Overview of HPC Technologies
Part-II

Dhabaleswar K. (DK) Panda
The Ohio State University
E-mail: panda@cse.ohio-state.edu
http://www.cse.ohio-state.edu/~panda
HPC Technologies

• Hardware
  – Interconnects – InfiniBand, RoCE, Omni-Path, etc.
  – Processors – GPUs, Multi-/Many-core CPUs, Tensor Processing Unit (TPU), FPGAs, etc.
  – Storage – NVMe, SSDs, Burst Buffers, etc.

• Communication Middleware
  – Message Passing Interface (MPI)
    • CUDA-Aware MPI, Many-core Optimized MPI runtimes (KNL-specific optimizations)
  – NVIDIA NCCL
Technology Constraints: Power Consumption

Transistors (Thousands)

SPECint Performance

Frequency (MHz)

Typical Power (W)

Number of Cores

Intel 48-Core Prototype
AMD 4-Core Opteron
DEC Alpha 21264
MIPS R2K

~15%/year
~9%/year


Courtesy
Christopher Barren
Intel Tick/Tock Product Releases

- **Nehalem**: 45nm, New Micro-architecture
  - **Tock**
  - 2009

- **Westmere**: 32nm, New Process Technology
  - **Tick**
  - 2010

- **Sandy Bridge**: 32nm, New Micro-architecture
  - **Tock**
  - 2011

- **Ivy Bridge**: 22nm, New Process Technology
  - **Tick**
  - 2012

- **Haswell**: 22nm, New Micro-architecture
  - **Tock**
  - 2013

- **Broadwell**: 14nm, New Process Technology
  - **Tick**
  - 2014

- **Skylake**: 14nm, New Micro-architecture
  - **Tock**
  - 2015

- **Kaby Lake**: 14nm, New Micro-architecture
  - **Tock**
  - 2016

- **Optimization**

**Courtesy**
Christopher Barren
Single-source approach to Multi- and Many-Core

Eliminates Need to Fork Application Code
Spectrum of Programming Models and Mindsets

Multi-Core Centric
- Xeon
- Multi-Core Hosted
  - General purpose serial and parallel computing
- Symmetric
  - Codes with balanced needs
- Offload
  - Codes with highly-parallel phases
- Multi-Core Hosted
  - Xeon
  - Many-Core Hosted
  - MIC

Range of models to meet application needs
• ~1.7B transistors in ~122 mm² in a 22nm tri-gate process
• Four out-of-order cores each with two SMT threads running at 4.0-4.2 GHz
• Three-level cache hierarchy with last-level on-chip cache capacity of 8MB
• Max thermal design power of 91W
• 2 DDR4 DRAM memory controllers, 34.1 GB/s max memory bandwidth
• Integrated 3D graphics processor running at 350 MHz to 1.15 GHz
• Pipelined bus on-chip network connecting cores, last-level cache banks, and GPU
# Intel® AVX Technology

## AVX
- 256-bit basic FP 16 registers
- NDS (and AVX128)
- Improved blend
- MASKMOV
- Implicit unaligned

## AVX2
- Float16 (IVB 2012) 256-bit FP FMA
- 256-bit integer
- PERMD
- Gather

## AVX-512
- 512-bit FP/Integer
- 32 registers
- 8 mask registers
- Embedded rounding
- Embedded broadcast
- Scalar/SSE/AVX “promotions”
- HPC additions
- Gather/Scatter

---

*New!*
AVX512: 512-bit SIMD Extensions
Intel® Turbo Boost Technology 2.0

Efficient.
✓ Adapts by varying turbo frequency to conserve energy depending upon the type of instructions

Dynamic.
✓ Boosts power level to achieve performance gains for high intensity "dynamic" workloads

Intelligent.
✓ Power averaging algorithm manages power and thermal headroom to optimize performance

Intel® Turbo Boost Technology 2.0 delivers intelligent and energy efficient performance on demand
GPU Technology

**GPU: Graphics Processing Unit**

- Hundreds of Cores
- Programmable
- Can be easily installed in most desktops
- Similar price to CPU
- GPU follows Moore's Law better than CPU
Introduction

Motivation:

- GT200 = GeForce GTX 280
- G92 = GeForce 9800 GTX
- G80 = GeForce 8800 GTX
- NV35 = GeForce FX 5950 Ultra
- NV30 = GeForce FX 5900
- NV40 = GeForce 6800 Ultra
- G71 = GeForce 7900 GTX
- G70 = GeForce 7800 GTX
- NV30 = GeForce FX 5900
- NV40 = GeForce 6800 Ultra
- G80 Ultra
- G92
GPU Hardware

