New Developments in Message Passing, PGAS and Big Data

Talk at HPC Advisory Council China Workshop (2013)

by

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

欢迎
Current and Next Generation HPC Systems and Applications

- Growth of High Performance Computing (HPC)
  - Growth in processor performance
    - Chip density doubles every 18 months
  - Growth in commodity networking
    - Increase in speed/features + reducing cost
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) environment is gaining a lot of momentum
  – Memcached is also used for Web 2.0
Expected to have an ExaFlop system in 2019-2022!
Trends for Commodity Computing Clusters in the Top 500 List (http://www.top500.org)

Timeline

Percentage of Clusters

Number of Clusters

HPC Advisory Council China Workshop, Oct '13
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
## Exascale System Targets

<table>
<thead>
<tr>
<th></th>
<th>2010</th>
<th>2018</th>
<th>Difference 2010 &amp; 2018</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System peak</strong></td>
<td>2 PFlop/s</td>
<td>1 EFlop/s</td>
<td>O(1,000)</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>6 MW</td>
<td>~20 MW (goal)</td>
<td></td>
</tr>
<tr>
<td><strong>System memory</strong></td>
<td>0.3 PB</td>
<td>32 – 64 PB</td>
<td>O(100)</td>
</tr>
<tr>
<td><strong>Node performance</strong></td>
<td>125 GF</td>
<td>1.2 or 15 TF</td>
<td>O(10) – O(100)</td>
</tr>
<tr>
<td><strong>Node memory BW</strong></td>
<td>25 GB/s</td>
<td>2 – 4 TB/s</td>
<td>O(100)</td>
</tr>
<tr>
<td><strong>Node concurrency</strong></td>
<td>12</td>
<td>O(1k) or O(10k)</td>
<td>O(100) – O(1,000)</td>
</tr>
<tr>
<td><strong>Total node interconnect BW</strong></td>
<td>3.5 GB/s</td>
<td>200 – 400 GB/s</td>
<td>O(100)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(1:4 or 1:8 from memory BW)</td>
<td></td>
</tr>
<tr>
<td><strong>System size (nodes)</strong></td>
<td>18,700</td>
<td>O(100,000) or O(1M)</td>
<td>O(10) – O(100)</td>
</tr>
<tr>
<td><strong>Total concurrency</strong></td>
<td>225,000</td>
<td>O(billion) + [O(10) to O(100) for latency hiding]</td>
<td>O(10,000)</td>
</tr>
<tr>
<td><strong>Storage capacity</strong></td>
<td>15 PB</td>
<td>500 – 1000 PB (&gt;10x system memory is min)</td>
<td>O(10) – O(100)</td>
</tr>
<tr>
<td><strong>IO Rates</strong></td>
<td>0.2 TB</td>
<td>60 TB/s</td>
<td>O(100)</td>
</tr>
<tr>
<td><strong>MTTI</strong></td>
<td>Days</td>
<td>O(1 day)</td>
<td>-O(10)</td>
</tr>
</tbody>
</table>

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

• DARPA Exascale Report – Peter Kogge, Editor and Lead

• 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
Designing Software 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, BlueGene)

Multi/Many-core Architectures

Accelerators (NVIDIA and MIC)

Co-Design Opportunities and Challenges across Various Layers
Performance
Scalability
Fault-Resilience

HPC Advisory Council China Workshop, Oct '13
New Developments and Challenges

• Message Passing Interface (MPI)

• Partitioned Global Address Space (PGAS)
  – Hybrid Programming: MPI + PGAS

• Big Data
  – Hadoop
  – Memcached (Web 2.0)
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.
- In this presentation, we concentrate on MPI first, then on PGAS and Hybrid MPI+PGAS
MPI Overview and History

• Message Passing Library standardized by MPI Forum
  – C and Fortran

• Goal: portable, efficient and flexible standard for writing parallel applications

• Not IEEE or ISO standard, but widely considered “industry standard” for HPC application

