MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

Bringing NVIDIA GPUs to the PGAS/OpenSHMEM World: Challenges and Solutions

GPU Technology Conference (GTC 2016)

by

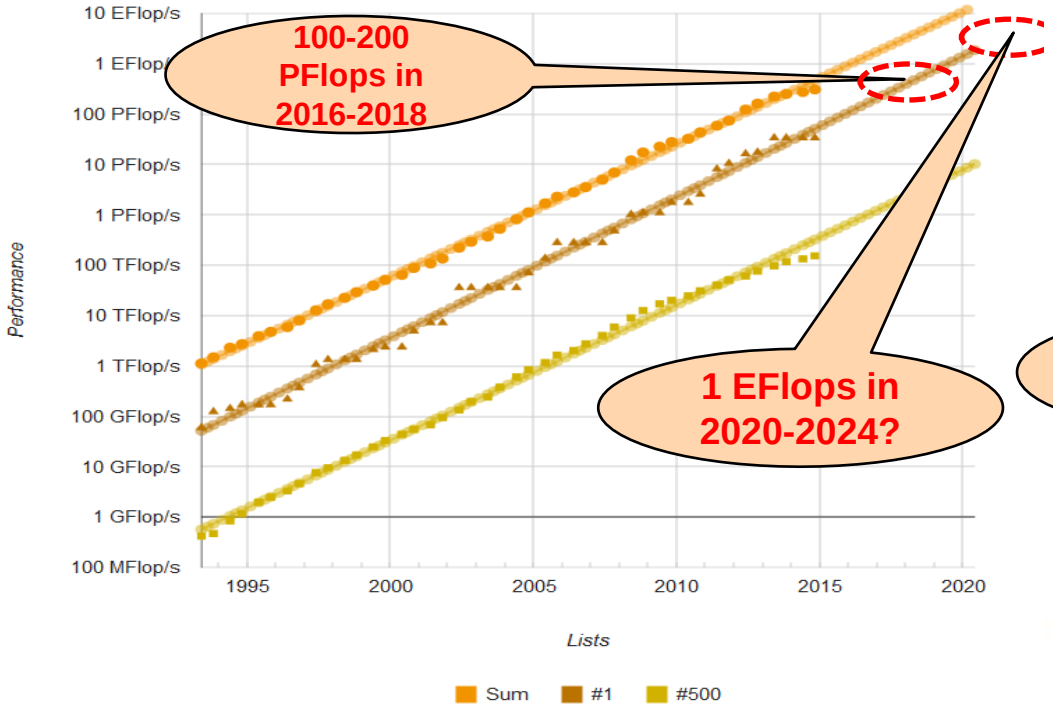
Dhabaleswar K. (DK) Panda

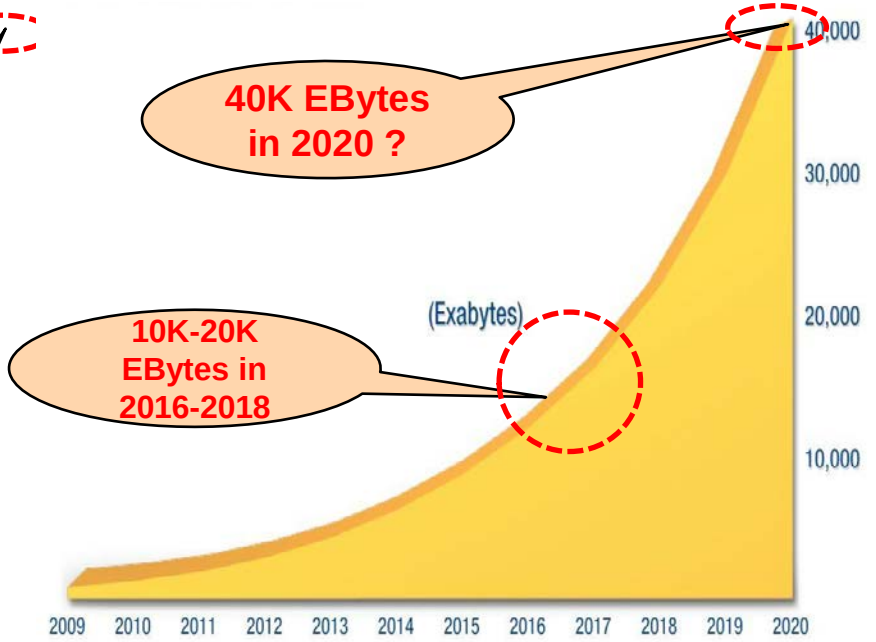
The Ohio State University

E-mail: panda@cse.ohio-state.edu

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

High-End Computing (HEC): ExaFlop & ExaByte

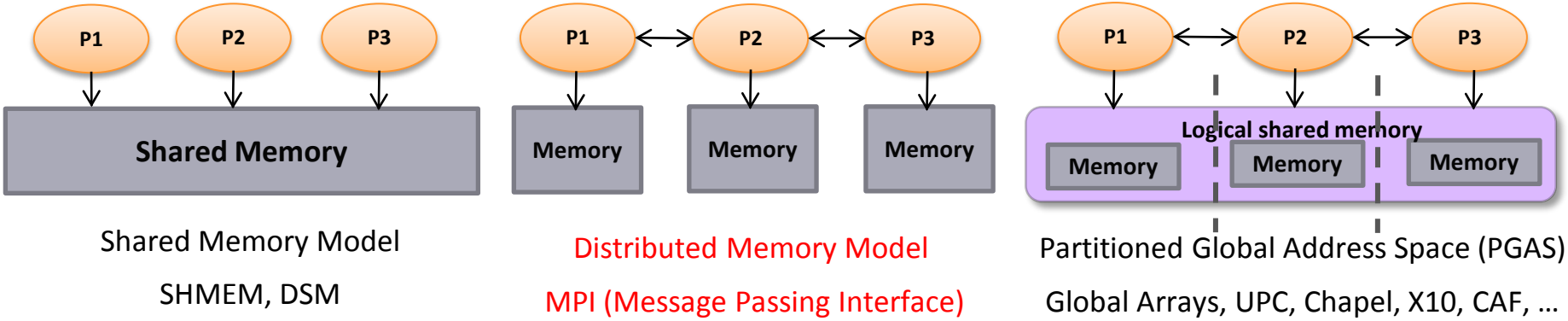




ExaFlop & HPC

ExaByte & BigData

Parallel Programming Models Overview



Shared Memory Model

SHMEM, DSM

Distributed Memory Model

MPI (Message Passing Interface)

Partitioned Global Address Space (PGAS)

Global Arrays, UPC, Chapel, X10, CAF, …

- 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

Outline

- Overview of PGAS models (UPC and OpenSHMEM)
- Limitations in PGAS models for GPU computing
- Proposed Designs and Alternatives
- Performance Evaluation
- Conclusions

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

OpenSHMEM

- SHMEM implementations – Cray SHMEM, SGI SHMEM, Quadrics SHMEM, HP SHMEM, GSHMEM
- Subtle differences in API, across versions – example:

	SGI SHMEM	Quadrics SHMEM	Cray SHMEM
Initialization	*start_pes(0)*	*shmem_init*	*start_pes*
Process ID	*_my_pe*	*my_pe*	*shmem_my_pe*

