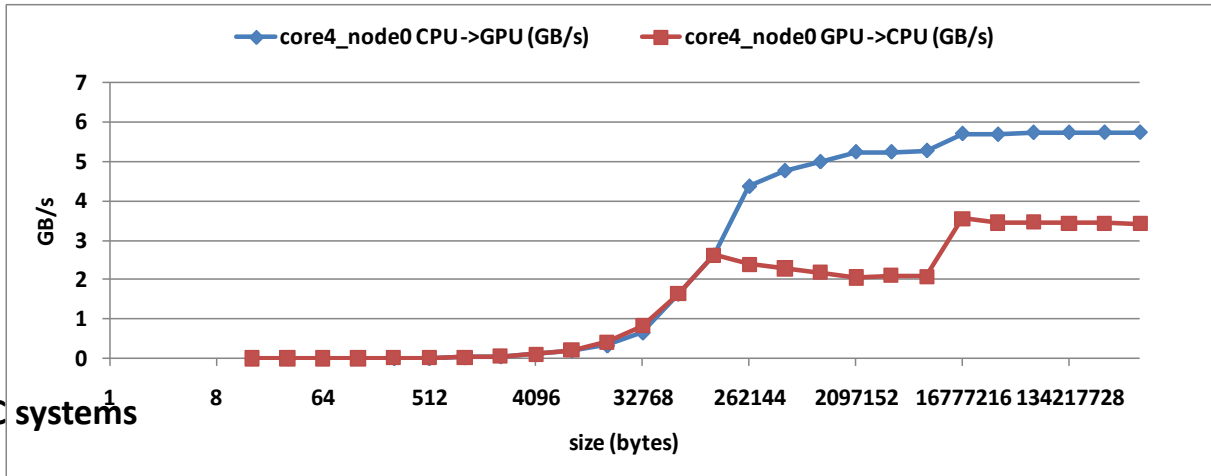
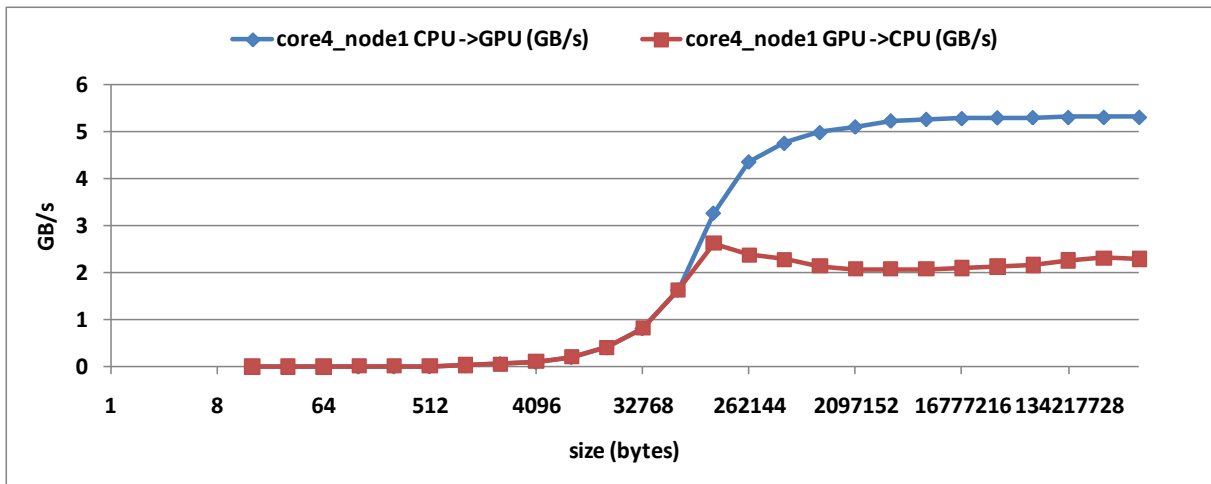
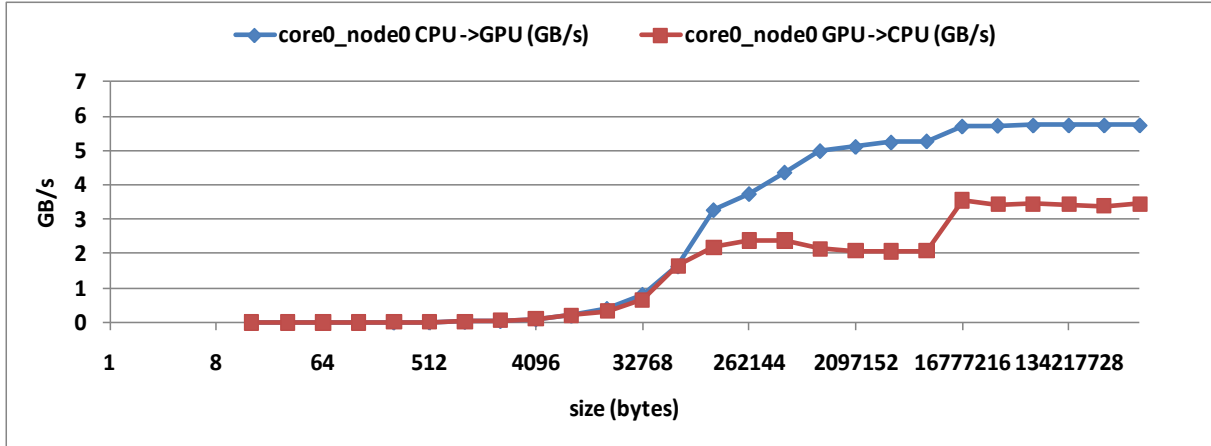
While we figure out why we have reads on MCT from GPU → CPU, lets look at the affinity/binding

