SMART NIC BoF

BoF Session at SC ‘21 (Nov. ‘21)

by

Hari Subramoni
The Ohio State University
E-mail: subramon@cse.ohio-state.edu
https://web.cse.ohio-state.edu/~subramoni.1/

Follow us on
https://twitter.com/mvapich
Outline

- Experience with SmartNICs
- Applications of SmartNICs
- Programming Models and Tools
- Architecture and Hardware
Proposed Offload Framework for SMART NICs

- Non-blocking collective operations are offloaded to a set of “worker processes”
- BlueField is set to separated host mode
- Worker processes are spawned to the ARM cores of BlueField
- Once the application calls a collective, host processes prepare a set of metadata and provide it to the Worker processes
- Using these metadata, worker processes can access host memory through RDMA
- Worker processes progress the collective on behalf of the host processes
- Once message exchanges are completed, worker processes notify the host processes about the completion of the non-blocking operation
Proposed Non-blocking Collective Designs

- Worker process performs RDMA Read to receive the data chunk from host main memory
- Once data is available in the ARM memory, worker process performs RDMA Write to the remote host memory
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
• 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 1.52 Million downloads from the OSU site directly
• Empowering many TOP500 clusters (Nov. ‘21 ranking)
  – 4th, 10,649,600-core (Sunway TaihuLight) at NSC, Wuxi, China
  – 13th, 448,448 cores (Frontera) at TACC
  – 26th, 288,288 cores (Lassen) at LLNL
  – 38th, 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 13th ranked TACC Frontera system
• Empowering Top500 systems for more than 16 years


Enhancing MVAPICH2 Software Architecture with DPU

High Performance Parallel Programming Models

- Message Passing Interface (MPI)
- PGAS (UPC, OpenSHMEM, CAF, UPC++)
- Hybrid --- MPI + X (MPI + PGAS + OpenMP/Cilk)

High Performance and Scalable Communication Runtime

Diverse APIs and Mechanisms

- Point-to-point Primitives
- Collectives Algorithms
- Job Startup
- Energy-Awareness
- Remote Memory Access
- I/O and File Systems
- Fault Tolerance
- Virtualization
- Active Messages
- Introspection & Analysis

Support for Modern Networking Technology
(InfiniBand, iWARP, RoCE, Omni-Path, Elastic Fabric Adapter)

Transport Protocols
- RC
- XRC
- UD
- DC

Modern Interconnect Features
- UMR
- ODP
- SR-IOV
- Multi Rail

Modern HCA Features
- Burst
- Poll
- Tag Match

Modern IB Features
- Multicast
- SHARP
- BlueField DPU

Network Based Computing Laboratory
SMART NIC BoF (SC’21)
Experimental Setup for Performance Evaluation

- HPC Advisory Council High-Performance Computing Center
  - Cluster has 32 compute-node with Broadwell series of Xeon dual-socket, 16-core processors operating at 2.60 GHz with 128 GB RAM
  - NVIDIA BlueField-2 adapters are equipped with 8 ARM cores operating at 2.0 GHz with 16 GB RAM
- Based on the MVAPICH2-DPU MPI library
- OSU Micro Benchmark for nonblocking Alltoall and P3DFFT Application
OSU Micro benchmark ialltoall

- **osu_ialltoall** benchmark metrics
  - **Pure communication time**
    - Latency $t$ is measured by calling MPI_ialltoall followed by MPI_Wait
  - **Total execution time**
    - Total $T = \text{MPI}_\text{ialltoall} + \text{synthetic compute} + \text{MPI}_\text{Wait}$
  - **Overlap**
    - Benchmark creates a synthetic computation block that takes $t$ microsecond to finish. Before starting compute, MPI_ialltoall is called and after that MPI_Wait. Overlap is calculated based on total execution time and compute time.
  - Part of the standard OSU Micro-Benchmark
Overlap of Communication and Computation with osu_ialltoall (32 nodes)

- **32 Nodes, 16 PPN**
- **32 Nodes, 32 PPN**

**Delivers peak overlap**

Overlap (osu_ialltoall)

- MVAPICH2
- MVAPICH2-DPU

<table>
<thead>
<tr>
<th>Message Size</th>
<th>Overlap (%)</th>
<th>MVAPICH2</th>
<th>MVAPICH2-DPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1K</td>
<td>100%</td>
<td>98%</td>
<td>100%</td>
</tr>
<tr>
<td>2K</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>4K</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>8K</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>16K</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>32K</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>64K</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>128K</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>256K</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>512K</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>
Pure Communication Latency with osu_ialltoall (32 nodes)