- 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

OpenSHMEM Memory Model

- Defines symmetric data objects that are globally addressable

  - Allocated using a collective *shmalloc* routine

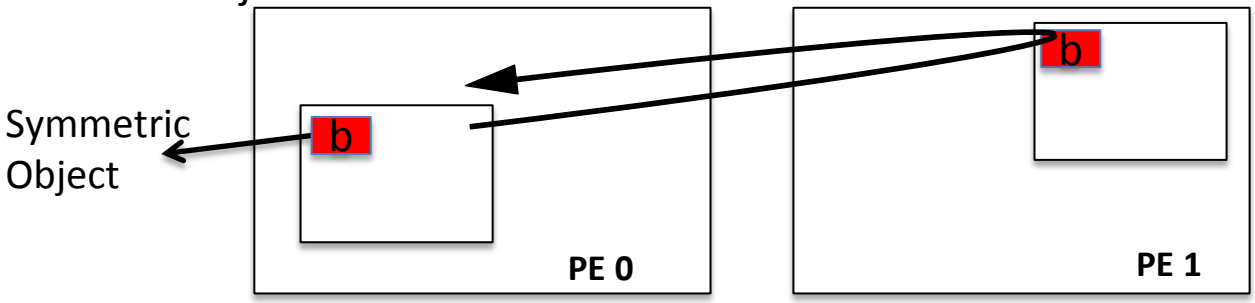
  - Same type, size and offset address at all processes/processing elements (PEs)

  - Address of a remote object can be calculated based on info of local object

```
int main (int c, char *v[]) {
   int *b;

   start_pes();
   b =  (int *) shmalloc (sizeof(int));

   shmem_int_get (b, b, 1 , 1);
}                          (dst, src, count, pe)
```



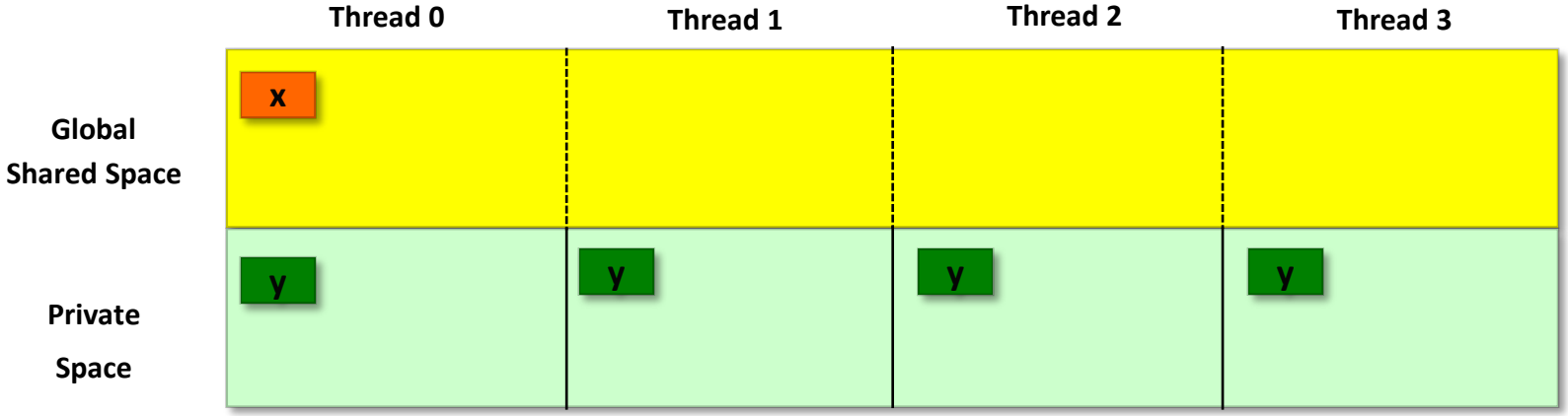

```
int main (int c, char *v[]) {
   int *b;

   start_pes();
   b =  (int *) shmalloc (sizeof(int));
}
```

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 Language Specifications, v1.3, In Progress - Draft Available
- 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, …

UPC: Memory Model



- Global Shared Space: can be accessed by all the threads
- Private Space: holds all the normal variables; can only be accessed by the local thread
- Examples:

```
shared int x;    //shared variable; allocated with affinity to Thread 0
int main() {
    int y;          //private variable
}
```