• Evolution of MPI
  – MPI-1: 1994
  – MPI-2: 1996
  – Next plans for MPI 3.1, 3.2, ....
Major MPI Features

- Point-to-point Two-sided Communication
- Collective Communication
- One-sided Communication
- Job Startup
- Parallel I/O
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)

• 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
Non-blocking Collective Operations

• Enables overlap of computation with communication
• Removes synchronization effects of collective operations (exception of barrier)
• Non-blocking calls do not match blocking collective calls
  – MPI implementation may use different algorithms for blocking and non-blocking collectives
  – Blocking collectives: optimized for latency
  – Non-blocking collectives: optimized for overlap
• User must call collectives in same order on all ranks
• Progress rules are same as those for point-to-point
• Example new calls: MPI_Ibarrier, MPI_Iallreduce, ...
Improved One-sided (RMA) Model

- New RMA proposal has major improvements
- Easy to express irregular communication pattern
- Better overlap of communication & computation
- MPI-2: public and private windows
  - Synchronization of windows explicit
- MPI-2: works for non-cache coherent systems
- MPI-3: two types of windows
  - Unified and Separate
  - Unified window leverages hardware cache coherence
MPI Tools Interface

- Extended tools support in MPI-3, beyond the PMPI interface
- Provide standardized interface (MPIT) to access MPI internal information
  - Configuration and control information
    - Eager limit, buffer sizes, . . .
  - Performance information
    - Time spent in blocking, memory usage, . . .
  - Debugging information
    - Packet counters, thresholds, . . .
- External tools can build on top of this standard interface
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
Compiler-based: Unified Parallel C

• **UPC**: a parallel extension to the C standard

• **UPC Specifications and Standards:**
  – Introduction to UPC and Language Specification, 1999
  – UPC Language Specifications, v1.0, Feb 2001
  – UPC Language Specifications, v1.1.1, Sep 2004
  – UPC Language Specifications, v1.2, 2005

• **UPC Consortium**
  – Government Institutions: ARSC, IDA, LBNL, SNL, US DOE…
  – Commercial Institutions: HP, Cray, Intrepid Technology, IBM, …

• **Supported by several UPC compilers**
  – Vendor-based commercial UPC compilers: HP UPC, Cray UPC, SGI UPC
  – Open-source UPC compilers: Berkeley UPC, GCC UPC, Michigan Tech MuPC

• **Aims for**: high performance, coding efficiency, irregular applications, …
OpenSHMEM

- SHMEM implementations – Cray SHMEM, SGI SHMEM, Quadrics SHMEM, HP SHMEM, GSHMEM

- Subtle differences in API, across versions – example:

<table>
<thead>
<tr>
<th>SGI SHMEM</th>
<th>Quadrics SHMEM</th>
<th>Cray SHMEM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initialization</strong></td>
<td><strong>start_pes(0)</strong></td>
<td><strong>shmemb_init</strong></td>
</tr>
<tr>
<td><strong>Process ID</strong></td>
<td><strong>_my_pe</strong></td>
<td><strong>my_pe</strong></td>
</tr>
</tbody>
</table>

- Made applications codes non-portable

- OpenSHMEM is an effort to address this:

  “A new, open specification to consolidate the various extant SHMEM versions into a widely accepted standard.” – OpenSHMEM Specification v1.0

  by University of Houston and Oak Ridge National Lab

  SGI SHMEM is the baseline
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

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”

Challenges in Designing (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)
  - Extremely minimum memory footprint
- 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
- Support for GPGPUs and Accelerators
- Scalable Collective communication
  - Offload
  - Non-blocking
  - Topology-aware
  - Power-aware
- Fault-tolerance/resiliency
- QoS support for communication and I/O
- Support for Hybrid MPI+PGAS programming (MPI + OpenMP, MPI + UPC, MPI + OpenSHMEM, ...)