Multiprocessor Structure:

CPU

GPU

Control

ALU

ALU

ALU

ALU

Cache

DRAM

DRAM
GPU Hardware

Multiprocessor Structure:

- N multiprocessors with M cores each
- SIMD – Cores share an Instruction Unit with other cores in a multiprocessor.
- Diverging threads may not execute in parallel.
GPU Hardware

Memory Hierarchy:

- Processors have 32-bit registers
- Multiprocessors have shared memory, constant cache, and texture cache
- Constant/texture cache are read-only and have faster access than shared memory.
GPU Hardware

NVIDIA GTX280 Specifications:

- 933 GFLOPS peak performance
- 10 thread processing clusters (TPC)
- 3 multiprocessors per TPC
- 8 cores per multiprocessor
- 16384 registers per multiprocessor
- 16 KB shared memory per multiprocessor
- 64 KB constant cache per multiprocessor
- 6 KB < texture cache < 8 KB per multiprocessor
- 1.3 GHz clock rate
- Single and double-precision floating-point calculation
- 1 GB DDR3 dedicated memory
GPU Hardware

GeForce GTX 280 Parallel Computing Architecture

- Thread Scheduler
- Thread Processing Clusters
- Atomic/Tex L2
- Memory
GPU Hardware

**Thread Scheduler:**
- Hardware-based
- Manages scheduling threads across thread processing clusters
- Nearly 100% utilization: If a thread is waiting for memory access, the scheduler can perform a zero-cost, immediate context switch to another thread
- Up to 30,720 threads on the chip
GPU Hardware

Thread Processing Cluster:

IU - instruction unit    TF - texture filtering
Programming Model

**Past:**
- The GPU was intended for graphics only, not general purpose computing.
- The programmer needed to rewrite the program in a graphics language, such as OpenGL
- Complicated

**Present:**
- NVIDIA developed CUDA, a language for general purpose GPU computing
- Simple
Programming Model

**CUDA:**
- Compute Unified Device Architecture
- Extension of the C language
- Used to control the device
- The programmer specifies CPU and GPU functions
  - The host code can be C++
  - Device code may only be C
- The programmer specifies thread layout
Thread Layout:

- Threads are organized into blocks.
- Blocks are organized into a grid.
- A multiprocessor executes one block at a time.
- A warp is the set of threads executed in parallel
- 32 threads in a warp
Programming Model

- Heterogeneous Computing:
  - GPU and CPU execute different types of code.
  - CPU runs the main program, sending tasks to the GPU in the form of kernel functions
  - Multiple kernel functions may be declared and called.
  - Only one kernel may be called at a time.
Programming Model: GPU vs. CPU Code

Supercomputing Products

**Tesla C1070:**
- Server Blade
- 4.14 TFLOPS peak performance
- Contains 4 Tesla GPUs
- 960 Cores
- 16GB DDR3
- 408 GB/s bandwidth
- 800W max power consumption
Trends in GPU Technology

- NVIDIA Volta is optimized for Deep Learning workloads
  - has dedicated “Tensor Cores” (FP16 or half precision) for both Training and Inference
  - 2.4X faster than Pascal GPUs for ResNet-50 training

Courtesy: https://devblogs.nvidia.com/parallelforall/inside-volta/
<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transistors</td>
<td>21B</td>
</tr>
<tr>
<td>Die area</td>
<td>815 mm²</td>
</tr>
<tr>
<td>SMs</td>
<td>80</td>
</tr>
<tr>
<td>CUDA Cores</td>
<td>5120</td>
</tr>
<tr>
<td>Tensor Cores</td>
<td>640</td>
</tr>
<tr>
<td>HBM2 Memory</td>
<td>16 GB</td>
</tr>
<tr>
<td>Memory bandwidth</td>
<td>900 GB/s</td>
</tr>
<tr>
<td>NVLink bandwidth</td>
<td>300 GB/s</td>
</tr>
</tbody>
</table>

