High-Performance and Scalable Designs of Programming Models for Exascale Systems


by

Dhabaleswar K. (DK) Panda
The Ohio State University
E-mail: panda@cse.ohio-state.edu
http://www.cse.ohio-state.edu/~panda
High-End Computing (HEC): PetaFlop to ExaFlop

100 PFlops in 2016

1 EFlops in 2020-2024?
Trends for Commodity Computing Clusters in the Top 500 List (http://www.top500.org)

![Graph showing the percentage and number of clusters over time, reaching 86% by Apr-14.](image-url)
Drivers of Modern HPC Cluster Architectures

- Multi-core processors are ubiquitous
- InfiniBand very popular in HPC clusters
- Accelerators/Coprocessors becoming common in high-end systems
- Pushing the envelope for Exascale computing
## Large-scale InfiniBand Installations

- 225 IB Clusters (45%) in the November 2014 Top500 list ([http://www.top500.org](http://www.top500.org))
- Installations in the Top 50 (21 systems):

<table>
<thead>
<tr>
<th>Installations</th>
<th>Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>519,640 cores (Stampede) at TACC (7th)</td>
<td>73,584 cores (Spirit) at USA/Air Force (31st)</td>
</tr>
<tr>
<td>72,800 cores Cray CS-Storm in US (10th)</td>
<td>77,184 cores (Curie thin nodes) at France/CEA (33rd)</td>
</tr>
<tr>
<td>160,768 cores (Pleiades) at NASA/Ames (11th)</td>
<td>65,320-cores, iDataPlex DX360M4 at Germany/Max-Planck (34th)</td>
</tr>
<tr>
<td>72,000 cores (HPC2) in Italy (12th)</td>
<td>120,640 cores (Nebulae) at China/NSCS (35th)</td>
</tr>
<tr>
<td>147,456 cores (Super MUC) in Germany (14th)</td>
<td>72,288 cores (Yellowstone) at NCAR (36th)</td>
</tr>
<tr>
<td>76,032 cores (Tsubame 2.5) at Japan/GSIC (15th)</td>
<td>70,560 cores (Helios) at Japan/IFERC (38th)</td>
</tr>
<tr>
<td>194,616 cores (Cascade) at PNNL (18th)</td>
<td>38,880 cores (Cartesisu) at Netherlands/SURFsara (45th)</td>
</tr>
<tr>
<td>110,400 cores (Pangea) at France/Total (20th)</td>
<td>23,040 cores (QB-2) at LONI/US (46th)</td>
</tr>
<tr>
<td>37,120 cores T-Platform A-Class at Russia/MSU (22nd)</td>
<td>138,368 cores (Tera-100) at France/CEA (47th)</td>
</tr>
<tr>
<td>50,544 cores Occigen at France/GENCI-CINES (26th)</td>
<td>and many more!</td>
</tr>
</tbody>
</table>
Two Major Categories of Applications

• **Scientific Computing**
  – Message Passing Interface (MPI), including MPI + OpenMP, is the Dominant Programming Model
  – Many discussions towards Partitioned Global Address Space (PGAS)
    • UPC, OpenSHMEM, CAF, etc.
  – Hybrid Programming: MPI + PGAS (OpenSHMEM, UPC)

• **Big Data/Enterprise/Commercial Computing**
  – Focuses on large data and data analysis
  – Hadoop (HDFS, HBase, MapReduce)
  – Spark is emerging for in-memory computing
  – Memcached is also used for Web 2.0
## Towards Exascale System (Today and Target)

