Operational Robustness of Accelerator Aware MPI

Sadaf Alam
Swiss National Supercomputing Centre (CSSC)
Switzerland

2nd Annual MVAPICH User Group (MUG) Meeting, 2014
and several T&D systems (incl. an IMB IDataPlex M3 with GPU and two dense GPU servers) plus networking and storage infrastructure …
Users & Operational Responsibilities

• Users priorities:
  - Robust and sustainable performance for production-level simulations
  - Debugging and performance measurement tools to identify and isolate issues (e.g. TAU, Vampir, vendor tools, DDT, Totalview, etc.)

• 24/7 operational support considerations:
  - Monitoring for degradation and failures, isolate components as needed (e.g. Ganglia, customized vendor interfaces)
  - Quick diagnostics and fixes of known problems
  - Alerting mechanisms for on-call services (e.g. Nagios)

# Realities of using bleeding edge technologies
Tools primarily available for non-accelerated clusters running MPI & OpenMP applications plus processors, memories, NICs, ...
Quick Intro to GPU Aware MPI

• **Users point of view**
  
  – MPI_xxxx (data pointers on GPU, ...)
  – No need for explicit copies to/from device

• **For details and up-to-date info please refer to the MUG ’14 tutorial yesterday (25-Aug-2014)**

• **Usage of GPU aware MPI**
  
  – GPU accelerated MD packages & other CUDA applications @ CSCS
  – OpenACC applications, example, S3D

First **CUDA** compiler SDK was released in **2007**
**GPU Aware MVAPICH2** became publically available in **2010**
PGI first **OpenACC** compiler was made available in **2012**
**MPI** standard was first published in **1993**
Evolution of GPU Awareness in MVAPICH2

• MVAPICH2 1.6 (2010-2011)
  – Optimization and enhanced performance for clusters with NVIDIA GPU adapters (with and without GPUDirect technology)

• MVAPICH2 1.8 (2011-2012)
  – Intra-node communication from GPU device buffers using CUDA IPC for better performance and correctness
  – Enabled shared memory communication for host transfers when CUDA is enabled
  – Optimized and tuned collectives for GPU device buffers
  – Enhanced pipelined inter-node device transfers
  – Non-contiguous datatype support in point-to-point and collective communication from GPU buffers
Evolution of GPU Awareness in MVAPICH2

• MVAPICH2 1.9 (2012-2013)
  - Pipelined design for intra-node communication performance
  - Non-blocking CUDA copies for inter-node communication performance
  - Optimal communication channel selection (intra-IOH & inter-IOH)

• MVAPICH2 & MVAPICH2-GDR 2.0 (2013-2014)
  - Dynamic CUDA initialization & support for GPU device selection
  - Multi-rail support for GPU communication
  - Initialize GPU resources only when used by MPI transfer
  - Optimized sub-array data-type processing for GPU-to-GPU communication
  - Non-blocking streams in asynchronous CUDA transfers
  - MPI-3 RMA (one-sided) and atomic operations using GPU buffers with GPUDirect RDMA and pipelining
  - NVIDIA GDR-COPY module & loopback design for inter-node communication
GPU Aware MPI in Action (1 GPU / node)

... Inter-node GPU Direct RDMA...
GPU Aware MPI in Action (2 GPUs / node)

Inter-node GPU Direct RDMA

Inter-IOH shared memory
GPU Aware MPI in Action (8 GPUs / node)

- Inter-node GPU Direct RDMA
- Inter-IOH shared memory
- Intra-IOH IPC
Performance measurements surprises

Latency (usec)

<table>
<thead>
<tr>
<th>Message size (bytes)</th>
<th>Min (1 GPU/node)</th>
<th>Avg. (1 GPU/node)</th>
<th>Min (2 GPU/node)</th>
<th>Avg. (2 GPU/node)</th>
<th>Min (8 GPU/node)</th>
<th>Avg. (8 GPU/node)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4096</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16384</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>65536</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>262144</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1048576</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Identifying affinities (CPU, GPU, memory, NIC, …)