*full GV100 chip contains 84 SMs
## GPU PERFORMANCE COMPARISON

<table>
<thead>
<tr>
<th>Feature</th>
<th>P100</th>
<th>V100</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training acceleration</strong></td>
<td>10 TOPS</td>
<td>120 TOPS</td>
<td></td>
</tr>
<tr>
<td><strong>Inference acceleration</strong></td>
<td>21 TFLOPS</td>
<td>120 TOPS</td>
<td></td>
</tr>
<tr>
<td><strong>FP64/FP32</strong></td>
<td>5/10 TFLOPS</td>
<td>7.5/15 TFLOPS</td>
<td></td>
</tr>
<tr>
<td><strong>HBM2 Bandwidth</strong></td>
<td>720 GB/s</td>
<td>900 GB/s</td>
<td></td>
</tr>
<tr>
<td><strong>NVLink Bandwidth</strong></td>
<td>160 GB/s</td>
<td>300 GB/s</td>
<td></td>
</tr>
<tr>
<td><strong>L2 Cache</strong></td>
<td>4 MB</td>
<td>6 MB</td>
<td></td>
</tr>
<tr>
<td><strong>L1 Caches</strong></td>
<td>1.3 MB</td>
<td>10 MB</td>
<td></td>
</tr>
</tbody>
</table>
NEW HBM2 MEMORY ARCHITECTURE

V100 measured on pre-production hardware.
VOLTA NVLINK

300GB/sec
50% more links
28% faster signaling
POWER8: Processor Performance Leadership & Accelerator Interfaces

- **Faster Cores**
  8 Threads Per Core

- **Larger Caches**
  Direct Accelerator Interconnect

- **3x Higher Memory Bandwidth, 1 TB Memory per Socket**

**POWER8**
12 Cores, 96 Threads
4 Level Large Caches
Up to 1 TB per socket
Up to 230 GB/s sustained
Accelerates Technology roadmap

**Mellanox Interconnect**
- **Connect-IB**
  - FDR Infiniband
  - PCIe Gen3
- **ConnectX-4**
  - EDR Infiniband
  - CAPI over PCIe Gen3
- **ConnectX-5**
  - Next-Gen Infiniband
  - Enhanced CAPI over PCIe Gen4

**NVIDIA GPUs**
- **Kepler**
  - PCIe Gen3
- **Pascal**
  - NVLink
- **Volta**
  - Enhanced NVLink

**IBM CPUs**
- **POWER8**
  - OpenPower CAPI Interface
- **POWER8 with NVLink**
- **POWER9**
  - Enhanced CAPI & NVLink

**IBM Systems**
- **2015**
- **2016**
- **2017**
Application Benchmarks

POWER8: Up to 2x Faster on Applications over Intel Haswell x86 CPUs
An introduction to ARM

ARM is the world's leading semiconductor intellectual property supplier.
We license to over 440 partners, are present in 95% of smart phones, 80% of digital cameras, 35% of all electronic devices, and a total of 72 billion ARM cores have been shipped since 1990.

Our CPU business model:
- License technology to partners, who use it to create their own system-on-chip (SoC) products.
- We may license an ISA (e.g. “ARMv8-A”) or a specific microarchitectural implementation (e.g. “Cortex-A72”)

…and our IP extends beyond the CPU
Why ARM in HPC?

Energy efficiency

- Energy has always been a first-class design constraint at ARM, the goal in our uArchitecture IP is to maintain that power efficiency advantage as we increase performance. The CPU core is only one element of the equation, we are also working to explore efficiency in the memory and on-chip interconnect space.

Choice

- Independent silicon providers within a shared software ecosystem.
- Wide range of price/performance/power/size options available.

Customisation

- ARM’s partners can build SoCs very quickly.
- Getting interesting in the US:
  - SoC for HPC: [http://www.socforhpc.org](http://www.socforhpc.org)
Expanding ARMv8 vector processing

- ARMv7 Advanced SIMD (aka ARM NEON instructions) now 12 years old
  - Integer, fixed-point and non-IEEE single-precision float, on well-conditioned data
  - 16×128-bit vector registers

- AArch64 Advanced SIMD was an evolution
  - Gained full IEEE double-precision float and 64-bit integer vector ops
  - Vector register file grew from 16×128b to 32×128b

- New markets for ARMv8-A are demanding more radical changes
  - Gather load & Scatter store
  - Per-lane predication
  - Longer vectors

- But what is the preferred vector length?
Introducing the Scalable Vector Extension (SVE)

- There is **no** preferred vector length
  - Vector Length (VL) is hardware choice, from 128 to 2048 bits, in increments of 128
  - Vector Length Agnostic (VLA) programming adjusts dynamically to the available VL
  - No need to recompile, or to rewrite hand-coded SVE assembler or C intrinsics

- SVE is **not** an extension of Advanced SIMD
  - A separate architectural extension with a new set of A64 instruction encodings
  - Focus is HPC scientific workloads, not media/image processing

- Amdahl says you need high vector utilisation to achieve significant speedups
  - Compilers often unable to vectorize due to intra-vector data & control dependencies
  - SVE also begins to address some of the traditional barriers to auto-vectorization
Enable the wide variety of ARM cores available today without adding complexity to the software ecosystem.