MPI+PGAS for Exascale Architectures and Applications

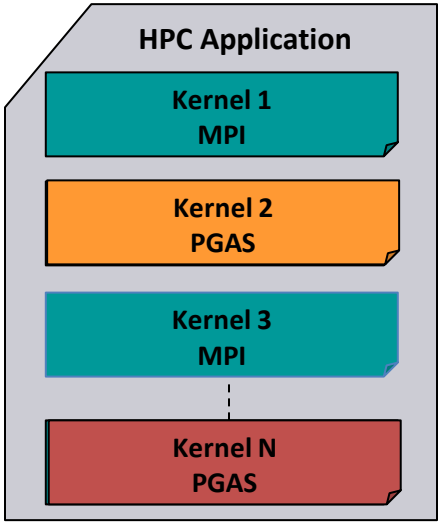
- Gaining attention in efforts towards Exascale computing
- Hierarchical architectures with multiple address spaces
- (MPI + PGAS) Model

  - 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
- 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"

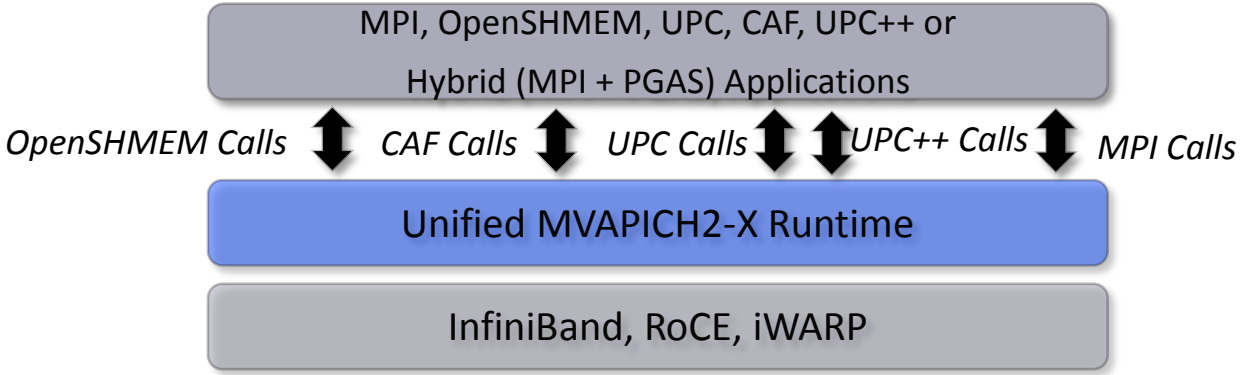


** The International Exascale Software Roadmap, Dongarra, J., Beckman, P. et al., Volume 25, Number 1, 2011, International Journal of High Performance Computer Applications, ISSN 1094-3420*

Overview of the MVAPICH2 Project

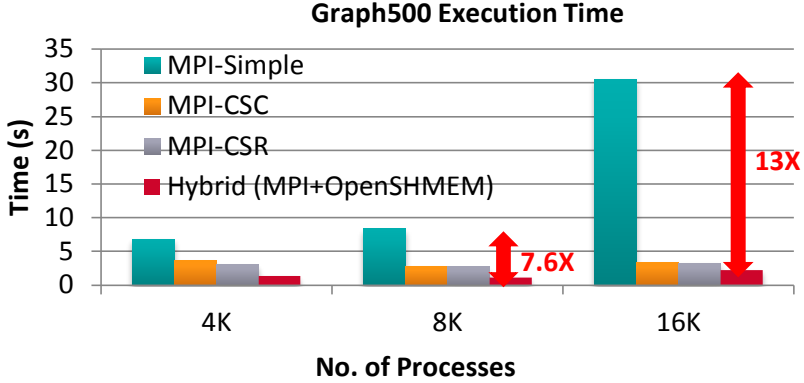
- High Performance open-source MPI Library for InfiniBand, 10-40Gig/iWARP, and RDMA over Converged Enhanced Ethernet (RoCE)
  - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.0), Available since 2002
  - **MVAPICH2-X (MPI + PGAS), Available since 2012**
  - Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  - Support for Virtualization (MVAPICH2-Virt), Available since 2015
  - Support for Energy-Awareness (MVAPICH2-EA), Available since 2015
  - **Used by more than 2,550 organizations in 79 countries**
  - **More than 360,000 (> 0.36 million) downloads from the OSU site directly**
  - Empowering many TOP500 clusters (Nov '15 ranking)
    - 10th ranked 519,640-core cluster (Stampede) at TACC
    - 13th ranked 185,344-core cluster (Pleiades) at NASA
    - 25th ranked 76,032-core cluster (Tsubame 2.5) at Tokyo Institute of Technology and many others
  - 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) ->
  - Stampede at TACC (10th in Nov'15, 519,640 cores, 5.168 Plops)