HPC Advisory Council China Workshop, Oct '13
MVAPICH2/MVAPICH2-X 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
  - MAPICH2-X (MPI + PGAS), Available since 2012
  - Used by more than 2,085 organizations (HPC Centers, Industry and Universities) in 71 countries
  - More than 193,000 downloads from OSU site directly
  - Empowering many TOP500 clusters
    - 6th ranked 462,462-core cluster (Stampede) at TACC
    - 19th ranked 125,980-core cluster (Pleiades) at NASA
    - 21st ranked 73,278-core cluster (Tsubame 2.0) 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
- Partner in the U.S. NSF-TACC Stampede System
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
  - Multicore-aware and Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware

- Support for GPGPUs

- Support for Intel MICs

- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, ...) with Unified Runtime
One-way Latency: MPI over IB

Small Message Latency

- 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
Bandwidth: MPI over IB

**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

**Bidirectional 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
MVAPICH2 Two-Sided Intra-Node Performance
(Shared memory and Kernel-based Zero-copy Support (LiMIC and CMA))

Latest MVAPICH2 2.0a
Intel Sandy-bridge

HPC Advisory Council China Workshop, Oct '13
eXtended Reliable Connection (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
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
  - Multicore-aware and Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware

- Support for GPGPUs
- Support for Intel MICs
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, ...) with Unified Runtime
Hardware Multicast-aware MPI_Bcast on Stampede

Small Messages (102,400 Cores)

Large Messages (102,400 Cores)

16 Byte Message

32 KByte Message

ConnectX-3-FDR (54 Gbps): 2.7 GHz Dual Octa-core (SandyBridge) Intel PCI Gen3 with Mellanox IB FDR switch
Shared-memory Aware Collectives

- MVAPICH2 Reduce/Allreduce with 4K cores on TACC Ranger (AMD Barcelona, SDR IB)

- MVAPICH2 Barrier with 1K Intel Westmere cores, QDR IB
Application benefits with Non-Blocking Collectives based on CX-2 Collective Offload

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

Can we design a highly scalable network topology detection service for IB?

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

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

CPMD Application Performance and Power Savings

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
  – Multicore-aware and Hardware-multicast-based
  – Offload and Non-blocking
  – Topology-aware
  – Power-aware

• Support for GPGPUs

• Support for Intel MICs

• Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, ...) with Unified Runtime
GPU-Direct

• Collaboration between Mellanox and NVIDIA to converge on one memory registration technique

• Both devices register a common host buffer
  – GPU copies data to this buffer, and the network adapter can directly read from this buffer (or vice-versa)

• *Note that GPU-Direct does not allow you to bypass host memory*

• *Latest GPUDirect RDMA achieves this*
Sample Code - Without MPI integration

- Naïve implementation with standard MPI and CUDA

At Sender:
```c
cudaMemcpy(sbuf, sdev, ...);
MPI_Send(sbuf, size, ...);
```

At Receiver:
```c
MPI_Recv(rbuf, size, ...);
cudaMemcpy(rdev, rbuf, ...);
```

• **High Productivity and Poor Performance**
Sample Code – User Optimized Code

- Pipelining at user level with non-blocking MPI and CUDA interfaces
- Code at Sender side (and repeated at Receiver side)

At Sender:
for (j = 0; j < pipeline_len; j++)
cudamemcpyAsync(sbuf + j * blk, sdev + j * blksz, . ..);
for (j = 0; j < pipeline_len; j++) {
  while (result != cudaSuccess) {
    result = cudaStreamQuery(...);
    if(j > 0) MPI_Test(...);
  }
  MPI_Isend(sbuf + j * block_sz, blksz . . .);
}
MPI_Waitall();

- User-level copying may not match with internal MPI design
- High Performance and Poor Productivity

HPC Advisory Council China Workshop, Oct '13
Sample Code – MVAPICH2-GPU

- MVAPICH2-GPU: standard MPI interfaces used
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:
  MPI_Send(s_device, size, ...);

