Performance of PGAS Models on Emerging Multi-/Many-core Architectures using MVAPICH2-X

Supercomputing’17 OSU Booth Talk

J. Hashmi, H. Subramoni and DK Panda
The Ohio State University
E-mail: {hashmi.29, Subramoni.1, panda.2}@osu.edu
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
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
    • Chapel
  – Libraries
    • OpenSHMEM
    • UPC++
    • Global Arrays
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
- Cons
  - Two different runtimes
  - Need great care while programming
  - Prone to deadlock if not careful
Overview of the MVAPICH2 Project

• High Performance open-source MPI Library for InfiniBand, Omni-Path, Ethernet/iWARP, and RDMA over Converged Ethernet (RoCE)
  – MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.0), Started in 2001, First version available in 2002
  – MVAPICH2-X (MPI + PGAS), Available since 2011
  – Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  – Support for Virtualization (MVAPICH2-Virt), Available since 2015
  – Support for Energy-Awareness (MVAPICH2-EA), Available since 2015
  – Support for InfiniBand Network Analysis and Monitoring (OSU INAM) since 2015
  – Used by more than 2,825 organizations in 85 countries
  – More than 432,000 (> 0.4 million) downloads from the OSU site directly
  – Empowering many TOP500 clusters (June ‘17 ranking)
    • 1st, 10,649,600-core (Sunway TaihuLight) at National Supercomputing Center in Wuxi, China
    • 15th, 241,108-core (Pleiades) at NASA
    • 20th, 462,462-core (Stampede) at TACC
    • 44th, 74,520-core (Tsubame 2.5) at Tokyo Institute of Technology
  – Available with software stacks of many vendors and Linux Distros (RedHat and SuSE)
    – http://mvapich.cse.ohio-state.edu

• Empowering Top500 systems for over a decade
  – System-X from Virginia Tech (3rd in Nov 2003, 2,200 processors, 12.25 TFlops) ->
  – Sunway TaihuLight (1st in Jun’17, 10M cores, 100 PFlops)
# MVAPICH2 Software Family

<table>
<thead>
<tr>
<th>High-Performance Parallel Programming Libraries</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MVAPICH2</strong></td>
<td>Support for InfiniBand, Omni-Path, Ethernet/iWARP, and RoCE</td>
</tr>
<tr>
<td><strong>MVAPICH2-X</strong></td>
<td>Advanced MPI features, OSU INAM, PGAS (OpenSHMEM, UPC, UPC++, and CAF), and MPI+PGAS programming models with unified communication runtime</td>
</tr>
<tr>
<td><strong>MVAPICH2-GDR</strong></td>
<td>Optimized MPI for clusters with NVIDIA GPUs</td>
</tr>
<tr>
<td><strong>MVAPICH2-Virt</strong></td>
<td>High-performance and scalable MPI for hypervisor and container based HPC cloud</td>
</tr>
<tr>
<td><strong>MVAPICH2-EA</strong></td>
<td>Energy aware and High-performance MPI</td>
</tr>
<tr>
<td><strong>MVAPICH2-MIC</strong></td>
<td>Optimized MPI for clusters with Intel KNC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Microbenchmarks</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OMB</strong></td>
<td>Microbenchmarks suite to evaluate MPI and PGAS (OpenSHMEM, UPC, and UPC++) libraries for CPUs and GPUs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tools</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OSU INAM</strong></td>
<td>Network monitoring, profiling, and analysis for clusters with MPI and scheduler integration</td>
</tr>
<tr>
<td><strong>OEMT</strong></td>
<td>Utility to measure the energy consumption of MPI applications</td>
</tr>
</tbody>
</table>
Performance of PGAS Models using MVAPICH2-X

- PGAS and Hybrid MPI+PGAS models support in MVAPICH2-X
- Optimizations of PGAS models for different architectures in MVAPICH2-X
  - Performance of Put and Get with OpenSHMEM, UPC, and UPC++
  - Comparison on KNL and Broadwell for OpenSHMEM point-to-point, collectives, and atomics Operations
  - Impact of AVX-512 Vectorization and MCDRAM on OpenSHMEM Application Kernels
  - Performance of UPC++ Application kernels on MVAPICH2-X communication runtime
MVAPICH2-X for Hybrid MPI + PGAS Applications

- **Current Model** – Separate Runtimes for OpenSHMEM/UPC/UPC++/CAF and MPI
  - Possible deadlock if both runtimes are not progressed
  - Consumes more network resource

- **Unified communication runtime for MPI, UPC, UPC++, OpenSHMEM, CAF**
  - Available with since 2012 (starting with MVAPICH2-X 1.9)
  - [http://mvapich.cse.ohio-state.edu](http://mvapich.cse.ohio-state.edu)
Application Level Performance with Graph500 and Sort

