MPI + X programming
George Bosilca
CS462 – Fall 2016

https://newton.utk.edu/doc/Documentation/

Rho Cluster with GPGPU
MPI

- Each programming paradigm only covers a particular spectrum of the hardware capabilities
  - MPI is about moving data between distributed memory machines
  - CUDA is about accessing the sheer computations power of a single GPU
  - OpenMP is about taking advantage of the multicores architectures
- What is involved in moving data between 2 machines
  - Bus (PCI/PCI-X)
  - Memory ( pageable, pinned, virtual)
  - OS (security)

Applications need to fully take advantage of all available hardware capabilities. It became imperative to combine different programming paradigms together!
### PCI Express link performance

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>x1</td>
</tr>
<tr>
<td>1.0</td>
<td>8b/10b</td>
<td>2.5 GT/s</td>
<td>250 MB/s</td>
</tr>
<tr>
<td>2.0</td>
<td>8b/10b</td>
<td>5 GT/s</td>
<td>500 MB/s</td>
</tr>
<tr>
<td>3.0</td>
<td>128b/130b</td>
<td>8 GT/s</td>
<td>984.6 MB/s</td>
</tr>
<tr>
<td>4.0 (expected in 2017)</td>
<td>128b/130b</td>
<td>16 GT/s</td>
<td>1.969 GB/s</td>
</tr>
<tr>
<td>5.0 (far future)[28][29]</td>
<td>128b/130b</td>
<td>32 or 25 GT/s[^1]</td>
<td>3.9, or 3.08 GB/s</td>
</tr>
</tbody>
</table>

[^1]: Higher values are possible with the addition of more lane(s) at the cost of extra power.
CUDA

• The CPU is the main driver, it launches kernels on the GPU that perform computations

\[ \text{sum} \ll<1,1\gg>(2, 3, \text{device}_z); \]

– Data must be moved between main memory and GPU prior to the computations

– And must be fetched back once the computation is completed

– In general these are explicit operations (cudaMemcpy)
MPI + CUDA

- MPI is handling main memory while CUDA kernels update the GPU memory. Explicit memory copy from the device to the CPU is necessary to ensure coherence.

```c
if( 0 == rank ) {
    cudaMemcpy(buf_host, buf_dev, size, cudaMemcpyDeviceToHost);
    MPI_Send(buf_host, size, MPI_CHAR, 1, tag, MPI_COMM_WORLD);
} else { // assume MPI rank 1
    MPI_Recv(buf_host, size, MPI_CHAR, 0, tag, MPI_COMM_WORLD, &status);
    cudaMemcpy(buf_dev, buf_host, size, cudaMemcpyHostToDevice);
}
```
Unified Virtual Addressing (UVA)

Devices have similar ranges of memory. Impossible to know where a memory range belongs to.

UVA: One address space for all CPU and GPU memory. No need to alter libraries, they can how identify on which device the memory is located.
Nvidia GPUDirect

- Allowed pinned pages to be shared between different users
  - No need for multiple intermediary buffers to ready the data to be sent over the NiC

CUDA 3.1
Nvidia GPUDirect P2P

- P2P (Peer-to-Peer) allows memory to be copied between devices on the same node without going through the main memory.
Nvidia GPUDirect RDMA

• Push the data out of the GPU directly into the NiC (or other hardware component).
  – Implement standard parts of the PCI-X protocol
if ( 0 == rank ) {
    cudaMemcpy(buf_host, buf_dev, size, cudaMemcpyDeviceToHost);
    MPI_Send(buf_dev, size, MPI_CHAR, 1, tag, MPI_COMM_WORLD);
} else { // assume MPI rank 1
    MPI_Recv(buf_dev, size, MPI_CHAR, 0, tag, MPI_COMM_WORLD, &status);
    cudaMemcpy(buf_dev, buf_host, size, cudaMemcpyHostToDevice);
}
CUDA-aware MPI

```c
if( 0 == rank ) {
    MPI_Send(buf_dev, size, MPI_CHAR, 1, tag, MPI_COMM_WORLD);
} else { // assume MPI rank 1
    MPI_Recv(buf_dev, size, MPI_CHAR, 0, tag, MPI_COMM_WORLD, &status);
}
```

**Graph:**
- **OpenMPI 1.7.4 MLNX FDR IB (4X) Tesla K40**
- **Latency (1 byte):** 19.04 µs, 16.91 µs, 5.52 µs

**Software Versions:**
- **MVAPICH2** 1.8/1.9b
- **OpenMPI** 1.7 (beta)
- **CRAY** MPI (MPT 5.6.2)
- **IBM Platform MPI** (8.3)
- **SGI MPI** (1.08)
\[
U_{i,j}^{n+1} = \frac{1}{4} \left( U_{i-1,j}^n + U_{i+1}^n + U_{i,j-1}^n + U_{i,j+1}^n \right)
\]

Laplace’s equation – MPI + CUDA

for \( j = 1 \) to \( j_{\text{max}} \)
for \( i = 1 \) to \( i_{\text{max}} \)
\[
U_{\text{new}}(i,j) = 0.25 \times ( U(i-1,j) + U(i+1,j) + U(i,j-1) + U(i,j+1) )
\]
end for
end for