Supporting PGAS Models (UPC and OpenSHMEM) on MIC Clusters

Presentation at IXPUG Meeting, July 2014

by

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

Khaled Hamidouche
The Ohio State University
E-mail: hamidouc@cse.ohio-state.edu
http://www.cse.ohio-state.edu/~hamidouc
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.
- Additionally, OpenMP can be used to parallelize computation within the node
- Each model has strengths and drawbacks - suit different problems or applications
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
    • Global Arrays
SHMEM

- SHMEM: Symmetric Hierarchical MEMory library
- One-sided communications library – had been around for a while
- Similar to MPI, processes are called PEs, data movement is explicit through library calls
- Provides globally addressable memory using symmetric memory objects (more in later slides)
- Library routines for
  - Symmetric object creation and management
  - One-sided data movement
  - Atomics
  - Collectives
  - Synchronization
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>Initialization</td>
<td>_start_pes(0)</td>
<td>_shmem_init</td>
</tr>
<tr>
<td>Process ID</td>
<td>_my_pe</td>
<td>my_pe</td>
</tr>
<tr>
<td></td>
<td></td>
<td>shmem_my_pe</td>
</tr>
</tbody>
</table>

- Made application 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
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, June 2005
- **UPC Consortium**
  - Academic Institutions: GWU, MTU, UCB, U. Florida, U. Houston, U. Maryland...
  - 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, ...
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”

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
  - MVAPICH2-X (MPI + PGAS), Available since 2012
  - Support for GPGPUs and MIC
  - Used by more than 2,150 organizations (HPC Centers, Industry and Universities) in 72 countries
  - More than 218,000 downloads from OSU site directly
  - Empowering many TOP500 clusters
    - 7th ranked 519,640-core cluster (Stampede) at TACC
    - 11th ranked 74,358-core cluster (Tsubame 2.5) at Tokyo Institute of Technology
    - 16th ranked 96,192-core cluster (Pleiades) at NASA 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
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

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 (with initial support for UPC 1.3)
- Scalable Inter-node and intra-node communication – point-to-point and collectives
Outline

• OpenSHMEM and UPC for Host
• Hybrid MPI + PGAS (OpenSHMEM and UPC) for Host
• OpenSHMEM for MIC
• UPC for MIC
OpenSHMEM Design in MVAPICH2-X

- OpenSHMEM Stack based on OpenSHMEM Reference Implementation
- OpenSHMEM Communication over MVAPICH2-X Runtime
  - Uses active messages, atomic and one-sided operations and remote registration cache

OpenSHMEM Collective Communication Performance

Reduce (1,024 processes)

- MV2X-SHMEM
- Scalable-SHMEM
- OMPI-SHMEM

Broadcast (1,024 processes)

- MV2X-SHMEM
- Scalable-SHMEM
- OMPI-SHMEM

Collect (1,024 processes)

- MV2X-SHMEM
- Scalable-SHMEM
- OMPI-SHMEM

Barrier

- MV2X-SHMEM
- Scalable-SHMEM
- OMPI-SHMEM

IXPUG14-OSU-PGAS
OpenSHMEM Application Evaluation

Heat Image

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

MVAPICH2-X Support for Berkeley UPC Runtime

- GASNet (Global-Address Space Networking) is a language-independent, low-level networking layer that provides support for PGAS language

- Support multiple networks through different conduit: MVAPICH2-X Conduit is available in MVAPICH2-X release, which support UPC/OpenMP/MPI on InfiniBand
**UPC Collectives Performance**

**Broadcast (2048 processes)**

- Time (us) vs Message Size
- **UPC-GASNet**
- **UPC-OSU**

**Scatter (2048 processes)**

- Time (ms) vs Message Size
- **UPC-GASNet**
- **UPC-OSU**

**Gather (2048 processes)**

- Time (ms) vs Message Size
- **UPC-GASNet**
- **UPC-OSU**

**Exchange (2048 processes)**

- Time (ms) vs Message Size
- **UPC-GASNet**
- **UPC-OSU**

---

*J. Jose, K. Hamidouche, J. Zhang, A. Venkatesh, and D. K. Panda, Optimizing Collective Communication in UPC (HiPS’14, in association with IPDPS’14)*
Outline