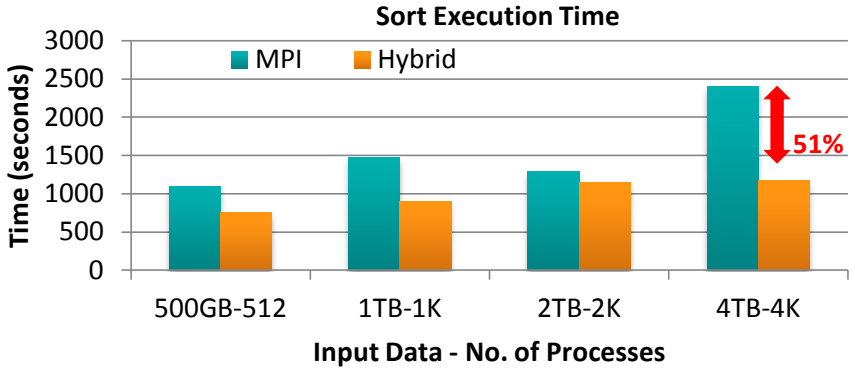
MVAPICH2-X for Advanced MPI and Hybrid MPI + PGAS Applications



- Unified communication runtime for MPI, UPC, OpenSHMEM, CAF available with MVAPICH2-X 1.9 (2012) onwards!
- UPC++ support available in the latest MVAPICH2-X 2.2RC1
- Feature Highlights
  - Supports MPI(+OpenMP), OpenSHMEM, UPC, CAF, UPC++, MPI(+OpenMP) + OpenSHMEM, MPI(+OpenMP) + UPC + CAF
  - MPI-3 compliant, OpenSHMEM v1.0 standard compliant, UPC v1.2 standard compliant (with initial support for UPC 1.3), CAF 2008 standard (OpenUH), UPC++ 1.0
  - Scalable Inter-node and intra-node communication – point-to-point and collectives

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

J. Jose, K. Kandalla, S. Potluri, J. Zhang and D. K. Panda, Optimizing Collective Communication in OpenSHMEM, Int'l Conference on Partitioned Global Address Space Programming Models (PGAS '13), October 2013.

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

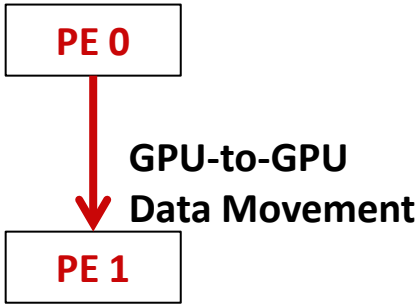
J. Jose, K. Kandalla, M. Luo and D. K. Panda, Supporting Hybrid MPI and OpenSHMEM over InfiniBand: Design and Performance Evaluation, Int'l Conference on Parallel Processing (ICPP '12), September 2012

Outline

- Overview of PGAS models (UPC and OpenSHMEM)
- Limitations in PGAS models for GPU computing
- Proposed Designs and Alternatives
- Performance Evaluation
- Conclusions

Limitations of OpenSHMEM for GPU Computing

- OpenSHMEM memory model does not support disjoint memory address spaces - case with GPU clusters