At Receiver:
  MPI_Recv(r_device, size, ...);

• **High Performance and High Productivity**
MPI-Level Two-sided Communication

- 45% improvement compared with a naïve user-level implementation (Memcpy+Send), for 4MB messages
- 24% improvement compared with an advanced user-level implementation (MemcpyAsync+Isend), for 4MB messages

MVAPICH2 1.8, 1.9 and 2.0a 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
Applications-Level Evaluation

- **LBM-CUDA** (Courtesy: Carlos Rosale, TACC)
  - Lattice Boltzmann Method for multiphase flows with large density ratios
  - One process/GPU per node, 16 nodes

- **AWP-ODC** (Courtesy: Yifeng Cui, SDSC)
  - A seismic modeling code, Gordon Bell Prize finalist at SC 2010
  - 128x256x512 data grid per process, 8 nodes

- Oakley cluster at OSC: two hex-core Intel Westmere processors, two NVIDIA Tesla M2070, one Mellanox IB QDR MT26428 adapter, and 48 GB of main memory
GPU-Direct RDMA with CUDA 5

- Fastest possible communication between GPU and other PCI-E devices
- Network adapter can directly read data from GPU device memory
- Avoids copies through the host
- Allows for better asynchronous communication

MVAPICH2 with GDR support under development

MVAPICH2-GDR Alpha is available for users
Initial Design of OSU-MVAPICH2 with GPU-Direct-RDMA

- **Peer2Peer (P2P) bottlenecks on Sandy Bridge**
- **Design of MVAPICH2**
  - Hybrid design
  - Takes advantage of GPU-Direct-RDMA for writes to GPU
  - Uses host-based buffered design in current MVAPICH2 for reads
  - Works around the bottlenecks transparently

![Diagram of system architecture](attachment:image.png)

- **P2P write:** 5.2 GB/s
- **P2P read:** < 1.0 GB/s
Performance of MVAPICH2 with GPU-Direct-RDMA

GPU-GPU Internode MPI Latency

**Small Message Latency**
- MVAPICH2-1.9
- MVAPICH2-1.9-GDR

**Large Message Latency**
- MVAPICH2-1.9
- MVAPICH2-1.9-GDR

Based on MVAPICH2-1.9
- Intel Sandy Bridge (E5-2670) node with 16 cores
- NVIDIA Tesla K20c GPU, Mellanox ConnectX-3 FDR HCA
- CUDA 5.5, OFED 1.5.4.1 with GPU-Direct-RDMA Patch
Performance of MVAPICH2 with GPU-Direct-RDMA

GPU-GPU Internode MPI Uni-Directional Bandwidth

Small Message Bandwidth
- MVAPICH2-1.9
- MVAPICH2-1.9-GDR

Large Message Bandwidth
- MVAPICH2-1.9
- MVAPICH2-1.9-GDR

Based on MVAPICH2-1.9

Intel Sandy Bridge (E5-2670) node with 16 cores
NVIDIA Telsa K20c GPU, Mellanox ConnectX-3 FDR HCA
CUDA 5.5, OFED 1.5.4.1 with GPU-Direct-RDMA Patch
Performance of MVAPICH2 with GPU-Direct-RDMA

**GPU-GPU Internode MPI Bi-directional Bandwidth**

**Small Message Bi-Bandwidth**
- MVAPICH2-1.9
- MVAPICH2-1.9-GDR

**Large Message Bi-Bandwidth**
- MVAPICH2-1.9
- MVAPICH2-1.9-GDR

Based on MVAPICH2-1.9
Intel Sandy Bridge (E5-2670) node with 16 cores
NVIDIA Tesla K20c GPU, Mellanox ConnectX-3 FDR HCA
CUDA 5.5, OFED 1.5.4.1 with GPU-Direct-RDMA Patch
Getting Started with GDR Experimentation?