Graph500 Execution Time

- 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

Sort Execution Time

- Performance of Hybrid (MPI+OpenSHMEM) Sort Application
  - 4,096 processes, 4 TB Input Size
    - MPI – 2408 sec; 0.16 TB/min
    - Hybrid – 1172 sec; 0.36 TB/min
    - 51% improvement over MPI-design

J. Jose, S. Potluri, H. Subramoni, X. Lu, K. Hamidouche, K. Schulz, H. Sundar and D. Panda Designing Scalable Out-of-core Sorting with Hybrid MPI+PGAS Programming Models, PGAS’14

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
Accelerating MaTEx k-NN with Hybrid MPI and OpenSHMEM

- **MaTEx**: MPI-based Machine learning algorithm library
- **k-NN**: a popular supervised algorithm for classification
- **Hybrid designs:**
  - Overlapped Data Flow; One-sided Data Transfer; Circular-buffer Structure

- Benchmark: KDD Cup 2010 (8,407,752 records, 2 classes, k=5)
- For truncated KDD workload on 256 cores, reduce 27.6% execution time
- For full KDD workload on 512 cores, reduce 9.0% execution time

J. Lin, K. Hamidouche, J. Zhang, X. Lu, A. Vishnu, D. Panda. Accelerating k-NN Algorithm with Hybrid MPI and OpenSHMEM, OpenSHMEM 2015
Performance of PGAS Models using MVAPICH2-X

- PGAS and Hybrid MPI+PGAS models support in MVAPICH2-X
- Optimizations of PGAS models for different architectures in MVAPICH2-X
  - Performance of Put and Get with OpenSHMEM, UPC, and UPC++
  - Comparison on KNL and Broadwell for OpenSHMEM point-to-point, collectives, and atomics Operations
  - Impact of AVX-512 Vectorization and MCDRAM on OpenSHMEM Application Kernels
  - Performance of UPC++ Application kernels on MVAPICH2-X communication runtime
Intel Knights Landing (KNL) Processor Architecture

• Hardware Multi-threaded cores
  – Up to 72 cores (model 7290)

• All cores divided into 36 Tiles

• Each tile contains two core
  – 2 VPU per core
  – 1MB shared L2 cache

• 512-bit wide vector registers
  – AVX-512 extension

A single Tile of KNL
Intel Knights Landing (KNL) Processor - MCDRAM

- On-package Multi Channel DRAM (MCDRAM)
  - 450 GB/s of theoretical bandwidth (4x of DDR)
  - Configurable in Flat, Cache, and Hybrid modes
**Motivation**

- Optimizing HPC programming models and runtimes on emerging multi-/many-cores is of great research interest.
- Exploring benefits of the architectural features of modern architectures for PGAS models and applications.
- Characterizing and understanding:
  - the impact of vectorization on application kernels
  - MCDRAM vs. DDR performance
  - Exploiting hardware multi-threading
Performance of PGAS Models using MVAPICH2-X

- PGAS and Hybrid MPI+PGAS models support in MVAPICH2-X
- Optimizations of PGAS models for different architectures in MVAPICH2-X
  - Performance of Put and Get with OpenSHMEM, UPC, and UPC++
  - Comparison on KNL and Broadwell for OpenSHMEM point-to-point, collectives, and atomics Operations
  - Impact of AVX-512 Vectorization and MCDRAM on OpenSHMEM Application Kernels
  - Performance of UPC++ Application kernels on MVAPICH2-X communication runtime
Performance of PGAS Models on KNL using MVAPICH2-X

- Intra-node performance of one-sided Put/Get operations of PGAS libraries/languages using MVAPICH2-X communication conduit
- Near-native communication performance is observed on KNL
Performance of PGAS Models on KNL using MVAPICH2-X

- Inter-node performance of one-sided Put/Get operations using MVAPICH2-X communication conduit with InfiniBand HCA (MT4115)
- Native IB performance for all three PGAS models is observed.
Performance of PGAS Models using MVAPICH2-X

- PGAS and Hybrid MPI+PGAS models support in MVAPICH2-X
- Optimizations of PGAS models for different architectures in MVAPICH2-X
  - Performance of Put and Get with OpenSHMEM, UPC, and UPC++
  - Comparison on KNL and Broadwell for OpenSHMEM point-to-point, collectives, and atomics Operations
  - Impact of AVX-512 Vectorization and MCDRAM on OpenSHMEM Application Kernels
  - Performance of UPC++ Application kernels on MVAPICH2-X communication runtime
Microbenchmark Evaluations (Intra-node Put/Get)

- Broadwell shows about 3X better performance than KNL on large message
- Muti-threaded memcpy routines on KNL could offset the degradation caused by the slower core on basic Put/Get operations

- Inter-node small message latency is only 2X worse on KNL. While large message performance is almost similar on both KNL and Broadwell.
Microbenchmark Evaluations (Collectives)

Shmem_reduce on 128 processes
- 2 KNL nodes (64 ppn) and 8 Broadwell nodes (16 ppn).
- 4X degradation is observed on KNL using collective benchmarks.
- Basic point-to-point performance difference is reflected in collectives as well.