- OpenSHMEM and UPC for Host
- Hybrid MPI + PGAS (OpenSHMEM and UPC) for Host
- OpenSHMEM for MIC
- UPC for MIC
Unified Runtime for Hybrid MPI + OpenSHMEM Applications

- Optimal network resource usage
- No deadlock because of single runtime
- Better performance

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+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 (MV2-X)

Outline

• OpenSHMEM and UPC for Host
• Hybrid MPI + PGAS (OpenSHMEM and UPC) for Host
• OpenSHMEM for MIC
• UPC for MIC
HPC Applications on MIC Clusters

- Flexibility in launching HPC jobs on Xeon Phi Clusters

Multi-core Centric

Host-only

Offload (/reverse Offload)

Symmetric

Coprocessor-only

Xeon

OpenSHMEM Program

OpenSHMEM Program

OpenSHMEM Program

Xeon Phi

Offloaded Computation

OpenSHMEM Program

OpenSHMEM Program
Data Movement on Intel Xeon Phi Clusters

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 on remote socket
and more . . .

- Critical for runtimes to optimize data movement, hiding the complexity
Need for Non-Uniform Memory Allocation in OpenSHMEM

- MIC cores have limited memory per core
- OpenSHMEM relies on symmetric memory, allocated using shmalloc()
- shmalloc() allocates same amount of memory on all PEs
- For applications running in symmetric mode, this limits the total heap size
- Similar issues for applications (even host-only) with memory load imbalance (Graph500, Out-of-Core Sort, etc.)
- **How to allocate different amounts of memory on host and MIC cores, and still be able to communicate?**
OpenSHMEM Design for MIC Clusters

- Non-Uniform Memory Allocation:
  - Team-based Memory Allocation
    (Proposed Extensions)
    
    ```c
    void shmem_team_create(shmem_team_t *team, int *ranks, int size, shmem_team_t *newteam);
    void shmem_team_destroy(shmem_team_t *team);
    
    void shmem_team_split(shmem_team_t *team, int color, int key, shmem_team_t *newteam);
    
    int shmem_team_rank(shmem_team_t *team);
    int shmem_team_size(shmem_team_t *team);
    
    void *shmalloc_team (shmem_team_t *team, size_t size);
    void shfree_team(shmem_team_t *team, void *addr);
    ```
  
  - Address Structure for non-uniform memory allocations
Proxy-based designs for OpenSHMEM

OpenSHMEM Put using Active Proxy

OpenSHMEM Get using Active Proxy

- Current generation architectures impose limitations on read bandwidth when HCA reads from MIC memory
  - Impacts both put and get operation performance

- Solution: Pipelined data transfer by proxy running on host using IB and SCIF channels

- Improves latency and bandwidth!
Proxy-based designs alleviate hardware limitations

- Put Latency of 4M message: Default: 3911us, Optimized: 838us
- Get Latency of 4M message: Default: 3889us, Optimized: 837us
Optimized designs for OpenSHMEM collective operations

- Broadcast Latency of 256KB message at 1,024 processes:
  - Default: 5955us, Optimized: 2268us

- Reduce Latency of 256KB message at 1,024 processes:
  - Default: 6581us, Optimized: 3294us
Performance Evaluations using Graph500

**Native Mode (8 procs/MIC)**

- Graph500 Execution Time (Native Mode):
  - At 512 processes, Default: 5.17s, Optimized: 4.96s
  - Performance Improvement from MIC-aware collectives design

**Symmetric Mode (16 Host+16MIC)**

- Graph500 Execution Time (Symmetric Mode):
  - At 1,024 processes, Default: 15.91s, Optimized: 12.41s
  - Performance Improvement from MIC-aware collectives and proxy-based designs
Graph500 Evaluations with Extensions