- On Node 0 ↔ GPU 0
- GPU 0 → CPU 0, u-3.5GB/s

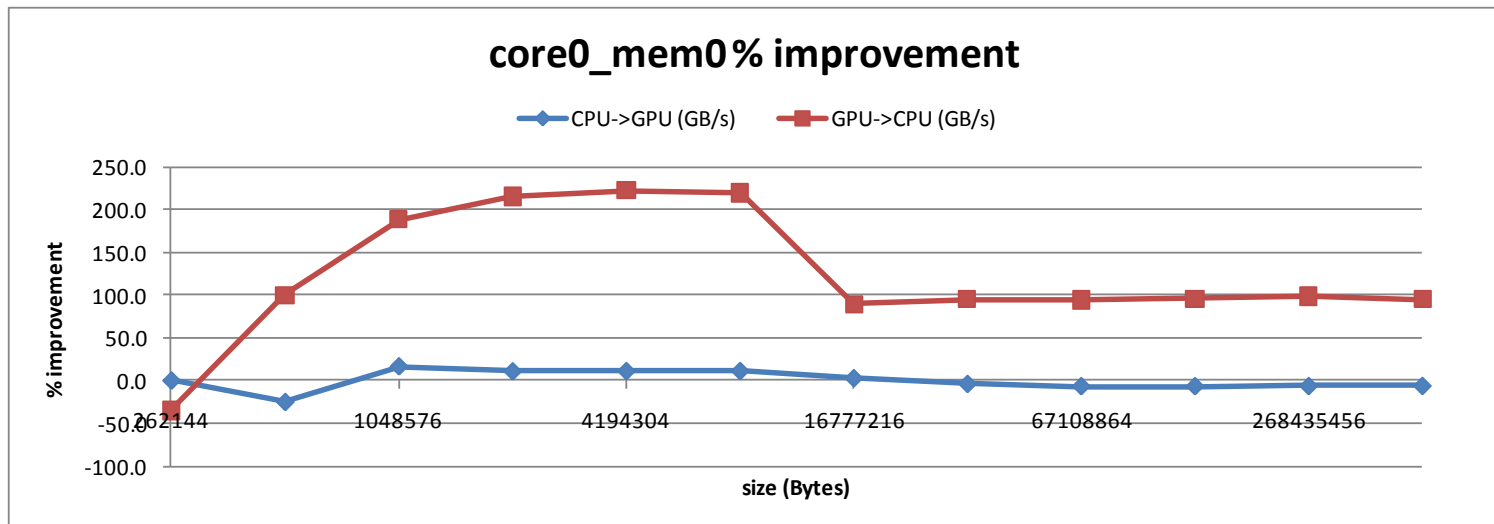
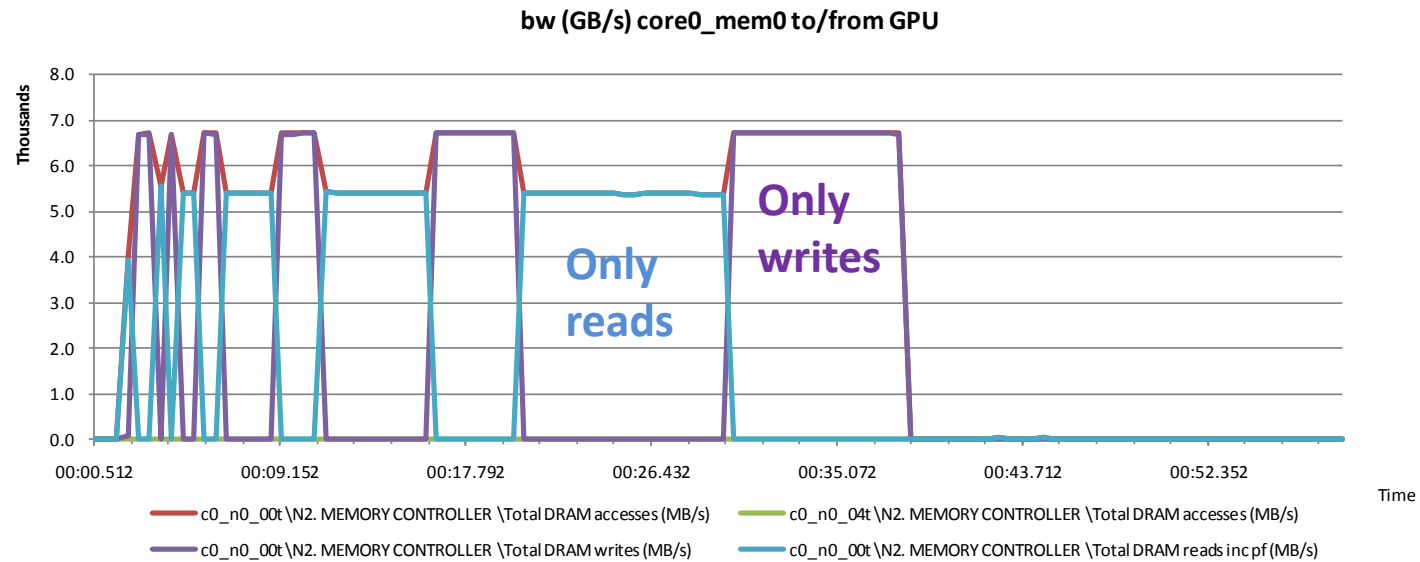
- On Node 1 ↔ GPU 0
- GPU 0 → CPU 1, u-2.1GB/s

- On Node 1 (process) but memory/GART in Node 0
- GPU 0 → CPU 1, u-3.5GB/s

• We cannot pin buffers for all Nodes, to the closest Node to the GPU, since overloads MCT of that Node.



