Performance of PGAS Models on KNL: A Comprehensive Study with MVAPICH2-X

IXPUG ’17 Presentation

J. Hashmi, M. Li, H. Subramoni and DK Panda

The Ohio State University

E-mail: {hashmi.29,li.2192,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
# MVAPICH2 Software Family

## High-Performance Parallel Programming Libraries

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVAPICH2</td>
<td>Support for InfiniBand, Omni-Path, Ethernet/iWARP, and RoCE</td>
</tr>
<tr>
<td>MVAPICH2-X</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>MVAPICH2-GDR</td>
<td>Optimized MPI for clusters with NVIDIA GPUs</td>
</tr>
<tr>
<td>MVAPICH2-Virt</td>
<td>High-performance and scalable MPI for hypervisor and container based HPC cloud</td>
</tr>
<tr>
<td>MVAPICH2-EA</td>
<td>Energy aware and High-performance MPI</td>
</tr>
<tr>
<td>MVAPICH2-MIC</td>
<td>Optimized MPI for clusters with Intel KNC</td>
</tr>
</tbody>
</table>

## Microbenchmarks

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

## Tools

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSU INAM</td>
<td>Network monitoring, profiling, and analysis for clusters with MPI and scheduler integration</td>
</tr>
<tr>
<td>OEMT</td>
<td>Utility to measure the energy consumption of MPI applications</td>
</tr>
</tbody>
</table>
MVAPICH2-X for Hybrid MPI + PGAS Applications

**High Performance Parallel Programming Models**

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI</td>
<td>Message Passing Interface</td>
</tr>
<tr>
<td>PGAS</td>
<td>(UPC, OpenSHMEM, CAF, UPC++)</td>
</tr>
<tr>
<td>Hybrid</td>
<td>(MPI + PGAS + OpenMP/Cilk)</td>
</tr>
</tbody>
</table>

**High Performance and Scalable Unified Communication Runtime**

<table>
<thead>
<tr>
<th>Diverse APIs and Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized Point-to-point Primitives</td>
</tr>
<tr>
<td>Remote Memory Access</td>
</tr>
<tr>
<td>Active Messages</td>
</tr>
<tr>
<td>Collectives Algorithms</td>
</tr>
<tr>
<td>Scalable Job Startup</td>
</tr>
<tr>
<td>Fault Tolerance</td>
</tr>
<tr>
<td>Introspection &amp; Analysis with OSU INAM</td>
</tr>
</tbody>
</table>

Support for Modern Networking Technologies

Support for Modern Multi-/Many-core Architectures

- 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

- 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

- 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

---


Performance of PGAS Models on KNL

- 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 on KNL

- 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

J. Hashmi, M. Li, H. Subramoni, D. Panda, Exploiting and Evaluating OpenSHMEM on KNL Architecture, Fourth Workshop on OpenSHMEM, Aug 2017
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
Microbenchmark Evaluations (Atomics)

OpenSHMEM atomics on 128 processes

- 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 on KNL

- 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
NAS Parallel Benchmark Evaluation

### NAS-BT (PDE solver), CLASS=B

<table>
<thead>
<tr>
<th>No. of processes</th>
<th>KNL (Default)</th>
<th>KNL (AVX-512)</th>
<th>KNL (AVX-512+MCDRAM)</th>
<th>Broadwell</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>60</td>
<td>40</td>
<td>30</td>
<td>40</td>
</tr>
<tr>
<td>36</td>
<td>50</td>
<td>40</td>
<td>30</td>
<td>40</td>
</tr>
<tr>
<td>64</td>
<td>40</td>
<td>30</td>
<td>30</td>
<td>40</td>
</tr>
<tr>
<td>100</td>
<td>30</td>
<td>20</td>
<td>20</td>
<td>40</td>
</tr>
</tbody>
</table>

### NAS-EP (RNG), CLASS=B

<table>
<thead>
<tr>
<th>No. of processes</th>
<th>KNL (Default)</th>
<th>KNL (AVX-512)</th>
<th>KNL (AVX-512+MCDRAM)</th>
<th>Broadwell</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>20</td>
<td>16</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>32</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>64</td>
<td>12</td>
<td>8</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>128</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

- 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 (Cont’d)

- 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

• A single node of KNL is evaluated against a single node of Broadwell using all the available physical cores
• HeatImage showed better performance than Intel Xeon
Performance of PGAS Models on KNL

- 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

- Developed and 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
Application Kernels Performance of UPC++ on MVAPICH2-X

Strong-scaling Performance of SpMV kernel (2Kx2K)
- SpMV and 2D Heat kernels using MVAPICH2-X shows good scalability on increasing number of processes of KNL

Strong-scaling Performance of 2D-Heat kernel (512x512)
Performance Results Summary

Put/Get and Atomics Performance

Collectives Performance

Core-by-core Application Performance

Node-by-node Application Performance

(KLC (Default)

KL (AVX512)

KL (AVX512 + MCDRAM)

Broadwell

(Closer to center is better)
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
Funding Acknowledgments

Funding Support by

Office of Science
NSF
Mellanox Technologies
Microsoft
Qlogic
Cray
 intel
Linux Network
NVIDIA
NetApp
Sun Microsystems
Cisco Systems

Equipment Support by

Mellanox Technologies
intel
AMD
Sun Microsystems
AFPR
advanced clustering technologies, inc.
Microwave
NVIDIA
Qlogic
Personnel Acknowledgments

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

Past Students
- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- S. Bhagvat (M.S.)
- A. Bhat (M.S.)
- D. Buntinas (Ph.D.)
- 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.)
- J. Jiang (M.S.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- K. Kulkarni (M.S.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)

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

Current Research Scientists
- X. Lu
- H. Subramoni

Current Research Specialist
- J. Smith
- M. Arnold

Past Research Scientist
- K. Hamidouche
- S. Sur

Past Programmers
- D. Bureddy
- J. Perkins

Current Post-doc
- A. Ruhela

Current Research Specialst
- J. Smith
- M. Arnold

Past Research Scientist
- K. Hamidouche
- S. Sur

Past Programmers
- D. Bureddy
- J. Perkins
Thank You!

panda@cse.ohio-state.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/