<table>
<thead>
<tr>
<th>Systems</th>
<th>2014 Tianhe-2</th>
<th>2020-2024</th>
<th>Difference Today &amp; Exascale</th>
</tr>
</thead>
<tbody>
<tr>
<td>System peak</td>
<td>55 PFlop/s</td>
<td>1 EFlop/s</td>
<td>~20x</td>
</tr>
<tr>
<td>Power</td>
<td>18 MW (3 Gflops/W)</td>
<td>~20 MW (50 Gflops/W)</td>
<td>O(1) ~15x</td>
</tr>
<tr>
<td>System memory</td>
<td>1.4 PB (1.024PB CPU + 0.384PB CoP)</td>
<td>32 – 64 PB</td>
<td>~50X</td>
</tr>
<tr>
<td>Node performance</td>
<td>3.43TF/s (0.4 CPU + 3 CoP)</td>
<td>1.2 or 15 TF</td>
<td>O(1)</td>
</tr>
<tr>
<td>Node concurrency</td>
<td>24 core CPU + 171 cores CoP</td>
<td>O(1k) or O(10k)</td>
<td>~5x - ~50x</td>
</tr>
<tr>
<td>Total node interconnect BW</td>
<td>6.36 GB/s</td>
<td>200 – 400 GB/s</td>
<td>~40x - ~60x</td>
</tr>
<tr>
<td>System size (nodes)</td>
<td>16,000</td>
<td>O(100,000) or O(1M)</td>
<td>~6x - ~60x</td>
</tr>
<tr>
<td>Total concurrency</td>
<td>3.12M 12.48M threads (4 /core)</td>
<td>O(billion) for latency hiding</td>
<td>~100x</td>
</tr>
<tr>
<td>MTTI</td>
<td>Few/day</td>
<td>Many/day</td>
<td>O(?)</td>
</tr>
</tbody>
</table>

**Courtesy: Prof. Jack Dongarra**
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.
Basic Design Challenges for Exascale Systems

- **Energy and Power Challenge**
  - Hard to solve power requirements for data movement

- **Memory and Storage Challenge**
  - Hard to achieve high capacity and high data rate

- **Concurrency and Locality Challenge**
  - Management of very large amount of concurrency (*billion* threads)

- **Resiliency Challenge**
  - Low voltage devices (for low power) introduce more faults
How does MPI Plan to Meet Exascale Challenges?

• Power required for data movement operations is one of the main challenges

• Non-blocking collectives
  – Overlap computation and communication

• Much improved One-sided interface
  – Reduce synchronization of sender/receiver

• Manage concurrency
  – Improved interoperability with PGAS (e.g. UPC, Global Arrays, OpenSHMEM, CAF)

• Resiliency
  – New interface for detecting failures
Major New Features in MPI-3

• Major features
  – Non-blocking Collectives
  – Improved One-Sided (RMA) Model
  – MPI Tools Interface

• Specification is available from: http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf

• MPI 3.1 and 4.0 are underway
Partitioned Global Address Space (PGAS) Models

• Key features
  - Simple shared memory abstractions
  - Light weight one-sided communication
  - Easier to express irregular communication

• Different approaches to PGAS
  - Languages
    • Unified Parallel C (UPC)
    • Co-Array Fortran (CAF)
    • X10
  - Libraries
    • OpenSHMEM
    • Global Arrays
    • Chapel
Hierarchical architectures with multiple address spaces

(MPI + PGAS) Model
  – MPI across address spaces
  – PGAS within an address space

MPI is good at moving data between address spaces

Within an address space, MPI can interoperate with other shared memory programming models

Can co-exist with OpenMP for offloading computation

Applications can have kernels with different communication patterns

Can benefit from different models

Re-writing complete applications can be a huge effort

Port critical kernels to the desired model instead
Hybrid (MPI+PGAS) Programming

- Application sub-kernels can be re-written in MPI/PGAS based on communication characteristics

- Benefits:
  - Best of Distributed Computing Model
  - Best of Shared Memory Computing Model

- Exascale Roadmap*:
  - “Hybrid Programming is a practical way to program exascale systems”

Designing Communication Libraries for Multi-Petaflop and Exaflop Systems: Challenges

**Application Kernels/Applications**

**Middleware**

**Programming Models**
- MPI, PGAS (UPC, Global Arrays, OpenSHMEM), CUDA, OpenACC, Cilk, Hadoop, MapReduce, etc.

**Communication Library or Runtime for Programming Models**
- Point-to-point Communication (two-sided & one-sided)
- Collective Communication
- Synchronization & Locks
- I/O & File Systems
- Fault Tolerance

**Networking Technologies**
- (InfiniBand, 40/100GigE, Aries, and Omni Scale)

**Multi/Many-core Architectures**

**Accelerators**
- (NVIDIA and MIC)

Co-Design Opportunities and Challenges across Various Layers

Performance
Scalability
Fault-Resilience
Broad Challenges in Designing Communication Libraries for (MPI+X) at Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)

- Scalable Collective communication
  - Offload
  - Non-blocking
  - Topology-aware
  - Power-aware

- Balancing intra-node and inter-node communication for next generation multi-core (128-1024 cores/node)
  - Multiple end-points per node

- Support for efficient multi-threading

- Integrated Support for GPGPUs and Accelerators

- Fault-tolerance/resiliency

- QoS support for communication and I/O

- Support for Hybrid MPI+PGAS programming (MPI + OpenMP, MPI + UPC, MPI + OpenSHMEM, CAF, ...)