- Two modules are needed
  - GPUDirect RDMA (GDR) driver from Mellanox
  - MVAPICH2-GDR from OSU
- Send a note to hpc@mellanox.com
- You will get alpha versions of GDR driver and MVAPICH2-GDR (based on MVAPICH2 2.0a release)
- You can get started with this version
- MVAPICH2 team is working on multiple enhancements (collectives,(datatypes, one-sided) to exploit the advantages of GDR
- As GDR driver matures, successive versions of MVAPICH2-GDR with enhancements will be made available to the community
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
  - Multicore-aware and Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware
- Support for GPGPUs
- Support for Intel MICs
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, ...) with Unified Runtime
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
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
- Comparison with Intel’s MPI Library
MIC-RemoteHost (Closer to HCA) Point-to-Point Communication

**osu_latency (small)**

- Latency (usec) vs. Message Size (Bytes)
- MV2 vs. MV2-MIC

**osu_latency (large)**

- Latency (usec) vs. Message Size (Bytes)

**osu_bw**

- Bandwidth (MB/sec) vs. Message Size (Bytes)

**osu_bibw**

- Bandwidth (MB/sec) vs. Message Size (Bytes)
Inter-Node Symmetric Mode Collective Communication

8 Nodes with 8 Procs/Host and 8 Procs/MIC

osu_scatter (small)

Latency (usec)

0 20 40 60 80 100 120

Message Size (Bytes)

1 4 16 64 256 1K 4K

MV2

MV2-MIC

osu_allgather (small)

Latency (usec)

0 1000 2000 3000 4000 5000

Message Size (Bytes)

1 4 16 64 256 1K 4K

Better

Better

osu_scatter (large)

Latency (usec)

0 5000 10000 15000 20000 25000

Message Size (Bytes)

8K 32K 128K 512K

osu_allgather (large)

Latency (usec)

0 100000 200000 300000 400000 500000 600000 700000 800000

Message Size (Bytes)

8K 32K 128K 512K

Better

Better

HPC Advisory Council China Workshop, Oct '13
InterNode – Symmetric Mode - P3DFFT Performance

• To explore different threading levels for host and Xeon Phi
MVAPICH2 and Intel MPI – Intra-MIC

**Small Message Latency**

- **Intel MPI**
- **MVAPICH2-MIC**

**Large Message Latency**

- **Intel MPI**
- **MVAPICH2-MIC**

**Bandwidth**

- **Intel MPI**
- **MVAPICH2-MIC**

**Bi-directional Bandwidth**

- **Intel MPI**
- **MVAPICH2-MIC**

Better
MVAPICH2 and Intel MPI – Intranode – Host-MIC

Small Message Latency

Latency (usec)

Message Size (Bytes)

Large Message Latency

Latency (usec)

Message Size (Bytes)

Bandwidth

Bandwidth (MBps)

Message Size (Bytes)

Bi-directional Bandwidth

Bandwidth (MBps)

Message Size (Bytes)
MVAPICH2 and Intel MPI – Internode MIC-MIC

Small Message Latency

- **Latency (usec)**
  - Intel MPI
  - MVAPICH2-MIC

- **Message Size (Bytes)**
  - 0 2 8 32 128 512 2K

Large Message Latency

- **Latency (usec)**
  - Intel MPI
  - MVAPICH2-MIC

- **Message Size (Bytes)**
  - 4K 16K 64K 256K 1M 4M

Bandwidth

- **Bandwidth (MBps)**
  - Intel MPI
  - MVAPICH2-MIC

- **Message Size (Bytes)**
  - 1 16 256 4K 64K 1M

Bi-directional Bandwidth

- **Bandwidth (MBps)**
  - Intel MPI
  - MVAPICH2-MIC

- **Message Size (Bytes)**
  - 1 16 256 4K 64K 1M
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
  - Multicore-aware and Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware
- Support for GPGPUs
- Support for Intel MICs
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, ...) with Unified Runtime
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
Micro-Benchmark Performance (OpenSHMEM)

