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

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

Transistors (Thousands)
- SPECint Performance
- Frequency (MHz)
- Typical Power (W)
- Number of Cores

~15%/year
~9%/year


Courtesy
Christopher Barren
~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

<table>
<thead>
<tr>
<th>AVX</th>
<th>AVX2</th>
</tr>
</thead>
<tbody>
<tr>
<td>256-bit basic FP 16 registers</td>
<td>Float16 (IVB 2012) 256-bit FP FMA</td>
</tr>
<tr>
<td>NDS (and AVX128)</td>
<td>256-bit integer</td>
</tr>
<tr>
<td>Improved blend</td>
<td>PERMD</td>
</tr>
<tr>
<td>MASKMOV</td>
<td>Gather</td>
</tr>
<tr>
<td>Implicit unaligned</td>
<td></td>
</tr>
</tbody>
</table>

### SNB

- HSW

### KNL

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

**New!**

Copyright © 2015, Intel Corporation. All rights reserved Avinash Sodani ISC 2015 Intel® Xeon Phi ™ Workshop.
AVX512: 512-bit SIMD Extensions
### AMD EPYC CPU architecture

**AMD EPYC series**
- EPYC v1 32-core (Naples)
- EPYC v2 64-core (Rome)

**High performance cores**
- High core count
- Larger cache sizes

**Infinity Fabric memory subsystem**
- 8 DD4 channels per socket
- Up to 21.3 GB/s

**Configurable NUMA domain**
- Up to 16 NUMA per socket

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Naples (EPYC 7001 series)</th>
<th>Rome (EPYC 7002 series)</th>
<th>Xeon Cascade Lake - AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Zen (1)</td>
<td>Zen 2</td>
<td>Cascade Lake - AP</td>
</tr>
<tr>
<td>Cores</td>
<td>8 to 32</td>
<td>8 to 64</td>
<td>32 to 56</td>
</tr>
<tr>
<td>PCI Lanes</td>
<td>128</td>
<td>128</td>
<td>64 (40 avail.)</td>
</tr>
<tr>
<td>PCI Generation</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Process Technology</td>
<td>14nm</td>
<td>7nm (CPU and Cache)</td>
<td>14nm</td>
</tr>
<tr>
<td>Memory Channels</td>
<td>8</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>Memory Speed</td>
<td>2666 MHz</td>
<td>3200MHz</td>
<td>2933MHz</td>
</tr>
</tbody>
</table>

IBM OpenPOWER Power9 Architecture

- Up to 22 cores per socket
- Very high degree of multi-threading
  - Up to 8 threads per core (SMT8)
- Very high core clock
  - Up to 3.8 GHz
- Up to 4 levels of cache
- Summit supercomputer
  - IBM Power9 CPU
  - NVIDIA Volta GPUs
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.

Customisation

- Wide range of price/performance/power/sizes available.
- ARM’s partners can build SoCs very quickly.
- Getting interesting in the US:
  - SoC for HPC: 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
ARM Performance Libraries

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
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:

![Graph showing the increase in peak GFLOPS from NV30 to GT200 over time, with key dates and corresponding GPUs.]
GPU Hardware

Multiprocessor Structure:

- Control
- ALU
- ALU
- ALU
- Cache
- DRAM

CPU

GPU
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
Programming Model

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/
TESLA V100

21B transistors
815 mm²

80 SM
5120 CUDA Cores
640 Tensor Cores

16 GB HBM2
900 GB/s HBM2
300 GB/s NVLink

*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>Training acceleration</td>
<td>10 TOPS</td>
<td>120 TOPS</td>
<td></td>
</tr>
<tr>
<td>Inference acceleration</td>
<td>21 TFLOPS</td>
<td>120 TOPS</td>
<td></td>
</tr>
<tr>
<td>FP64/FP32</td>
<td>5/10 TFLOPS</td>
<td>7.5/15 TFLOPS</td>
<td></td>
</tr>
<tr>
<td>HBM2 Bandwidth</td>
<td>720 GB/s</td>
<td>900 GB/s</td>
<td></td>
</tr>
<tr>
<td>NVLink Bandwidth</td>
<td>160 GB/s</td>
<td>300 GB/s</td>
<td></td>
</tr>
<tr>
<td>L2 Cache</td>
<td>4 MB</td>
<td>6 MB</td>
<td></td>
</tr>
<tr>
<td>L1 Caches</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
AMD Radeon Instinct GPU