- Commercially supported 64-bit ARMv8 vendor math libraries for scientific computing.
- Built and validated using technology from the Numerical Algorithms Group (NAG).
- ARM silicon partners provide us with tuned kernels.

**Capabilities:**
- BLAS
- LAPACK
- FFT

**Tuned for:**
- Cortex-A57,
- Applied Micro X-Gene®
- Cavium® ThunderX
Google TPU

- CISC style instruction set
- Uses systolic arrays as the heart of multiply unit

  
Intel Nervana also has a TPU

nervana tensor processing unit

- Unprecedented compute density
- Scalable distributed architecture
- Memory near computation
- Learning and inference
- Exploit limited precision
- Incorporate latest advances
- Power efficiency

Model and substrate for computation

Do this instead:

Feasible, but still hard.

Custom ASIC

- Model description language
- Hardware abstraction layer
- Distributed primitives
- Compilers, drivers

Deep learning model

Intel Neural Network Processor (NNP)

- Intel® Nervana™ Neural Network Processors (NNP)
  - formerly known as “Lake Crest”
- Recently announced as part of Intel’s strategy for next-gen. AI systems
- Purpose built architecture for deep learning
- 1 TB/s High Bandwidth Memory (HBM)
- Spatial Architecture
- FlexPoint format
  - Similar performance (in terms of accuracy) to FP32 while using 16 bits of storage

GraphCore – Intelligence Processing Unit (IPU)

- New processor that’s the first to be specifically designed for machine intelligence workloads – an Intelligence Processing Unit (IPU)
  - Massively parallel
  - Low-precision floating-point compute
  - Higher compute density
- UK-based Startup
- Early benchmarks show 10-100x speedup over GPUs
  - Presented at NIPS 2017

Courtesy: https://www.graphcore.ai/posts/preliminary-ipu-benchmarks-providing-previously-unseen-performance-for-a-range-of-machine-learning-applications
Poplar Graph Programming Framework

- Poplar -- graph programming framework for IPU accelerated platforms
- C++ framework that provides a seamless interface DL frameworks like Tensorflow and MXNet
- Existing applications written for Tensorflow will work out of the box on an IPU.
- Set of drivers, application libraries and debugging and analysis tools

[https://www.graphcore.ai/hubfs/assets/Poplar%20technical%20overview%20NEW%20BRAND.pdf](https://www.graphcore.ai/hubfs/assets/Poplar%20technical%20overview%20NEW%20BRAND.pdf)
HPC Technologies

• Hardware
  – Interconnects – InfiniBand, RoCE, Omni-Path, etc.
  – Processors – GPUs, Multi-/Many-core CPUs, Tensor Processing Unit (TPU), FPGAs, etc.
  – Storage – NVMe, SSDs, Burst Buffers, etc.

• Communication Middleware
  – Message Passing Interface (MPI)
    • CUDA-Aware MPI, Many-core Optimized MPI runtimes (KNL-specific optimizations)
  – NVIDIA NCCL
Trends in High-Performance Storage

<table>
<thead>
<tr>
<th>NVMe</th>
<th>NVRAM</th>
<th>3D XPoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>High Speed interface for SSDs in a PCIe form factor used as block storage</td>
<td>Non-volatile DRAM backed up by battery or super capacitor used as byte addressable memory</td>
</tr>
<tr>
<td>Form Factor</td>
<td>Connects to PCIe bus</td>
<td>Connects to a DDR3 DIMM slot</td>
</tr>
<tr>
<td>Max Capacity</td>
<td>2 TB</td>
<td>16GB</td>
</tr>
<tr>
<td>Read IOPS (Random)</td>
<td>750,000</td>
<td>1.4 Million</td>
</tr>
<tr>
<td>Write IOPS (Random)</td>
<td>430,000</td>
<td>1.4 Million</td>
</tr>
<tr>
<td>Latency</td>
<td>15 Microsecond</td>
<td>10 Nanoseconds</td>
</tr>
<tr>
<td>Ideal Use Cases</td>
<td>Caching Tier: Transactional workloads requiring high IOPS</td>
<td>Byte Addressable memory for metadata &amp; client side caching, reduce write amplification</td>
</tr>
<tr>
<td>Price ($/Gig)</td>
<td>$</td>
<td>$$$</td>
</tr>
</tbody>
</table>


What is NVM Express™?