- Virtualization
Additional Challenges for Designing Exascale Software Libraries

• Extreme Low Memory Footprint
  – Memory per core continues to decrease

• D-L-A Framework
  – Discover
    • Overall network topology (fat-tree, 3D, …)
    • Network topology for processes for a given job
    • Node architecture
    • Health of network and node
  – Learn
    • Impact on performance and scalability
    • Potential for failure
  – Adapt
    • Internal protocols and algorithms
    • Process mapping
    • Fault-tolerance solutions
  – Low overhead techniques while delivering performance, scalability and fault-tolerance
MVAPICH2 Software

- High Performance open-source MPI Library for InfiniBand, 10Gig/iWARP, and RDMA over Converged Enhanced Ethernet (RoCE)
  - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.0), Available since 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2012
  - Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  - Used by more than 2,300 organizations in 75 countries
  - More than 232,000 downloads from the OSU site directly
  - Empowering many TOP500 clusters (Nov ‘14 ranking)
    - 7th ranked 519,640-core cluster (Stampede) at TACC
    - 11th ranked 160,768-core cluster (Pleiades) at NASA
    - 15th ranked 76,032-core cluster (Tsubame 2.5) at Tokyo Institute of Technology and many others
  - Available with software stacks of many IB, HSE, and server vendors including Linux Distros (RedHat and SuSE)
    - http://mvapich.cse.ohio-state.edu

- System-X from Virginia Tech (3rd in Nov 2003, 2,200 processors, 12.25 TFlops) ->

Stampede at TACC (7th in Nov 2014, 519,640 cores, 5.168 Plops)
Overview of A Few Challenges being Addressed by MVAPICH2 Project for Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided RMA)
  - Extremely minimum memory footprint

- Collective communication
  - Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware

- Integrated Support for GPGPUs
- Integrated Support for Intel MICs
- MPI-T Interface
- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, ...)
- Virtualization
Latency & Bandwidth: MPI over IB with MVAPICH2

Small Message Latency

- Qlogic-DDR
- Qlogic-QDR
- ConnectX-DDR
- ConnectX2-PCIe2-QDR
- ConnectX3-PCIe3-FDR
- Sandy-ConnectIB-DualFDR
- Ivy-ConnectIB--DualFDR

Unidirectional Bandwidth

- DDR, QDR - 2.4 GHz Quad-core (Westmere) Intel PCI Gen2 with IB switch
- FDR - 2.6 GHz Octa-core (SandyBridge) Intel PCI Gen3 with IB switch
- ConnectIB-Dual FDR - 2.6 GHz Octa-core (SandyBridge) Intel PCI Gen3 with IB switch
- ConnectIB-Dual FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch
MVAPICH2 Two-Sided Intra-Node Performance
(Shared memory and Kernel-based Zero-copy Support (LiMIC and CMA))

Latest MVAPICH2 2.1rc1
Intel Ivy-bridge

HPC Advisory Council Stanford Conference (Feb '15)
MPI-3 RMA Get/Put with Flush Performance

Latest MVAPICH2 2.1rc1, Intel Sandy-bridge with Connect-IB (single-port)
Minimizing Memory Footprint with XRC and Hybrid Mode

- Memory usage for 32K processes with 8-cores per node can be 54 MB/process (for connections)
- NAMD performance improves when there is frequent communication to many peers

- Both UD and RC/XRC have benefits
  - Hybrid for the best of both
- Available since MVAPICH2 1.7 as integrated interface
- Runtime Parameters: RC - default;
  - UD - MV2_USE_ONLY_UD=1
  - Hybrid - MV2_HYBRID_ENABLE_THRESHOLD=1

M. Koop, J. Sridhar and D. K. Panda, “Scalable MPI Design over InfiniBand using eXtended Reliable Connection,” Cluster ’08
Minimizing Memory Footprint further by DC Transport