AMD Vega20 series
  MI25, MI50, MI60
  Upcoming MI100
NVIDIA V100 about 6% better than MI60 for DL workloads
MI60 about 11% better for DGEMM¹
Higher performance-to-price ratio
Upcoming MI100 and beyond to be installed in Exascale systems
  Frontier
  El Capitan

<table>
<thead>
<tr>
<th>Configuration</th>
<th>MI60</th>
<th>MI50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute Units</td>
<td>64</td>
<td>60</td>
</tr>
<tr>
<td>Peak FP32 TFOPS</td>
<td>Up to 14.7</td>
<td>Up to 13.4</td>
</tr>
<tr>
<td>Peak FP64 TFLOPS</td>
<td>Up to 7.4</td>
<td>Up to 6.7</td>
</tr>
<tr>
<td>Memory Size</td>
<td>32 GB HBM2</td>
<td>16 GB HBM2</td>
</tr>
<tr>
<td>Memory Bandwidth</td>
<td>1 TB/s</td>
<td>1 TB/s</td>
</tr>
<tr>
<td>Memory Interface</td>
<td>4096 bits</td>
<td>4096 bits</td>
</tr>
<tr>
<td>PCIe Gen4 Capable</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Infinity Fabric Link</td>
<td>2 Links</td>
<td>2 Links</td>
</tr>
</tbody>
</table>

Intel’s XE GPU codenamed “Ponte Vecchio” Based on 7nm architecture
  Intel’s first Exascale GPU
Will use Intel’s Embedded Multi-die interconnect Bridge (EMIB)
  Join multiple chiplets together
GPU to GPU communication via Computer eXpress Link (CXL) interface
  Layered on top of PCIe 5.0
Ponte Vecchio will be showcased in Aurora supercomputer
  To be installed at Argonne National Laboratory in 2021
  6x Ponte Vecchio per node
Intel’s newly introduced “OneAPI” will be the primary programming model
Hardware for DNN Training and Inference: TPUs

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

**Courtesy:** https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu
https://www.nextplatform.com/2017/04/05/first-depth-look-googles-tpu-architecture/
**Hardware for DNN Training and Inference: IPUs**

- Specifically designed for AI workloads – an Intelligence Processing Unit (IPU)
  - Massively parallel
  - Low-precision floating-point compute
  - Higher compute density
- Early benchmarks show 10-100x speedup over GPUs
  - Presented at NIPS 2017
- HPC Wire: Microsoft Azure IPU instances

**Courtesy:** [https://www.graphcore.ai/posts/preliminary-ipu-benchmarks-providing-previously-unseen-performance-for-a-range-of-machine-learning-applications](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/PoplarC2%81%20technical%20overview%20NEW%20BRAND.pdf
Hardware for DNN Training: Habana Gaudi

- Habana Labs – Training Accelerator called Gaudi – (HotChips ‘19)
- Gaudi – AI processor with RoCE integrated
- Gaudi software – Enables high-level frameworks
- **Intel has acquired Habana for $2 billion!**


**Figure 1. Gaudi emulated performance.** For training the simple ResNet-50 model, Habana’s Gaudi card offers throughput similar to that of Nvidia’s high-end V100 GPU at half the power. It also beats Nvidia’s Tesla T4 card in performance per watt.

**Hardware for DNN Training: Cerebras**

- Cerebras: First-Gen Wafer-Scale Engine (WSE) contains 400,000 Sparse Linear Algebra Compute (SLAC) Cores
- Swarm Communication fabric in a 2D mesh with 100 Pb/s of bandwidth
- Teased World’s Largest Chip with **2.6 Trillion** 7nm Transistors and 850000 Cores (HotChips ‘20)

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></th>
<th>NVMe</th>
<th>NVRAM</th>
<th>3D XPoint</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Definition</strong></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>
<td>Non-volatile high performance (1000X NAND), high density (6-10X DRAM), high endurance (1000X NAND) byte addressable memory</td>
</tr>
<tr>
<td><strong>Form Factor</strong></td>
<td>Connects to PCIe bus</td>
<td>Connects to a DDR3 DIMM slot</td>
<td>Connects to a DDR3 DIMM slot</td>
</tr>
<tr>
<td><strong>Max Capacity</strong></td>
<td>2 TB</td>
<td>16GB</td>
<td>128 GB</td>
</tr>
<tr>
<td><strong>Read IOPS (Random)</strong></td>
<td>750,000</td>
<td>1.4 Million</td>
<td>In millions</td>
</tr>
<tr>
<td><strong>Write IOPS (Random)</strong></td>
<td>430,000</td>
<td>1.4 Million</td>
<td>In millions</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td>15 Microsecond</td>
<td>10 Nanoseconds</td>
<td>10 Nanoseconds</td>
</tr>
<tr>
<td><strong>Ideal Use Cases</strong></td>
<td>Caching Tier: Transactional workloads requiring high IOPS</td>
<td>Byte Addressable memory for metadata &amp; client side caching, reduce write amplification</td>
<td>Highly Dense Byte Addressable memory for high speed caching, staging deduplication</td>
</tr>
<tr>
<td><strong>Price ($/Gig)</strong></td>
<td>$$</td>
<td>$$ $$</td>
<td>$$ $$</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 today).