Existing OpenSHMEM Model with CUDA

```
PE 0
host_buf = shmalloc (…)
cudaMemcpy (host_buf, dev_buf, . . . )
shmem_putmem (host_buf, host_buf, size, pe)
shmem_barrier (…)

PE 1
host_buf = shmalloc (…)
shmem_barrier (…) . )
cudaMemcpy (dev_buf, host_buf, size, . . . )
```

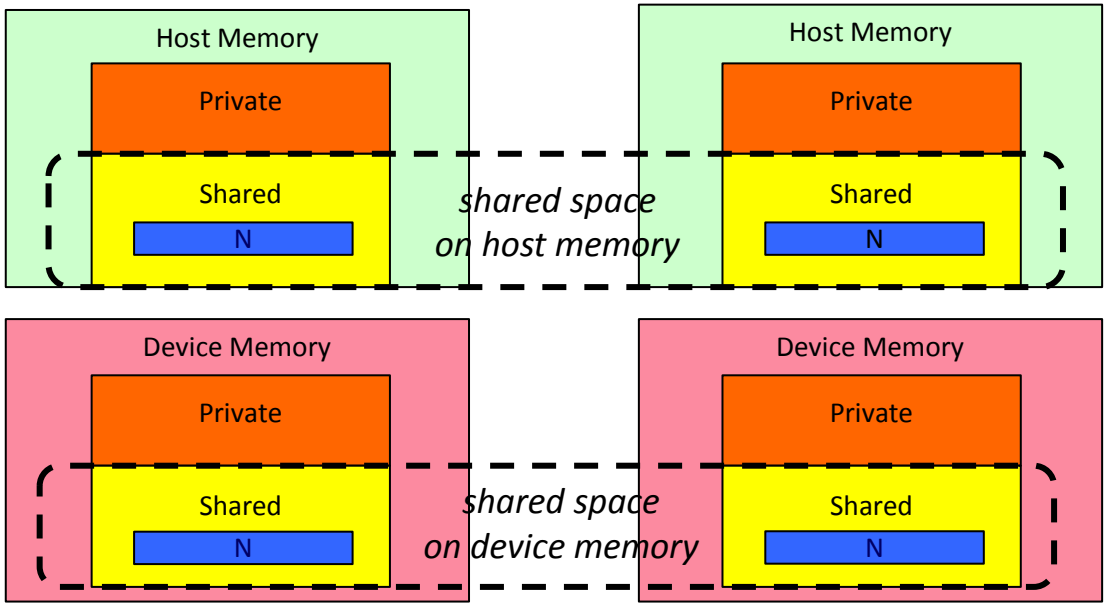
- Copies severely limit the performance
- Synchronization negates the benefits of one-sided communication
- Similar issues with UPC

Outline

- Overview of PGAS models (UPC and OpenSHMEM)
- Limitations in PGAS models for GPU computing
- Proposed Designs and Alternatives
- Performance Evaluation
- Conclusions

Global Address Space with Host and Device Memory

- Extended APIs:
- heap_on_device/heap_on_host
- a way to indicate location of heap
- *host_buf = shmalloc (sizeof(int), 0);*
- *dev_buf = shmalloc (sizeof(int), 1);*



CUDA-Aware OpenSHMEM
Same design for UPC

PE 0

```
dev_buf = shmalloc (size, 1);
shmem_putmem (dev_buf, dev_buf, size, pe)
```