- Constant connection cost (*One QP for any peer*)
- Full Feature Set (RDMA, Atomics etc)
- Separate objects for send (DC Initiator) and receive (DC Target)
  - DC Target identified by “DCT Number”
  - Messages routed with (DCT Number, LID)
  - Requires same “DC Key” to enable communication
- Initial study done in MVAPICH2
- DCT support available in Mellanox OFED

**Memory Footprint for Alltoall**

**NAMD - Apoa1: Large data set**

Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-X for Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided RMA)
  - Extremely minimum memory footprint
- Collective communication
  - Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware
- Integrated Support for GPGPUs
- Integrated Support for Intel MICs
- MPI-T Interface
- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, ...)
- Virtualization
Hardware Multicast-aware MPI_Bcast on Stampede

Small Messages (102,400 Cores)

- Default
- Multicast

Latency (us)

Message Size (Bytes)

2  8  32  128  512

Large Messages (102,400 Cores)

- Default
- Multicast

Latency (us)

Message Size (Bytes)

2K  8K  32K  128K

16 Byte Message

- Default
- Multicast

Latency (us)

Number of Nodes

16  32  64  128  512  1K  2K  4K  6K

32 KByte Message

- Default
- Multicast

Latency (us)

Number of Nodes

16  32  64  128  512  1K  2K  4K  6K

ConnectX-3-FDR (54 Gbps): 2.7 GHz Dual Octa-core (SandyBridge) Intel PCI Gen3 with Mellanox IB FDR switch
Co-Design with MPI-3 Non-Blocking Collectives and Collective Offload Hardware

Modified P3DFFT with Offload-Alltoall does up to 17% better than default version (128 Processes)

Modified Pre-Conjugate Gradient Solver with Offload-Allreduce does up to 21.8% better than default version

K. Kandalla, et. al., High-Performance and Scalable Non-Blocking All-to-All with Collective Offload on InfiniBand Clusters: A Study with Parallel 3D FFT, ISC 2011
K. Kandalla, et. al, Designing Non-blocking Broadcast with Collective Offload on InfiniBand Clusters: A Case Study with HPL, HotI 2011
K. Kandalla, et. al., Designing Non-blocking Allreduce with Collective Offload on InfiniBand Clusters: A Case Study with Conjugate Gradient Solvers, IPDPS ’12

Network-Topology-Aware Placement of Processes

How do we design the MPI communication library in a network-topology-aware manner to efficiently leverage the topology information?

What are the potential benefits of using a network-topology-aware MPI library on the performance of parallel scientific applications?

Overall performance and Split up of physical communication for MILC on Ranger

- Reduce network topology discovery time from $O(N_{\text{hosts}}^2)$ to $O(N_{\text{hosts}})$
- 15% improvement in MILC execution time @ 2048 cores
- 15% improvement in Hypre execution time @ 1024 cores

Power and Energy Savings with Power-Aware Collectives

Performance and Power Comparison: MPI_Alltoall with 64 processes on 8 nodes


HPC Advisory Council Stanford Conference (Feb '15)
Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-X for Exascale

- **Scalability for million to billion processors**
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided RMA)
  - Extremely minimum memory footprint

- **Collective communication**
  - Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware

- **Integrated Support for GPGPUs**
- **Integrated Support for Intel MICs**
- **MPI-T Interface**
- **Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, ...)**
- **Virtualization**
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
MPI + CUDA - Advanced

- 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 = cudaMemcpyAsync(...);
        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 MPI Library: MVAPICH2-GPU

- 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:

\[ \text{MPI}_\text{Send}(s\_\text{devbuf}, \text{size}, \ldots); \]

At Receiver:

\[ \text{MPI}_\text{Recv}(r\_\text{devbuf}, \text{size}, \ldots); \]

High Performance and High Productivity
CUDA-Aware MPI: MVAPICH2 1.8-2.1 Releases

- Support for MPI communication from NVIDIA GPU device memory
- High performance RDMA-based inter-node point-to-point communication (GPU-GPU, GPU-Host and Host-GPU)
- High performance intra-node point-to-point communication for multi-GPU adapters/node (GPU-GPU, GPU-Host and Host-GPU)
- Taking advantage of CUDA IPC (available since CUDA 4.1) in intra-node communication for multiple GPU adapters/node
- Optimized and tuned collectives for GPU device buffers
- MPI datatype support for point-to-point and collective communication from GPU device buffers
GPU-Direct RDMA (GDR) with CUDA