- 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-path-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.

---

**NVMe Enabled Host**

**NVMe over Fabrics**

**NVMe Subsystem**
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>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter</strong></td>
</tr>
<tr>
<td>count</td>
</tr>
<tr>
<td>datatype</td>
</tr>
<tr>
<td>root</td>
</tr>
<tr>
<td>comm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input/Output Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter</strong></td>
</tr>
<tr>
<td>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)
```

<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>

Sendbuf (Before)

```
T1  
1  
2  
3  
4  
```

Recvbuf (After)

```
T1  
4  
8  
12  
16  
T2  
4  
8  
12  
16  
T3  
4  
8  
12  
16  
T4  
4  
8  
12  
16  
```
Overview of the MVAPICH2 Project

- High Performance open-source MPI Library
- Support for multiple interconnects
  - InfiniBand, Omni-Path, Ethernet/iWARP, RDMA over Converged Ethernet (RoCE), and AWS EFA
- Support for multiple platforms
  - x86, OpenPOWER, ARM, Xeon-Phi, GPGPUs (NVIDIA and AMD)
- Started in 2001, first open-source version demonstrated at SC ’02
- Supports the latest MPI-3.1 standard
- [http://mvapich.cse.ohio-state.edu](http://mvapich.cse.ohio-state.edu)
- Additional optimized versions for different systems/environments:
  - MVAPICH2-X (Advanced MPI + PGAS), since 2011
  - MVAPICH2-GDR with support for NVIDIA GPGPUs, since 2014
  - MVAPICH2-MIC with support for Intel Xeon-Phi, since 2014
  - MVAPICH2-Virt with virtualization support, since 2015
  - MVAPICH2-EA with support for Energy-Awareness, since 2015
  - MVAPICH2-Azure for Azure HPC IB instances, since 2019
  - MVAPICH2-X-AWS for AWS HPC+EFA instances, since 2019
- Tools:
  - OSU MPI Micro-Benchmarks (OMB), since 2003
  - OSU InfiniBand Network Analysis and Monitoring (INAM), since 2015

- Used by more than 3,200 organizations in 89 countries
- More than 1.46 Million downloads from the OSU site directly
- Empowering many TOP500 clusters (June ‘21 ranking)
  - 4th, 10,649,600-core (Sunway TaihuLight) at NSC, Wuxi, China
  - 10th, 448,448 cores (Frontera) at TACC
  - 20th, 288,288 cores (Lassen) at LLNL
  - 31st, 570,020 cores (Nurion) in South Korea and many others
- Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, OpenHPC, and Spack)
- Partner in the 10th ranked TACC Frontera system
- Empowering Top500 systems for more than 15 years
MPI + CUDA - Naive

• Data movement in applications with standard MPI and CUDA interfaces

At Sender:

cudaMemcpy(s_hostbuf, s_devbuf, ...);
MPI_Send(s_hostbuf, size, ...);

At Receiver:

MPI_Recv(r_hostbuf, size, ...);
cudaMemcpy(r_devbuf, r_hostbuf, ...);

High Productivity and Low Performance
Pipelining at user level with non-blocking MPI and CUDA interfaces

**At Sender:**

```c
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:

MPISend(s_devbuf, size, ...);

At Receiver:

MPI_Recv(r_devbuf, size, ...);

High Performance and High Productivity
Optimized MVAPICH2-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

**MVAPICH2-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

- NVIDIA Collective Communication Library (NCCL)
- Main Motivation: Deep Learning workloads
- NCCL1—efficient dense-GPU communication within the node
- NCCL2—multiple DGX systems connected to each other with InfiniBand systems

Courtesy: https://developer.nvidia.com/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/
Performance of NCCL 1.x Collectives

NCCL 2: Multi-node GPU Collectives

[Graph showing CNTK scaling for ResNet50, images/s with data points for Ideal, MPI, and NCCL]