Atomic Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>OpenSHMEM-GASNet</th>
<th>OpenSHMEM-OSU</th>
</tr>
</thead>
<tbody>
<tr>
<td>fadd</td>
<td>6.0</td>
<td>3.6</td>
</tr>
<tr>
<td>finc</td>
<td>2.0</td>
<td>1.2</td>
</tr>
<tr>
<td>cswap</td>
<td>4.0</td>
<td>2.4</td>
</tr>
<tr>
<td>swap</td>
<td>2.0</td>
<td>1.2</td>
</tr>
<tr>
<td>add</td>
<td>4.0</td>
<td>2.4</td>
</tr>
<tr>
<td>inc</td>
<td>2.0</td>
<td>1.2</td>
</tr>
</tbody>
</table>

41% improvement for cswap
16% improvement for swap

Broadcast (256 processes)

<table>
<thead>
<tr>
<th>Size (bytes)</th>
<th>Time (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>64</td>
<td>100</td>
</tr>
<tr>
<td>256</td>
<td>1000</td>
</tr>
<tr>
<td>1K</td>
<td>10000</td>
</tr>
<tr>
<td>4K</td>
<td>100000</td>
</tr>
<tr>
<td>16K</td>
<td>1000000</td>
</tr>
<tr>
<td>64K</td>
<td>10000000</td>
</tr>
<tr>
<td>256K</td>
<td>100000000</td>
</tr>
<tr>
<td>1M</td>
<td>1000000000</td>
</tr>
</tbody>
</table>

Collect (256 processes)

<table>
<thead>
<tr>
<th>Size (bytes)</th>
<th>Time (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>64</td>
<td>100</td>
</tr>
<tr>
<td>256</td>
<td>1000</td>
</tr>
<tr>
<td>1K</td>
<td>10000</td>
</tr>
<tr>
<td>4K</td>
<td>100000</td>
</tr>
<tr>
<td>16K</td>
<td>1000000</td>
</tr>
<tr>
<td>64K</td>
<td>10000000</td>
</tr>
<tr>
<td>256K</td>
<td>100000000</td>
</tr>
<tr>
<td>1M</td>
<td>1000000000</td>
</tr>
</tbody>
</table>

Reduce (256 processes)

<table>
<thead>
<tr>
<th>Size (bytes)</th>
<th>Time (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>64</td>
<td>100</td>
</tr>
<tr>
<td>256</td>
<td>1000</td>
</tr>
<tr>
<td>1K</td>
<td>10000</td>
</tr>
<tr>
<td>4K</td>
<td>100000</td>
</tr>
<tr>
<td>16K</td>
<td>1000000</td>
</tr>
<tr>
<td>64K</td>
<td>10000000</td>
</tr>
<tr>
<td>256K</td>
<td>100000000</td>
</tr>
<tr>
<td>1M</td>
<td>1000000000</td>
</tr>
</tbody>
</table>

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
Hybrid MPI+UPC NAS-FT

- Modified NAS FT UPC all-to-all pattern using MPI_Alltoall
- Truly hybrid program
- For FT (Class C, 128 processes)
  - 34% improvement over UPC-GASNet
  - 30% improvement over UPC-OSU

J. Jose, M. Luo, S. Sur and D. K. Panda, Unifying UPC and MPI Runtimes: Experience with MVAPICH, Fourth Conference on Partitioned Global Address Space Programming Model (PGAS ’10), October 2010
MVAPICH2/MVPICH2-X – Plans for Exascale

- Performance and Memory scalability toward 500K-1M cores
  - Dynamically Connected Transport (DCT) service with Connect-IB
- Enhanced Optimization for GPU Support and Accelerators
  - Extending the GPGPU support (GPU-Direct RDMA) with CUDA 6.0 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 collectives
- 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
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) environment is gaining a lot of momentum
  – Memcached is also used for Web 2.0
Big Data Applications and Analytics

- **Big Data** has become one of the most important elements of business analytics
- Provides groundbreaking opportunities for enterprise information management and decision making
- The amount of data is exploding; companies are capturing and digitizing more information than ever
- The rate of information growth appears to be exceeding Moore’s Law
- **35 Zettabytes** of data will be generated and consumed by the end of this decade
- The Meaning of Big Data - **3 V’s**
  - **Big Volume**
  - **Big Velocity**
  - **Big Variety**

Can High-Performance Interconnects Benefit Big Data Processing?