- OFED with support for GPUDirect RDMA is developed by NVIDIA and Mellanox
- OSU has a design of MVAPICH2 using GPUDirect RDMA
  - Hybrid design using GPU-Direct RDMA
    - GPUDirect RDMA and Host-based pipelining
    - Alleviates P2P bandwidth bottlenecks on SandyBridge and IvyBridge
  - Support for communication using multi-rail
  - Support for Mellanox Connect-IB and ConnectX VPI adapters
  - Support for RoCE with Mellanox ConnectX VPI adapters
Performance of MVAPICH2 with GPU-Direct-RDMA: Latency

GPU-GPU Internode MPI Latency

Small Message Latency

- MV2-GDR2.0
- MV2-GDR2.0b
- MV2 w/o GDR

Message Size (bytes) vs. Latency (us)

- MVAPICH2-GDR-2.0
- Intel Ivy Bridge (E5-2680 v2) node with 20 cores
- NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
- CUDA 6.5, Mellanox OFED 2.1 with GPU-Direct-RDMA

- 90% reduction
- 71% reduction
- 2.18 usec
Performance of MVAPICH2 with GPU-Direct-RDMA: Bandwidth

GPU-GPU Internode MPI Uni-Directional Bandwidth

Small Message Bandwidth

- MV2-GDR2.0
- MV2-GDR2.0b
- MV2 w/o GDR

Bandwidth (MB/s)

Message Size (bytes)

MVAPICH2-GDR-2.0
Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 6.5, Mellanox OFED 2.1 with GPU-Direct-RDMA
Performance of MVAPICH2 with GPU-Direct-RDMA: Bi-Bandwidth

GPU-GPU Internode MPI Bi-directional Bandwidth

Small Message Bi-Bandwidth

- MV2-GDR2.0
- MV2-GDR2.0b
- MV2 w/o GDR

MVAPICH2-GDR-2.0
Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 6.5, Mellanox OFED 2.1 with GPU-Direct-RDMA

Message Size (bytes)

Bi-Bandwidth (MB/s)
Performance of MVAPICH2 with GPU-Direct-RDMA: MPI-3 RMA

GPU-GPU Internode MPI Put latency (RMA put operation Device to Device)

MPI-3 RMA provides flexible synchronization and completion primitives

![Graph showing small message latency vs message size](graph.png)

- **MVAPICH2-GDR2.0**
  - Intel Ivy Bridge (E5-2680 v2) node with 20 cores
  - NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
  - CUDA 6.5, Mellanox OFED 2.1 with GPU-Direct-RDMA
Application-Level Evaluation (HOOMD-blue)

- Platform: Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- MV2-GDR 2.1a (released on 12/20/14): Try it out!!
  - GDRCOPY enabled: MV2_USE_CUDA=1 MV2_IBA_HCA=mlx5_0
    MV2_IBA_EAGER_THRESHOLD=32768 MV2_VBUF_TOTAL_SIZE=32768
    MV2_USE_GPUDIRECT_LOOPBACK_LIMIT=32768 MV2_USE_GPUDIRECT_GDRCOPY=1
    MV2_USE_GPUDIRECT_GDRCOPY_LIMIT=16384
• MVAPICH2-2.1a with GDR support can be downloaded from
  https://mvapich.cse.ohio-state.edu/download/#mv2gdr/

• System software requirements
  • Mellanox OFED 2.1 or later
  • NVIDIA Driver 331.20 or later
  • NVIDIA CUDA Toolkit 6.0 or later
  • Plugin for GPUDirect RDMA
    – Strongly Recommended: use the new GDRCOPY module from NVIDIA
      • https://github.com/NVIDIA/gdrcopy

• Has optimized designs for point-to-point communication using GDR

• Contact MVAPICH help list with any questions related to the package
  mvapich-help@cse.ohio-state.edu
Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-X for Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided RMA)
  - Extremely minimum memory footprint
- Collective communication
  - Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware
- Integrated Support for GPGPUs
- Integrated Support for Intel MICs
- MPI-T Interface
- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, ...)
- Virtualization
MPI Applications on MIC Clusters