![Graph 1](image1)

Comm. Time, BF-2 (osu_ialltoall)

- MVAPICH2
- MVAPICH2-DPU

Message Size: 64K, 128K, 256K, 512K
Comm. Time (ms): 0, 100, 200, 300, 400, 500, 600

![Graph 2](image2)

Comm. Time, BF-2 (osu_ialltoall)

- MVAPICH2
- MVAPICH2-DPU

Message Size: 64K, 128K, 256K, 512K
Comm. Time (ms): 0, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600

32 Nodes, 16 PPN

32 Nodes, 32 PPN
Benefits in Total execution time (Compute + Communication)
P3DFFT Application Execution Time (16 nodes)

Benefits in application-level execution time

16 Nodes, 16 PPN

16 Nodes, 32 PPN
P3DFFT Application Execution Time (32 nodes)

Benefits in application-level execution time

32 Nodes, 16 PPN

32 Nodes, 32 PPN
Total Execution Time with osu_iallgather (16 nodes)

Total Execution Time, BF-2 (osu_iallgather)

- MVAPICH2
- MVAPICH2-DPU

Message Size:
- 256K
- 512K
- 1M
- 2M

Comm. Time (ms):

16 Nodes, 1 PPN

23% 39% 24% 41%

Total Execution Time, BF-2 (osu_iallgather)

- MVAPICH2
- MVAPICH2-DPU

Message Size:
- 256K
- 512K
- 1M
- 2M

Comm. Time (ms):

16 Nodes, 16 PPN

23% 43% 40% 48%
Total Execution Time with osu_ibcast (16 nodes)

- **Total Execution Time, BF-2 (osu_ibcast)**
  - MVAPICH2
  - MVAPICH2-DPU

**Message Size**
- 2M
- 4M
- 8M
- 16M

**Comm. Time (ms)**
- 0.00
- 5.00
- 10.00
- 15.00
- 20.00
- 25.00
- 30.00
- 35.00
- 40.00

**16 Nodes, 16 PPN**
- MVAPICH2: 59%
- MVAPICH2-DPU: 50%

**16 Nodes, 32 PPN**
- MVAPICH2: 58%
- MVAPICH2-DPU: 52%

**Total Execution Time with osu_Ibcast (16 nodes)**
Benefits of SMART NICs to DL Applications

- Training ShuffleNet on Tiny ImageNet Dataset
- Training ResNet-56 on SVHN Dataset
- Training ShuffleNet on Tiny ImageNet Dataset

Offload achieves 13.9% speedup on average on 1-16 nodes

Offload achieves 9.3% speedup on average on 16 nodes

Offload achieves 10.2% speedup on average on 16 nodes

- Everything or Based on the capabilities?
- Offloading compute (as things stand now) – bad idea!
- What is best suited to the capability of the DPU – orchestration of communication and I/O
  - Offload Data Augmentation (O-DA)
  - Offload Model Validation (O-MV)

Packet Processing Engines or General-Purpose Accelerator

- SMART NICs can be used as both PPEs or GPAs
  - Examples of PPEs
    - Hardware Tag Matching to perform rendezvous offload
    - Streaming reduction
  - Examples of GPAs
    - Enhanced Data Type Processing
    - Offloading complex collective communication patterns
Requirements for Next-Generation MPI Libraries

• Message Passing Interface (MPI) libraries are used for HPC and AI applications

• Requirements for a high-performance and scalable MPI library:
  – Low latency communication
  – High bandwidth communication
  – Minimum contention for host CPU resources to progress non-blocking collectives
  – High overlap of computation with communication

• CPU based non-blocking communication progress can lead to sub-par performance as the main application has less CPU resources for useful application-level computation
Can MPI Functions be Offloaded?

• The area of network offloading of MPI primitives is still nascent and cannot be used as a universal solution
• State-of-the-art BlueField DPUs bring more compute power into the network
• Can we exploit additional compute capabilities of modern BlueField DPUs into existing MPI middleware to extract
  – Peak pure communication performance
  – Overlap of communication and computation

For dense non-blocking collective communications?
Programming Models and Tools

- We have not used any specialized tools to utilize SMART NICs
- We see a clear need for a standardized interface
  - OpenSNAPI
- Currently SMART NICs appear as separate hosts to user level libraries
- Can next-gen SMART NICs be enhanced to provide direct access to host memory
  - Allow to initiate transfers on behalf of the host from host memory

![Diagram of SMART NICs and RDMA operations](image-url)