Shmem_broadcast on 128 processes
- ~4X
- ~2X
Microbenchmark Evaluations (Atomics)

- Using multiple nodes of KNL, atomic operations showed about 2.5X degradation on compare-swap, and Inc atomics.
- Fetch-and-add (32-bit) showed up to 4X degradation on KNL.
Performance of PGAS Models using MVAPICH2-X

- PGAS and Hybrid MPI+PGAS models support in MVAPICH2-X
- Optimizations of PGAS models for different architectures in MVAPICH2-X
  - Performance of Put and Get with OpenSHMEM, UPC, and UPC++
  - Comparison on KNL and Broadwell for OpenSHMEM point-to-point, collectives, and atomics Operations
  - Impact of AVX-512 Vectorization and MCDRAM on OpenSHMEM Application Kernels
  - Performance of UPC++ Application kernels on MVAPICH2-X communication runtime
• AVX-512 vectorized execution of BT kernel on KNL showed 30% improvement over default execution while EP kernel didn’t show any improvement.

• Broadwell showed 20% improvement over optimized KNL on BT and 4X improvement over all KNL executions on EP kernel (random number generation).
• Similar performance trends are observed on BT and MG kernels as well
• On SP kernel, MCDRAM based execution showed up to 20% improvement over default at 16 processes.
On heat diffusion based kernels AVX-512 vectorization showed better performance.

MCDRAM showed significant benefits on Heat-Image kernel for all process counts. Combined with AVX-512 vectorization, it showed up to 4X improved performance.
• Vectorization helps in matrix multiplication and vector operations.
• Due to heavily compute bound nature of these kernels, MCDRAM didn’t show any significant performance improvement.
Application Kernels Evaluation (contd.)

Scalable Integer Sort Kernel (ISx)

- Up to 3X improvement on un-optimized execution is observed on KNL
- Broadwell showed up to 2X better performance for core-by-core comparison
Node-by-node Evaluation using Application Kernels

Application Kernels on a single KNL vs. single Broadwell node

- A single node of KNL is evaluated against a single node of Broadwell using all the available physical cores
- HeatImage and ISx kernels showed better performance than Intel Xeon
Performance of PGAS Models using MVAPICH2-X

- PGAS and Hybrid MPI+PGAS models support in MVAPICH2-X
- Optimizations of PGAS models for different architectures in MVAPICH2-X
  - Performance of Put and Get with OpenSHMEM, UPC, and UPC++
  - Comparison on KNL and Broadwell for OpenSHMEM point-to-point, collectives, and atomics Operations
  - Impact of AVX-512 Vectorization and MCDRAM on OpenSHMEM Application Kernels
  - Performance of UPC++ Application kernels on MVAPICH2-X communication runtime
UPC++ Application Kernels Performance on KNL

- We used two application kernels to evaluate UPC++ model using MVAPICH2-X as communication runtime
- Sparse Matrix Vector Multiplication (SpMV)
- Adaptive Mesh Refinement (AMR) kernel
  - 2D-Heat conduction using Jacobi iterative
- We designed 2D-Heat kernel using pure UPC++ asynchronous primitives and provide MVAPICH2-X based communication support to achieve near-native performance.
- We observed near optimal speed-up for these kernels on two KNL nodes
• Implemented 2D Heat application kernels in UPC++
• SpMV and 2D Heat kernels using MVAPICH2-X showed good scalability on increasing number of processes of KNL

Performance Results Summary

- Put/Get and Atomics Performance
- Collectives Performance
- Core-by-core Application Performance
- Node-by-node Application Performance

(Closer to center is better)

- KNL (Default)
- KNL (AVX512)
- KNL (AVX512 + MCDRAM)
- Broadwell
Conclusion

• Comprehensive performance evaluation of MVAPICH2-X based OpenSHMEM, UPC, and UPC++ models over the KNL architecture

• Observed significant performance gains on application kernels when using AVX-512 vectorization
  – 2.5x performance benefits in terms of execution time

• MCDRAM benefits are not prominent on most of the application kernels
  – Lack of memory bound operations

• KNL showed up to 3X worse performance than Broadwell for core-by-core evaluation

• KNL showed better or on-par performance than Broadwell on Heat-Image and ISx kernels for Node-by-Node evaluation

• The runtime implementations need to take advantage of the concurrency of KNL cores
Thank You!

hashmi.29@osu.edu

Network-Based Computing Laboratory

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

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

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

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/