• Flexibility in launching MPI jobs on clusters with Xeon Phi

Multi-core Centric

Host-only

Offload
(/reverse Offload)

Symmetric

Coprocessor-only

Many-core Centric

Xeon

MPI Program

MPI Program

MPI Program

Xeon Phi

Offloaded Computation

MPI Program

MPI Program
Data Movement on Intel Xeon Phi Clusters

- Connected as PCIe devices – Flexibility but Complexity

1. Intra-Socket
2. Inter-Socket
3. Inter-Node
4. Intra-MIC
5. Intra-Socket MIC-MIC
6. Inter-Socket MIC-MIC
7. Inter-Node MIC-MIC
8. Intra-Socket MIC-Host
9. Inter-Socket MIC-Host
10. Inter-Node MIC-Host
11. Inter-Node MIC-MIC with IB adapter on remote socket and more . . .

- Critical for runtimes to optimize data movement, hiding the complexity

HPC Advisory Council Stanford Conference (Feb ’15)
MVAPICH2-MIC Design for Clusters with IB and MIC

- Offload Mode
- Intranode Communication
  - Coprocessor-only Mode
  - Symmetric Mode
- Internode Communication
  - Coprocessors-only
  - Symmetric Mode
- Multi-MIC Node Configurations
MIC-Remote-MIC P2P Communication with Proxy-based Communication

**Intra-socket P2P**

**Latency (Large Messages)**

- MV2
- MV2-MIC

**Bandwidth**

**Inter-socket P2P**

**Latency (Large Messages)**

**Bandwidth**

**HPC Advisory Council Stanford Conference (Feb '15)**
Optimized MPI Collectives for MIC Clusters (Allgather & Alltoall)

32-Node-Allgather (16H + 16 M) Small Message Latency

- MV2-MIC
- MV2-MIC-Opt

76%

32-Node-Allgather (8H + 8 M) Large Message Latency

- MV2-MIC
- MV2-MIC-Opt

58%

32-Node-Alltoall (8H + 8 M) Large Message Latency

- MV2-MIC
- MV2-MIC-Opt

55%

P3DFFT Performance

- Communication
- Computation


HPC Advisory Council Stanford Conference (Feb '15)
Latest Status on MVAPICH2-MIC

• Running on three major systems
  – Stampede
  – Blueridge (Virginia Tech)
  – Beacon (UTK)

• Public version of MVAPICH2-MIC 2.0 is released (12/02/14)

• Try it out!!
  – http://mvapich.cse.ohio-state.edu/downloads/#mv2mic
Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-X for Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
  - Extremely minimum memory footprint
- Collective communication
  - Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware
- Integrated Support for GPGPUs
- Integrated Support for Intel MICs
- MPI-T Interface
- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, ...)
- Virtualization
MPI Tools Interface

• Introduced to expose internals of MPI tools and applications
• Generalized interface – no defined variables in the standard
• Variables can differ between
  • MPI implementations
  • Compilations of same MPI library (production vs debug)
  • Executions of the same application/MPI library
  • There could be no variables provided
• Two types of variables supported
  • **Control Variables (CVARS)**
    • Typically used to configure and tune MPI internals
    • Environment variables, configuration parameters and toggles
  • **Performance Variables (PVARS)**
    • Insights into performance of an MPI library
    • Highly-implementation specific
    • Memory consumption, timing information, resource-usage, data transmission info.
    • Per-call basis or an entire MPI job
Co-designing Applications to use MPI-T

Example: Optimizing the eager limit dynamically ->

Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-X for Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided RMA)
  - Extremely minimum memory footprint
- Collective communication
  - Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware
- Integrated Support for GPGPUs
- Integrated Support for Intel MICs
- MPI-T Interface
- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, ...)
- Virtualization
MVAPICH2-X for Hybrid MPI + PGAS Applications

- Unified communication runtime for MPI, UPC, OpenSHMEM available with MVAPICH2-X 1.9 onwards!
  - [http://mvapich.cse.ohio-state.edu](http://mvapich.cse.ohio-state.edu)

- Feature Highlights
  - Supports MPI(+OpenMP), OpenSHMEM, UPC, MPI(+OpenMP) + OpenSHMEM, MPI(+OpenMP) + UPC
  - MPI-3 compliant, OpenSHMEM v1.0 standard compliant, UPC v1.2 standard compliant
  - Scalable Inter-node and Intra-node communication – point-to-point and collectives
OpenSHMEM Application Evaluation