PE 1

```
dev_buf = shmalloc (size, 1);
```

S. Potluri, D. Bureddy, H. Wang, H. Subramoni and D. K. Panda, Extending OpenSHMEM for GPU Computing, IPDPS'13

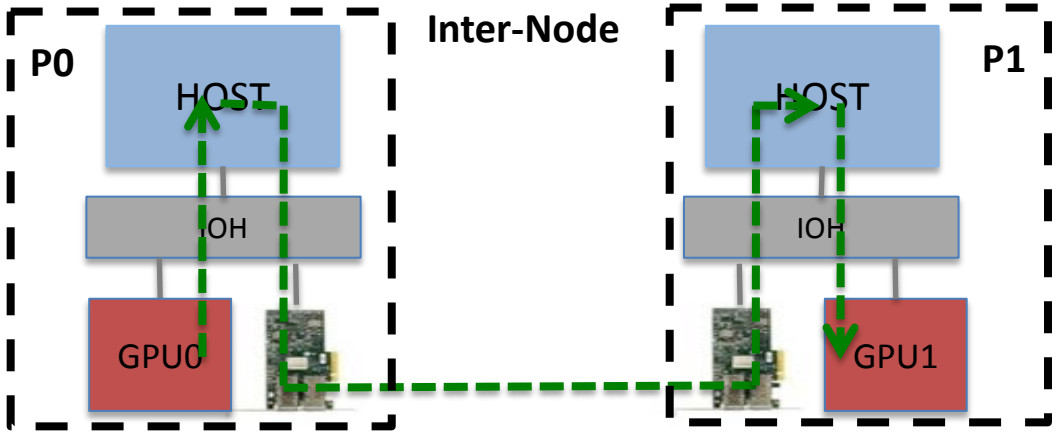
CUDA-aware OpenSHMEM and UPC runtimes

- After device memory becomes part of the global shared space:
  - Accessible through standard UPC/OpenSHMEM communication APIs
  - Data movement transparently handled by the runtime
  - Preserves one-sided semantics at the application level
- Efficient designs to handle communication
  - Inter-node transfers use host-staged transfers with pipelining
  - Intra-node transfers use CUDA IPC
  - Possibility to take advantage of GPUDirect RDMA (GDR)

- Goal: Enabling High performance one-sided communications semantics with GPU devices

Inter-Node Communication

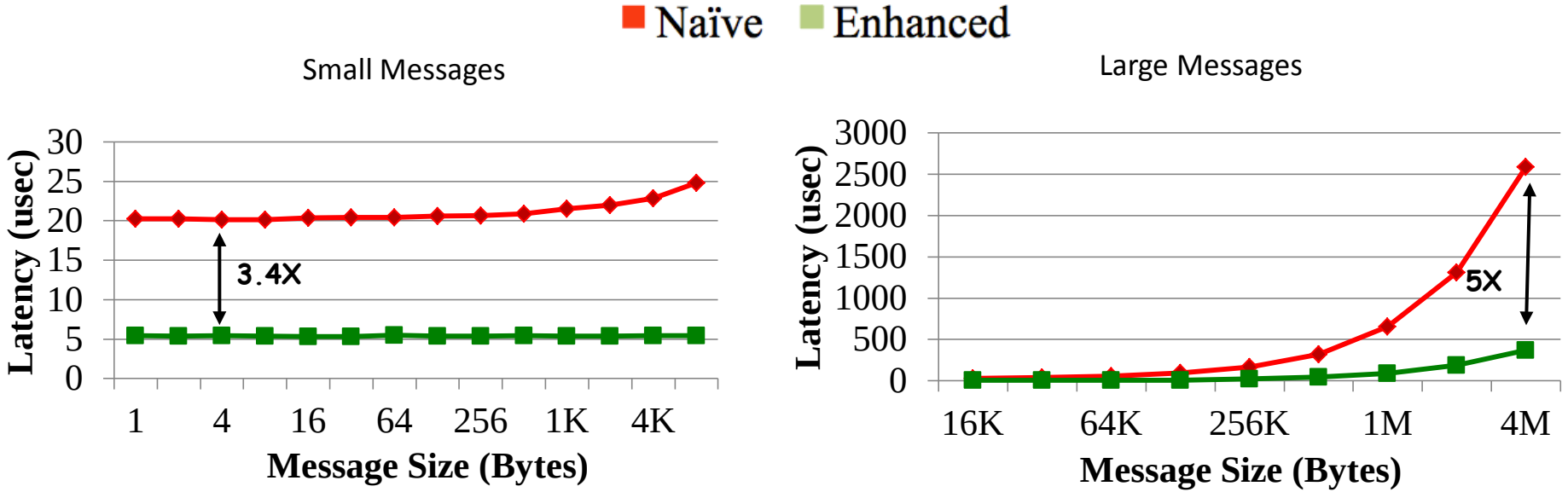
- Pipelined data transfers through host memory - overlap between CUDA copies and IB transfers
- Done transparently by the runtime
- **Designs with GPUDirect RDMA can help considerably improve performance**
- **Hybrid design:**
  - **GPUDirect RDMA**
  - **Pipeline 1 Hop**
  - **Proxy-based design**