# Linux/Windows driver fix got rid of reads on GPU→CPU

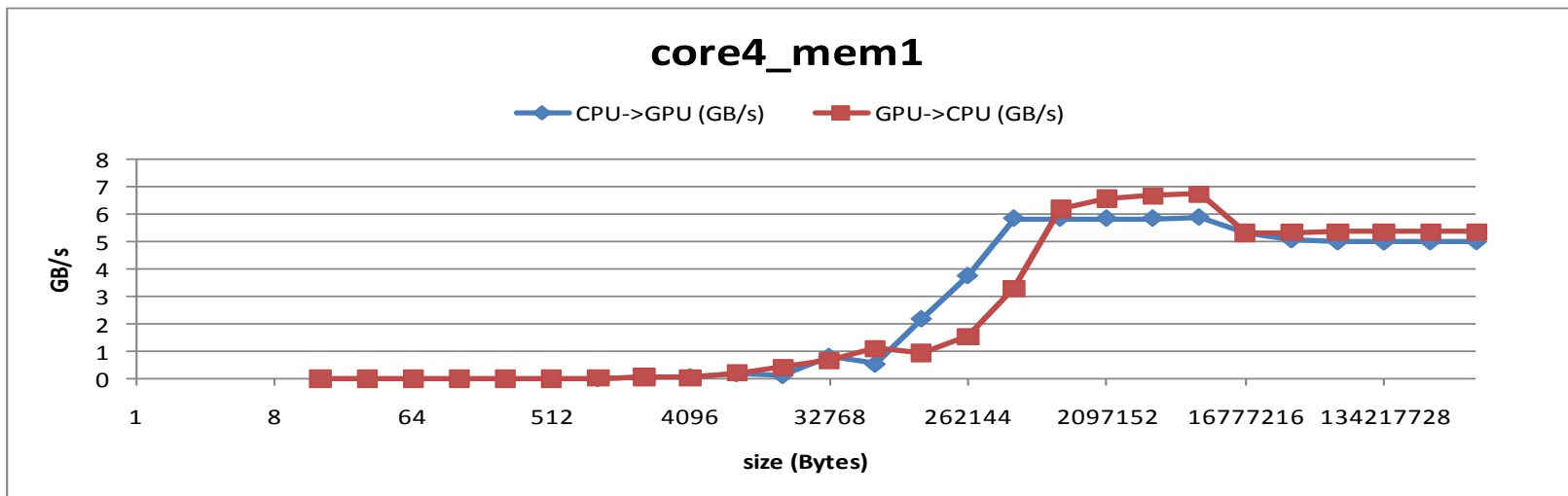
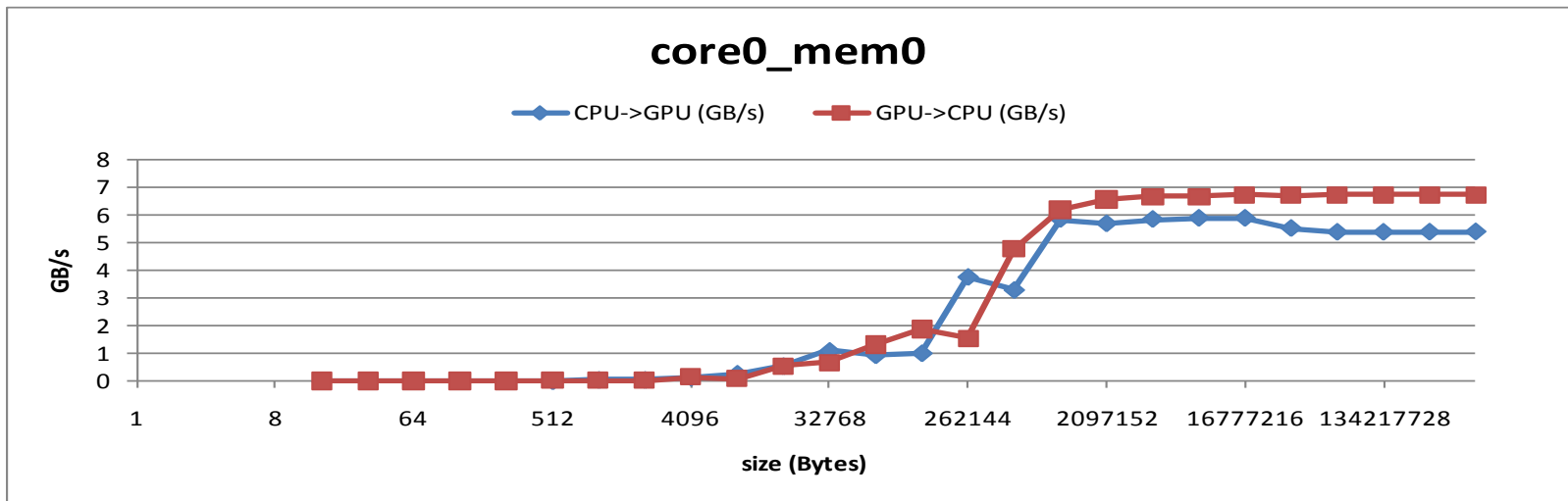


# Linux affinity/binding for CPU and GPU

It is important to set process/thread affinity/binding as well as memory affinity/binding.