- Improved performance for OMPI-SHMEM and Scalable-SHMEM with FCA
- Execution time for 2D Heat Image at 512 processes (sec):
- Execution time for DAXPY at 512 processes (sec):

Hybrid MPI+OpenSHMEM Graph500 Design

- Performance of Hybrid (MPI+OpenSHMEM) Graph500 Design
  - 8,192 processes
    - 2.4X improvement over MPI-CSR
    - 7.6X improvement over MPI-Simple
  - 16,384 processes
    - 1.5X improvement over MPI-CSR
    - 13X improvement over MPI-Simple

J. Jose, S. Potluri, K. Tomko and D. K. Panda, Designing Scalable Graph500 Benchmark with Hybrid MPI+OpenSHMEM Programming Models, International Supercomputing Conference (ISC’13), June 2013
J. Jose, K. Kandalla, M. Luo and D. K. Panda, Supporting Hybrid MPI and OpenSHMEM over InfiniBand: Design and Performance Evaluation, Int'l Conference on Parallel Processing (ICPP '12), September 2012
Hybrid MPI+OpenSHMEM Sort Application

- Performance of Hybrid (MPI+OpenSHMEM) Sort Application
  - Execution Time
    - 4TB Input size at 4,096 cores: MPI – 2408 seconds, Hybrid: 1172 seconds
    - 51% improvement over MPI-based design
  - Strong Scalability (configuration: constant input size of 500GB)
    - At 4,096 cores: MPI – 0.16 TB/min, Hybrid – 0.36 TB/min
    - 55% improvement over MPI based design

MVAPICH2-X Support of OpenUH CAF

- Support multiple networks through different conduits: MVAPICH2-X Conduit is available in MVAPICH2-X release, which support CAF and Hybrid MPI+CAF on InfiniBand
- Optimized Collective designs


HPC Advisory Council Stanford Conference (Feb '15)
Performance Evaluations for One-sided Communication

- Micro-benchmark improvement (MV2X vs. GASNet-IBV, UH CAF test-suite)
  - Put bandwidth: 3.5X improvement on 4KB; Put latency: reduce 29% on 4B
- Application performance improvement (NAS-CAF one-sided implementation)
  - Reduce the execution time by 12% (SP.D.256), 18% (BT.D.256)
Performance Evaluations for Collective Communication

Reduce on 64 cores

- Bandwidth improvement (MV2X-Hybrid vs. GASNet-IBV, UH CAF test-suite)
  - CO_REDUCE: 2.3X improvement on 1KB, 1.1X improvement on 128KB
  - CO_BROADCAST: 4.0X improvement on 1KB, 1.3X improvement on 1MB
  - CAF support will be available in the next release of MVAPICH2-X
Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-X for Exascale

• Scalability for million to billion processors
  – Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided RMA)
  – Extremely minimum memory footprint
• Collective communication
  – Hardware-multicast-based
  – Offload and Non-blocking
  – Topology-aware
  – Power-aware
• Integrated Support for GPGPUs
• Integrated Support for Intel MICs
• MPI-T Interface
• Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, ...)
• Virtualization
Can HPC and Virtualization be Combined?

- Virtualization has many benefits
  - Job migration
  - Compaction
- Not very popular in HPC due to overhead associated with Virtualization
- New SR-IOV (Single Root – IO Virtualization) support available with Mellanox InfiniBand adapters
- Initial designs of MVAPICH2 with SR-IOV support with Openstack
- Will be available publicly soon

J. Zhang, X. Lu, J. Jose, R. Shi and D. K. Panda, Can Inter-VM Shmem Benefit MPI Applications on SR-IOV based Virtualized InfiniBand Clusters? EuroPar’14

J. Zhang, X. Lu, J. Jose, M. Li, R. Shi and D.K. Panda, High Performance MPI Libray over SR-IOV enabled InfiniBand Clusters, HiPC’14

IB-Level Performance – Inter-node Pt2Pt Latency & Bandwidth

- Native vs. SR-IOV at basic IB level
- Compared to native latency, 0-24% overhead
- Compared to native bandwidth, 0-17% overhead
MPI-Level Performance – Inter-node Pt2Pt Latency & Bandwidth