- Industry standard for PCIe SSDs
  - High-performance, low-latency, PCIe SSD interface
    - Command set + PCIe register interface
  - In-box NVMe host drivers for Linux, Windows, VmWare, ...
  - Standard h/w drive form factors, mobile to enterprise

- NVMe community is 80+ companies strong and growing
  - Learn more at nvmexpress.org
Non-Volatile Memory Express (NVMe) began as an industry standard solution for efficient PCIe attached non-volatile memory storage (e.g., NVMe PCIe SSDs).

- Low latency and high IOPS direct-attached NVM storage
- Multiple companies shipping and deploying NVMe PCIe SSDs today
Expanding NVMe to Fabrics

- Built on common NVMe architecture with additional definitions to support multi-based NVMe operations
- Standardization of NVMe over a range Fabric types
  - Initial fabrics; RDMA(RoCE, iWARP, InfiniBand™) and Fibre Channel First release candidate specification in early 2016
  - NVMe.org Fabrics Linux Driver WG developing host and target drivers
Why NVMe Over Fabrics

- **End-to-End NVMe semantics across a range of topologies** Retains NVMe efficiency and performance over network fabrics Eliminates unnecessary protocol translations Enables low-latency and high IOPS remote NVMe storage solutions
HPC Technologies

• Hardware
  – Interconnects – InfiniBand, RoCE, Omni-Path, etc.
  – Processors – GPUs, Multi-/Many-core CPUs, Tensor Processing Unit (TPU), FPGAs, etc.
  – Storage – NVMe, SSDs, Burst Buffers, etc.

• Communication Middleware
  – Message Passing Interface (MPI)
    • CUDA-Aware MPI, Many-core Optimized MPI runtimes (KNL-specific optimizations)
  – NVIDIA NCCL
Parallel Programming Models Overview

• Programming models provide abstract machine models
• Models can be mapped on different types of systems
  – e.g. Distributed Shared Memory (DSM), MPI within a node, etc.
• PGAS models and Hybrid MPI+PGAS models are gradually receiving importance
MPI Features and Implementations

• Major MPI features
  – Point-to-point Two-sided Communication
  – Collective Communication
  – One-sided Communication

• Message Passing Interface (MPI)
  – MVAPICH2
  – OpenMPI, IntelMPI, CrayMPI, IBM Spectrum MPI
  – And many more...
Broadcast Collective Communication Pattern

- Broadcast a message from process with rank of "root" to all other processes in the communicator

```c
int MPI_Bcast( void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm );
```

<table>
<thead>
<tr>
<th>Input-only Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>Number of entries in buffer</td>
</tr>
<tr>
<td>datatype</td>
<td>Data type of buffer</td>
</tr>
<tr>
<td>root</td>
<td>Rank of broadcast root</td>
</tr>
<tr>
<td>comm</td>
<td>Communicator handle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input/Output Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer</td>
<td>Starting address of buffer</td>
</tr>
</tbody>
</table>
Allreduce Collective Communication Pattern

- Element-wise Sum data from all processes and sends to all processes

```c
int MPI_Allreduce (const void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype,
                  MPI_Op operation, MPI_Comm comm)
```

**Input-only Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sendbuf</td>
<td>Starting address of send buffer</td>
</tr>
<tr>
<td>recvbuf</td>
<td>Starting address of recv buffer</td>
</tr>
<tr>
<td>type</td>
<td>Data type of buffer elements</td>
</tr>
<tr>
<td>count</td>
<td>Number of elements in the buffers</td>
</tr>
<tr>
<td>operation</td>
<td>Reduction operation to be performed (e.g. sum)</td>
</tr>
<tr>
<td>comm</td>
<td>Communicator handle</td>
</tr>
</tbody>
</table>

**Input/Output Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>recvbuf</td>
<td>Starting address of receive buffer</td>
</tr>
</tbody>
</table>

**Sendbuf (Before)**

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

**Recvbuf (After)**

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>16</td>
</tr>
</tbody>
</table>
Overview of the MVAPICH2 Project