• Beginning to draw interest from the enterprise domain
  – Oracle, IBM, Google are working along these directions
• Performance in the enterprise domain remains a concern
• Where do the bottlenecks lie?
• Can these bottlenecks be alleviated with new designs?
Can Big Data Processing Systems be Designed with High-Performance Networks and Protocols?

- Sockets not designed for high-performance
  - Stream semantics often mismatch for upper layers (Memcached, HBase, Hadoop)
  - Zero-copy not available for non-blocking sockets
Big Data Processing with Hadoop

- The open-source implementation of MapReduce programming model for Big Data Analytics
- Framework includes: MapReduce, HDFS, and HBase
- Underlying **Hadoop Distributed File System (HDFS)** used by MapReduce and HBase
- Model scales but high amount of communication during intermediate phases
Hadoop-RDMA Project

• High-Performance Design of Hadoop over RDMA-enabled Interconnects
  – High performance design with native InfiniBand support at the verbs-level for HDFS, MapReduce, and RPC components
  – Easily configurable for both native InfiniBand and the traditional sockets-based support (Ethernet and InfiniBand with IPoIB)

• Current release: 0.9.1
  – Based on Apache Hadoop 0.20.2
  – A version based on Apache Hadoop 1.2.1 will be released soon
  – Compliant with Apache Hadoop 0.20.2 APIs and applications
  – Tested with
    • Mellanox InfiniBand adapters (DDR, QDR and FDR)
    • Various multi-core platforms
    • Different file systems with disks and SSDs
  – [http://hadoop-rDMA.cse.ohio-state.edu](http://hadoop-rDMA.cse.ohio-state.edu)
Reducing Communication Time with HDFS-RDMA Design

- Cluster with HDD DataNodes
  - 30% improvement in communication time over IPoIB (QDR)
  - 56% improvement in communication time over 10GigE
- Similar improvements are obtained for SSD DataNodes

Evaluations using TestDFSIO for HDFS-RDMA Design

- Cluster with 8 HDD DataNodes, single disk per node
  - 24% improvement over IPoIB (QDR) for 20GB file size
- Cluster with 4 SSD DataNodes, single SSD per node
  - 61% improvement over IPoIB (QDR) for 20GB file size
Performance of Sort using Map-Reduce-RDMA with HDD

- With 4 HDD DataNodes for 20GB sort
  - 26% improvement over IPoIB (QDR)
  - 38% improvement over Hadoop-A-IB (QDR)

- With 8 HDD DataNodes for 40GB sort
  - 24% improvement over IPoIB (QDR)
  - 32% improvement over Hadoop-A-IB (QDR)

TeraSort Performance with Map-Reduce-RDMA

- 100GB TeraSort with 12 DataNodes and 200GB TeraSort in 24 DataNodes
  - 41% improvement over IPoIB (QDR)
  - 7% improvement over Hadoop-A-IB (QDR)
• 46% improvement in Adjacency List over IPoIB (32Gbps) for 30 GB data size
• 33% improvement in Sequence Count over IPoIB (32 Gbps) for 50 GB data size
Memcached Architecture

- **Distributed Caching Layer**
  - Allows to aggregate spare memory from multiple nodes
  - General purpose
- Typically used to cache database queries, results of API calls
- Scalable model, but typical usage very network intensive
- Native IB-verbs-level Design and evaluation with
  - Memcached Server: 1.4.9
  - Memcached Client: (libmemcached) 0.52
Memcached Get Latency (Small Message)