- EC2 C3.2xlarge instances
- Similar performance with MV2-SR-IOV-Def
- Compared to Native, similar overhead as basic IB level
- Compared to EC2, up to 29X and 16X performance speedup on Latency & Bandwidth
**MPI-Level Performance – Intra-node Pt2Pt Latency & Bandwidth**

- EC2 C3.2xlarge instances
- Compared to MV2-SR-IOV-Def, up to 84% and 158% performance improvement on Latency & Bandwidth
- Compared to Native, 3-7% overhead for Latency, 3-8% overhead for Bandwidth
- Compared to EC2, up to 160X and 28X performance speedup on Latency & Bandwidth
Application-Level Performance (8 VM * 8 Core/VM)

- Compared to Native, 1-9% overhead for NAS
- Compared to Native, 4-9% overhead for Graph500
NSF Chameleon Cloud: A Powerful and Flexible Experimental Instrument

• Large-scale instrument
  – Targeting Big Data, Big Compute, Big Instrument research
  – ~650 nodes (~14,500 cores), 5 PB disk over two sites, 2 sites connected with 100G network

• Reconfigurable instrument
  – Bare metal reconfiguration, operated as single instrument, graduated approach for ease-of-use

• Connected instrument
  – Workload and Trace Archive
  – Partnerships with production clouds: CERN, OSDC, Rackspace, Google, and others
  – Partnerships with users

• Complementary instrument
  – Complementing GENI, Grid’5000, and other testbeds

• Sustainable instrument
  – Industry connections

http://www.chameleoncloud.org/
MVAPICH2 – Plans for Exascale

- Performance and Memory scalability toward 900K-1M cores
  - Dynamically Connected Transport (DCT) service with Connect-IB
- Enhanced Optimization for GPGPU and Coprocessor Support
  - Extending the GPGPU support (GPU-Direct RDMA) with CUDA 7.0 and Beyond
  - Support for Intel MIC (Knight Landing)
- Taking advantage of Collective Offload framework
  - Including support for non-blocking collectives (MPI 3.0)
- RMA support (as in MPI 3.0)
- Extended topology-aware collectives
- Power-aware communication
- Support for MPI Tools Interface (as in MPI 3.0)
- Checkpoint-Restart and migration support with in-memory checkpointing
- Hybrid MPI+PGAS programming support with GPGPUs and Accelerators
- Virtualization
Looking into the Future ....

• Exascale systems will be constrained by
  – Power
  – Memory per core
  – Data movement cost
  – Faults

• Programming Models and Runtimes need to be designed for
  – Scalability
  – Performance
  – Fault-resilience
  – Energy-awareness
  – Programmability
  – Productivity

• Highlighted some of the issues and challenges
• Need continuous innovation on all these fronts
More Details on Accelerating Big Data will be covered..

• Accelerating Big Data Processing with Hadoop, Spark and Memcached
  • (Feb 3rd, 11:00-11:30am)
Funding Acknowledgments

Funding Support by

[Logos of various sponsors]

Equipment Support by

[Logos of various sponsors]
Personnel Acknowledgments

Current Students
- A. Awan (Ph.D.)
- A. Bhat (M.S.)
- S. Chakraborty (Ph.D.)
- C.-H. Chu (Ph.D.)
- N. Islam (Ph.D.)
- M. Li (Ph.D.)
- M. Rahman (Ph.D.)
- D. Shankar (Ph.D.)
- A. Venkatesh (Ph.D.)
- J. Zhang (Ph.D.)

Past Students
- P. Balaji (Ph.D.)
- D. Buntinas (Ph.D.)
- S. Bhagvat (M.S.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)

Past Post-Docs
- H. Wang
- X. Besseron
- H.-W. Jin
- M. Luo
- E. Mancini
- S. Marcarelli
- J. Vienne

Past Senior Research Associates
- K. Hamidouche
- X. Lu
- J. Lin
- J. Perkins
- M. Arnold
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- R. Rajachandrasekar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)

Past Research Scientist
- S. Sur

Past Programmers
- D. Bureddy
- H. Subramoni
Thank You!

panda@cse.ohio-state.edu

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

The MVAPICH2 Project
http://mvapich.cse.ohio-state.edu/

The High-Performance Big Data Project
http://hibd.cse.ohio-state.edu/