#### GPU Direct Comm Matrix

<table>
<thead>
<tr>
<th></th>
<th>GPU0</th>
<th>GPU1</th>
<th>GPU2</th>
<th>GPU3</th>
<th>GPU4</th>
<th>GPU5</th>
<th>GPU6</th>
<th>GPU7</th>
<th>mlx4_0</th>
<th>CPU Affinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU0</td>
<td>X</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
</tr>
<tr>
<td>0,1,2,3,4,5,6,7,8,9,20,21,22,23,24,25,26,27,28,29</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
</tr>
<tr>
<td>GPU1</td>
<td>PIX</td>
<td>X</td>
<td>PHB</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
</tr>
<tr>
<td>0,1,2,3,4,5,6,7,8,9,20,21,22,23,24,25,26,27,28,29</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
</tr>
<tr>
<td>GPU2</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>PIX</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
</tr>
<tr>
<td>0,1,2,3,4,5,6,7,8,9,20,21,22,23,24,25,26,27,28,29</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
</tr>
<tr>
<td>GPU3</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>X</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
</tr>
<tr>
<td>0,1,2,3,4,5,6,7,8,9,20,21,22,23,24,25,26,27,28,29</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
</tr>
<tr>
<td>GPU4</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>X</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>PHB</td>
</tr>
<tr>
<td>10,11,12,13,14,15,16,17,18,19,30,31</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>X</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>PHB</td>
</tr>
<tr>
<td>GPU5</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>PIX</td>
<td>X</td>
<td>PHB</td>
<td>PHB</td>
<td>PHB</td>
</tr>
<tr>
<td>10,11,12,13,14,15,16,17,18,19,30,31</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>PIX</td>
<td>PHB</td>
</tr>
<tr>
<td>GPU6</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>X</td>
<td>PHB</td>
</tr>
<tr>
<td>10,11,12,13,14,15,16,17,18,19,30,31</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>X</td>
<td>PHB</td>
</tr>
<tr>
<td>GPU7</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>PHB</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>PHB</td>
</tr>
<tr>
<td>10,11,12,13,14,15,16,17,18,19,30,31</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>SOC</td>
<td>PHB</td>
<td>PHB</td>
<td>PHB</td>
<td>PHB</td>
<td>PHB</td>
</tr>
</tbody>
</table>

**Legend:**

- **X** = Self
- **SOC** = Path traverses a socket-level link (e.g. QPI)
- **PHB** = Path traverses a PCIe host bridge
- **PIX** = Path traverses multiple PCIe internal switches
- **CPU Affinity** = The cores that are most ideal for NUMA

---

nvidia-healthmon output for a node with 8 GPU devices and two Intel Xeon sockets
Using binding (CPU, GPU, memory, etc.)

- Use hwloc to find socket, CPU, hw thread & GPU (CUDA core & NVML bindings)
  - hwloc 1.9 (PCI & CUDA enabled version)
  - MPI can use hwloc-topology (extends to GPU?)

- Manual binding options
  - Using numactl
  - Using MPI_LOCAL_RANK info
  - Highly experimental

- Performance evaluation results with bindings
  - Consistent min and max values
  - Variability still high with some improvements in avg. latencies
Using nvidia-healthmon (removing MPI effect)

PCI Bandwidth
Host-to-GPU pinned memory bandwidth: 6235.334473 MB/s
GPU-to-host pinned memory bandwidth: 6399.732422 MB/s
Bidirectional pinned memory bandwidth: 10990.615234 MB/s

Result: SUCCESS

Peer to Peer Bandwidth

Peer-to-peer access between devices 0 and 1 is supported
Bandwidth from device 0 to 1: 6385.51 MB/s
Bandwidth from device 1 to 0: 6364.49 MB/s
Bidirectional bandwidth between device 0 and 1: 10449.60 MB/s

Peer-to-peer access between devices 0 and 2 is supported
Bandwidth from device 0 to 2: 5511.22 MB/s
GPU has a lower peer bandwidth (5511.22 MB/s) than expected (6000.00 MB/s)
Bandwidth from device 2 to 0: 5728.75 MB/s
GPU has a lower peer bandwidth (5728.75 MB/s) than expected (6000.00 MB/s)
Bidirectional bandwidth between device 0 and 2: 7044.35 MB/s

Peer-to-peer access between devices 0 and 3 is supported
Bandwidth from device 0 to 3: 5511.00 MB/s
GPU has a lower peer bandwidth (5511.00 MB/s) than expected (6000.00 MB/s)
Bandwidth from device 3 to 0: 5729.22 MB/s
GPU has a lower peer bandwidth (5729.22 MB/s) than expected (6000.00 MB/s)
Bidirectional bandwidth between device 0 and 3: 7044.67 MB/s

Peer-to-peer access between devices 0 and 4 is not supported.

Peer-to-peer transfer between devices 0 and 4 will use host memory
Bandwidth from device 0 to 4: 5726.79 MB/s
Bandwidth from device 4 to 0: 5746.19 MB/s
Bidirectional bandwidth between device 0 and 4: 10082.57 MB/s

Result: WARNING
Next Steps to minimize variability

• Understand PCIe technologies & impact on on- and off-node communication
  – Extend user and system level tools accordingly

• Understand behavior of driver technologies
  – Extend probes/alerts & tool interfaces

• Better integration into resource managers, e.g. SLURM
  – Portable solutions—no wizardry involved
Final Thoughts

Me wearing a computer scientist hat

Me wearing an operational staff hat

![Graph showing performance over time]

Success

![Graph showing average slowdown over time]

Success
Thank you