- Memcached Get latency
  - 4 bytes RC/UD – DDR: 6.82/7.55 us; QDR: 4.28/4.86 us
  - 2K bytes RC/UD – DDR: 12.31/12.78 us; QDR: 8.19/8.46 us
- Almost factor of *four* improvement over 10GE (TOE) for 2K bytes on the DDR cluster
Memcached Get TPS (4byte)

- Memcached Get transactions per second for 4 bytes
  - On IB QDR 1.4M/s (RC), 1.3 M/s (UD) for 8 clients
- Significant improvement with native IB QDR compared to SDP and IPoIB
Application Level Evaluation – Workload from Yahoo

- Real Application Workload
  - RC – 302 ms, UD – 318 ms, Hybrid – 314 ms for 1024 clients
- 12X times better than IPoIB for 8 clients
- Hybrid design achieves comparable performance to that of pure RC design


HPC Advisory Council China Workshop, Oct ’13
More Details on Big Data and Benchmarking will be covered.

- Keynote Talk, Accelerating Big Data Processing with Hadoop-RDMA
  - Forum of Big Data Computing (Oct 30th, 2:00-2:30pm)
- Keynote Talk, Challenges in Benchmarking and Evaluating Big Data Processing Middleware on Modern Clusters
  - Big Data Benchmarks, Performance Optimization and Emerging Hardware (BPOE) Workshop (Oct 31st, 2:05-2:45pm)
Concluding Remarks

- InfiniBand with RDMA feature is gaining momentum in HPC and Big Data systems with best performance and greater usage.
- As the HPC community moves to Exascale, new solutions are needed in the MPI+PGAS stacks for supporting GPUs and Accelerators.
- Demonstrated how such solutions can be designed with MVAPICH2/MVAPICH2-X and their performance benefits.
- New solutions are also needed to re-design software libraries for Big Data environments to take advantage of InfiniBand and RDMA.
- Such designs will allow application scientists and engineers to take advantage of upcoming exascale systems.
Funding Acknowledgments

Funding Support by

Office of Science
NSF
Mellanox Technologies
Cisco Systems
QLogic

intel
Linux Networx
NVIDIA
NetApp
Sun Microsystems

Equipment Support by

Mellanox Technologies
intel
AMD
Sun Microsystems
AHPR
advanced clustering technologies, inc.
Microway
NVIDIA
QLogic
## Personnel Acknowledgments

### Current Students
- N. Islam (Ph.D.)
- J. Jose (Ph.D.)
- M. Li (Ph.D.)
- S. Potluri (Ph.D.)
- R. Rajachandrasekhar (Ph.D.)
- M. Rahman (Ph.D.)
- R. Shir (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.)
- 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.)

### Current Senior Research Associate
- H. Subramoni

### Current Post-Docs
- X. Lu
- M. Luo
- K. Hamidouche

### Current Programmers
- M. Arnold
- J. Perkins

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

### Past Research Scientist
- S. Sur

### Past Programmers
- D. Bureddy
Pointers

http://nowlab.cse.ohio-state.edu

http://mvapich.cse.ohio-state.edu

http://hadoop-rdma.cse.ohio-state.edu

panda@cse.ohio-state.edu

http://www.cse.ohio-state.edu/~panda
Multiple Positions Available in My Group

- Looking for Bright and Enthusiastic Personnel to join as
  - Post-Doctoral Researchers (博士后)
  - PhD Students (博士研究生)
  - MPI Programmer/Software Engineer (MPI软件开发工程师)
  - Hadoop/Big Data Programmer/Software Engineer (大数据软件开发工程师)

- If interested, please contact me at this conference and/or send an e-mail to panda@cse.ohio-state.edu or luxi@cse.ohio-state.edu
Thanks, Q&A?

谢谢，请提问？