- Externally using numactl. Caution, getting obsolete on Multi-Chip-Modules (MCM), since it sees a single chip of all the cores on the socket with single memory controller and single L3 shared cache.
- New tool: HWLOC ([www.open-mpi.org/projects/hwloc](http://www.open-mpi.org/projects/hwloc)), not yet implemented memory binding. Being used in OpenMPI and MVAPICH.
- HWLOC replaces also PLPA (obsolete) since It cannot see properly MCM processors (eg. Magny-Cours, Beckton).
- For CPU and GPU work, set the process/thread affinity and local memory binding to it. Do not rely on first touch for GPU.
- GPU driver can put GART on any node and incur into remote memory access when sending data from/to GPU.
- Enforcing memory binding , will create GART buffers on same memory node, maximizing I/O throughput.

# Linux results with processor and memory binding



# Windows affinity/binding for CPU and GPU

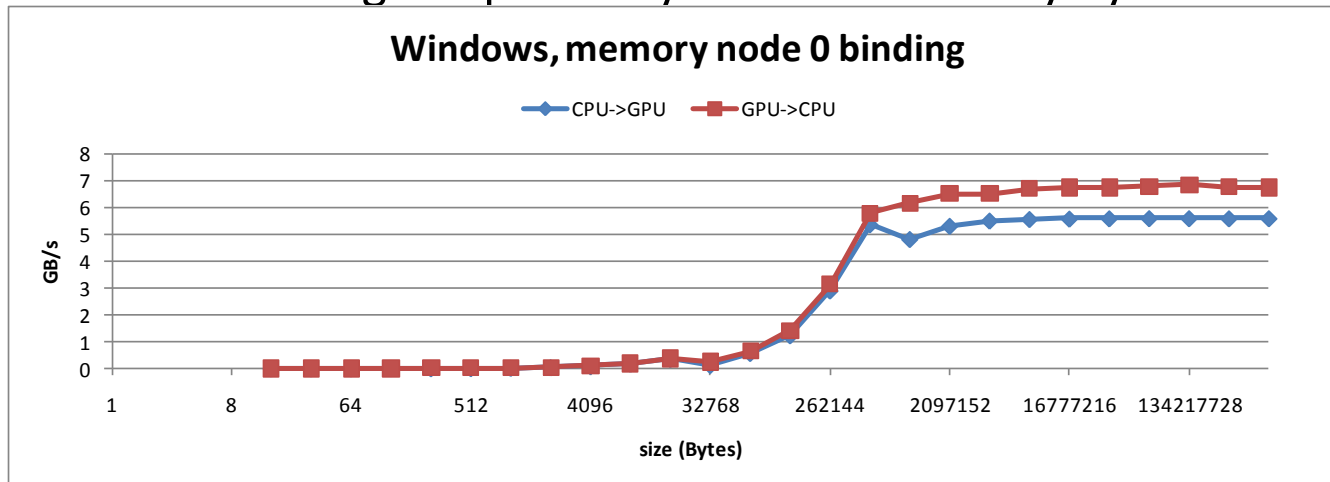
- There is no numactl command on Windows
- There is start /affinity command on DOS prompt that just sets process affinity.
- HWLOC tool also available on Windows, but again memory node binding is mandatory.
- Huge performance penalties if not done memory node.
- Requires usage of API provided by Microsoft for Windows HPC and Windows 7. Main function call: `VirtualAllocNumaEx`
- Debate among SW development teams on where the fix needs to be introduced. Possible SW: Application, OpenCL, CAL, User Driver, Kernel Driver.
- Ideally, OpenCL/CAL should read affinity of process thread and before creating GART resources, set numa node binding.
- Running out of time, quick fix implemented at application level (ie. microbenchmark). Concern about complexity since application developer needs to be aware of cHT topology and NUMA nodes.