• High Performance open-source MPI Library for InfiniBand, Omni-Path, Ethernet/iWARP, and RDMA over Converged Ethernet (RoCE)
  – MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.1), Started in 2001, First version available in 2002
  – MVAPICH2-X (MPI + PGAS), Available since 2011
  – Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  – Support for Virtualization (MVAPICH2-Virt), Available since 2015
  – Support for Energy-Awareness (MVAPICH2-EA), Available since 2015
  – Support for InfiniBand Network Analysis and Monitoring (OSU INAM) since 2015

– Used by more than 3,025 organizations in 89 countries
– More than 572,000 (> 0.5 million) downloads from the OSU site directly
– Empowering many TOP500 clusters (Nov ‘18 ranking)
  • 3rd, 10,649,600-core (Sunway TaihuLight) at National Supercomputing Center in Wuxi, China
  • 5th, 448,448 cores (Frontera) at TACC
  • 8th, 391,680 cores (ABCI) in Japan
  • 15th, 570,020 cores (Neurion) in South Korea and many others
– Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, and OpenHPC)

– [http://mvapich.cse.ohio-state.edu](http://mvapich.cse.ohio-state.edu) Partner in the TACC Frontera System

• Empowering Top500 systems for over a decade
**MPI + CUDA - Naive**

- Data movement in applications with standard MPI and CUDA interfaces

**At Sender:**
```c
cudaMemcpy(s_hostbuf, s_devbuf, ...);
MPI_Send(s_hostbuf, size, ...);
```

**At Receiver:**
```c
MPI_Recv(r_hostbuf, size, ...);
cudaMemcpy(r_devbuf, r_hostbuf, ...);
```

*High Productivity and Low Performance*
At Sender:
for (j = 0; j < pipeline_len; j++)
cudaMemcpyAsync(s_hostbuf + j * blk, s_devbuf + j * blkSz, ...);
for (j = 0; j < pipeline_len; j++) {
    while (result != cudaSuccess) {
        result = cudaStreamQuery(...);
        if(j > 0) MPI_Test(...);
    }
    MPI_Isend(s_hostbuf + j * block_sz, blkSz, ...);
}
MPI_Waitall();

<<Similar at receiver>>

Low Productivity and High Performance
GPU-Aware (CUDA-Aware) MPI Library: MVAPICH2-GDR

- Standard MPI interfaces used for unified data movement
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

```
MPI_Send(s_devbuf, size, ...);
```

At Receiver:

```
MPI_Recv(r_devbuf, size, ...);
```

High Performance and High Productivity
Optimized MVAPICHER2-GDR Design

**GPU-GPU Inter-node Latency**

- MV2-(NO-GDR)
- MV2-GDR 2.3rc1

**GPU-GPU Inter-node Bandwidth**

- MV2-(NO-GDR)
- MV2-GDR-2.3rc1

MVAPICHER2-GDR-2.3

Intel Haswell (E5-2687W @ 3.10 GHz) node - 20 cores

NVIDIA Volta V100 GPU

Mellanox Connect-X4 EDR HCA

CUDA 9.0

Mellanox OFED 4.0 with GPU-Direct-RDMA
NCCL Communication Library

• Collective Communication with a caveat!
  – GPU buffer exchange
  – **Dense Multi-GPU** systems  
    (Cray CS-Storm, DGX-1)
  – MPI-like – but not MPI standard compliant

• NCCL (pronounced Nickel)
  – Open-source Communication Library by NVIDIA
  – Topology-aware, ring-based (linear) collective communication library for GPUs
  – Divide bigger buffers to smaller chunks
  – Good performance for large messages
    • Kernel-based threaded copy (Warp-level Parallel) instead of cudaMemcpy

NVIDIA NCCL

• NCCL is a collective communication library
  – NCCL 1.x is only for Intra-node communication on a single-node
• NCCL 2.0 supports inter-node communication as well
• Design Philosophy
  – Use Rings and CUDA Kernels to perform efficient communication
• NCCL is optimized for dense multi-GPU systems like the DGX-1 and DGX-1V

Courtesy: https://www.nextplatform.com/2016/05/04/nvlink-takes-gpu-acceleration-next-level/

Fully connected quad
120 GB/s per GPU bidirectional for peer traffic
40 GB/s per GPU bidirectional to CPU
Direct Load/store access to CPU Memory
High Speed Copy Engines for bulk data movement
Performance of NCCL 1.x Collectives

[Graphs showing performance of NCCL 1.x collectives: Broadcast, All-Reduce, All-Gather, Reduce-Scatter.]

NCCL 2: Multi-node GPU Collectives