• Redesigned Graph500 using MIC to overlap computation/communication
  – Data Transfer to MIC memory; MIC cores pre-processes received data
  – Host processes traverses vertices, and sends out new vertices

• Graph500 Execution time at 1,024 processes:
  – Host-Only: .33s, Host+MIC with Extensions: .26s

• Magnitudes of improvement compared to default symmetric mode
  – Default Symmetric Mode: 12.1s, Host+MIC Extensions: 0.16s

Outline

- OpenSHMEM and UPC for Host
- Hybrid MPI + PGAS (OpenSHMEM and UPC) for Host
- OpenSHMEM for MIC
- UPC for MIC
HPC Applications on MIC Clusters

- Flexibility in launching HPC jobs on Xeon Phi Clusters

- Multi-core Centric
  - Host-only
  - Offload (reverse Offload)
  - Symmetric
  - Coprocessor-only

- Many-core Centric

- Xeon
  - UPC Program

- Xeon Phi
  - Offloaded Computation
  - UPC Program
  - UPC Program
  - UPC Program
UPC on MIC clusters

• Process Model
  – Communication via shared memory – single copy/double copy
  – One connection per process
  – Drawbacks: Communication Overheads, Network Connections

• Thread Model
  – Direct access to shared array: same memory space
  – No extra copy
  – No shared memory mapping entries
  – Drawback: sharing of network connections
  – Our work on multi-endpoint addresses this issue
UPC Runtime on MIC: Remote Memory Access between Host and MIC

- **Original connection mode**: Each UPC thread pins down the global memory region that has affinity with this UPC thread; Establish all-to-all connections between every pair of UPC threads between MIC and host
- **Leader-to-all connection mode**: Only one UPC thread pinned down the whole global memory space on MIC; all UPC threads on the host will get access with the leader UPC thread
- **Connections**: $N_{MIC} \times N_{HOST} \rightarrow N_{MIC} + N_{HOST}$
UPC Runtime on MIC: Remote Memory Access between Host and MIC

- Process-based Runtime: The whole global memory region need to be mapped as shared memory region, such as PSHM
- Thread-based Runtime: As the threads share the same memory space, the leader can access other global memory region on MIC without mapping to shared memory
Application Benchmark Evaluation for Native Mode

- Host and MIC are occupied with 16 and 60 UPC threads, respectively
- MIC performs 80%, 67%, and 54% as 16-CPU host for MG, EP, and FT, respectively
- For communication-intensive benchmarks CG and IS, MIC spends 7x and 3x execution times
- Default applications without modification – no representation of optimal performance

Concluding Remarks

- HPC Systems are evolving to support growing needs of exascale applications
- Hybrid programming (MPI + PGAS) is a practical way to program exascale systems
- Presented designs to demonstrate performance potential of PGAS and hybrid MPI+PGAS models
- MIC support for OpenSHMEM and UPC will be available in future MVAPICH2-X releases
Funding Acknowledgments

Funding Support by


Equipment Support by

Personnel Acknowledgments

Current Students
- S. Chakraborthy (Ph.D.)
- N. Islam (Ph.D.)
- J. Jose (Ph.D.)
- M. Li (Ph.D.)
- R. Rajachandrasekhar (Ph.D.)
- M. Rahman (Ph.D.)
- D. Shankar (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.)

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

Current Senior Research Associates
- H. Subramoni
- X. Lu

Current Post-Docs
- K. Hamidouche
- J. Lin

Current Programmers
- M. Arnold
- J. Perkins

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

Past Research Scientist
- S. Sur

Past Programmers
- D. Bureddy
Web Pointers

http://www.cse.ohio-state.edu/~panda
http://www.cse.ohio-state.edu/~hamidouc
http://nowlab.cse.ohio-state.edu

MVAPICH Web Page
http://mvapich.cse.ohio-state.edu

panda@cse.ohio-state.edu
hamidouc@cse.ohio-state.edu