# Portion of code changed and results

Example of allocating simple array with memory node binding:

```
a = (double *) VirtualAllocExNuma( hProcess, NULL, array_sz_bytes, MEM_RESERVE | MEM_COMMIT,  
                                PAGE_READWRITE, NodeNumber);  
  
/* a=(double *)malloc(array_sz_bytes); */  
for (i=0; i<array_sz; i++) a[i]= function(i);  
VirtualFreeEx( hProcess, (void *)a, array_sz_bytes, MEM_RELEASE);  
/* free( (void *)a); */
```

For PCIeSpeedTest, fix introduced at USER LEVEL with **VirtualAllocNumaEx + calResCreate2D** instead of **calResAllocRemote2D** which does inside CAL the allocations without having the possibility to set the affinity by the USER.

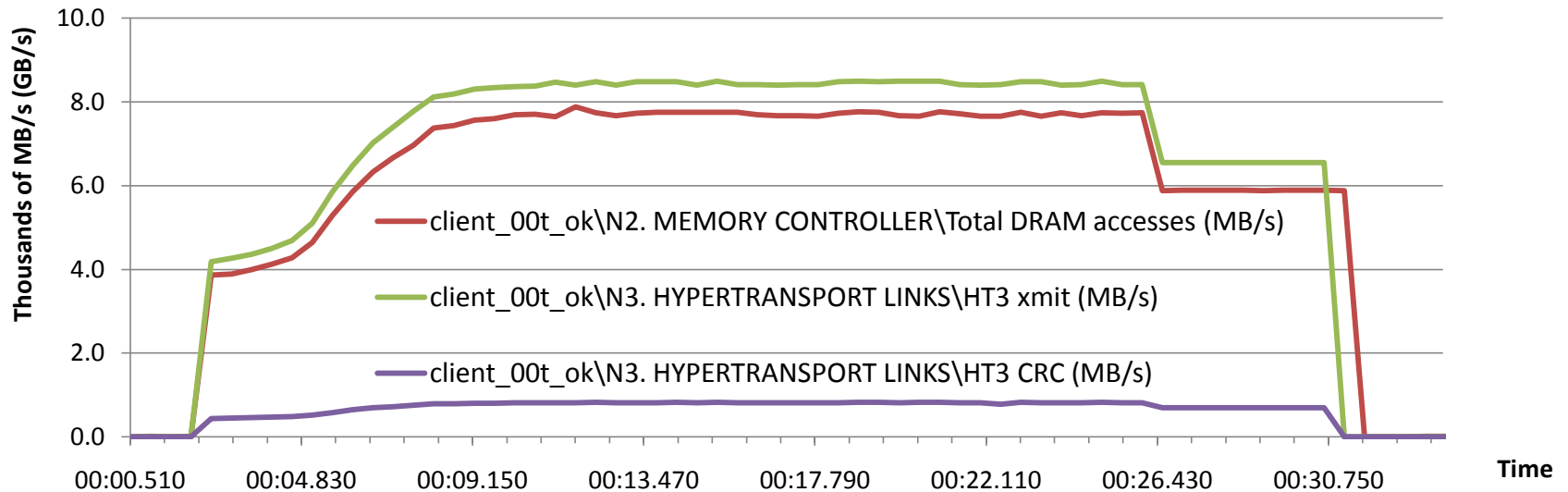


# Q&A

# THANK YOU

How many QDR IB cards can handle a single chipset ? Same performance as Gemini interconnect.

3 QDR IB cards, single PClegn2, RDMA write (GB/s) on client side



# Appendix: Understanding workload requirements

## How to assess Application Efficiency with performance counters

- Efficiency questions (Theoretical vs maximum achievable vs real applications) on IO subsystems: cHT, ncHT, (multi) chipsets, (multi) HCAs, (multi) GPUs, HDs, SSDs.
- Most of those efficiency questions can be represented graphically.

**Example:** roofline model (Williams), which allows to compare architectures and how efficiently the workloads exploit them. Values obtained through performance counters.

2P G34 Magny Cours 2.2 GHz