Outline

- Overview of PGAS models (UPC and OpenSHMEM)
- Limitations in PGAS models for GPU computing
- Proposed Designs and Alternatives
- Performance Evaluation
- Conclusions

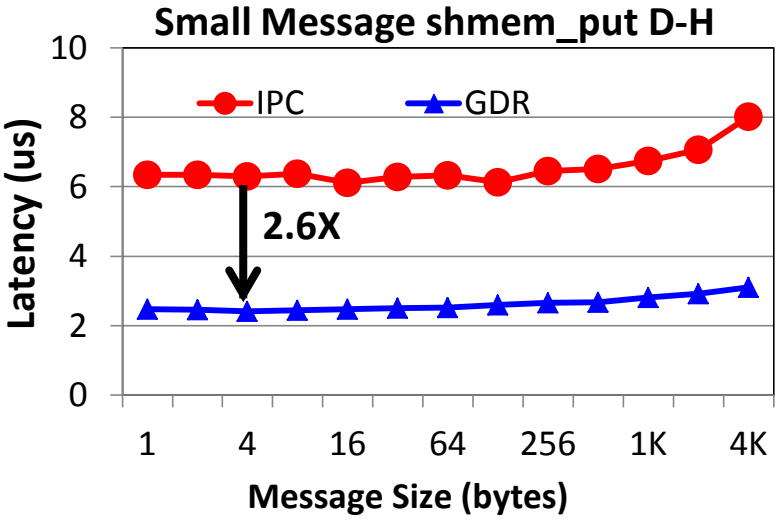
Shmem_putmem Intra-node Communication (IPC)

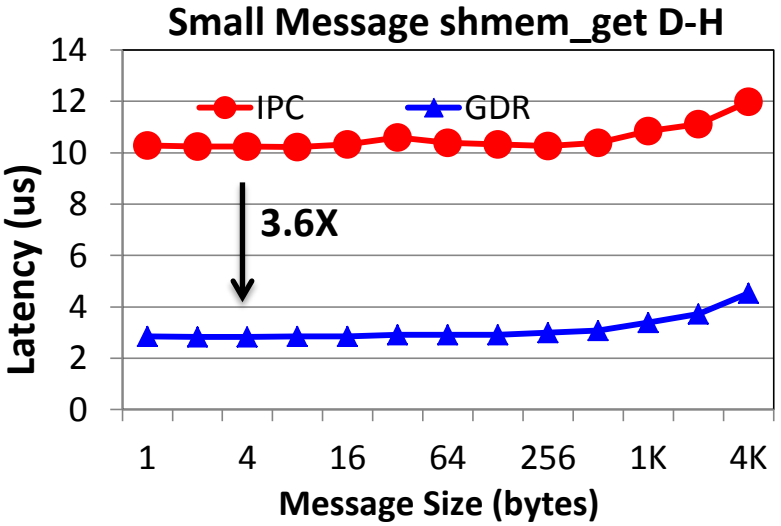


- Using IPC for intra-node communication
- Small messages – 3.4X improvement for 4Byte messages
- Large messages – 5X for 4MByte messages

Based on MVAPICH2X-2.0b + Extensions
Intel WestmereEP node with 8 cores
2 NVIDIA Tesla K20c GPUs, Mellanox QDR HCA
CUDA 6.0RC1

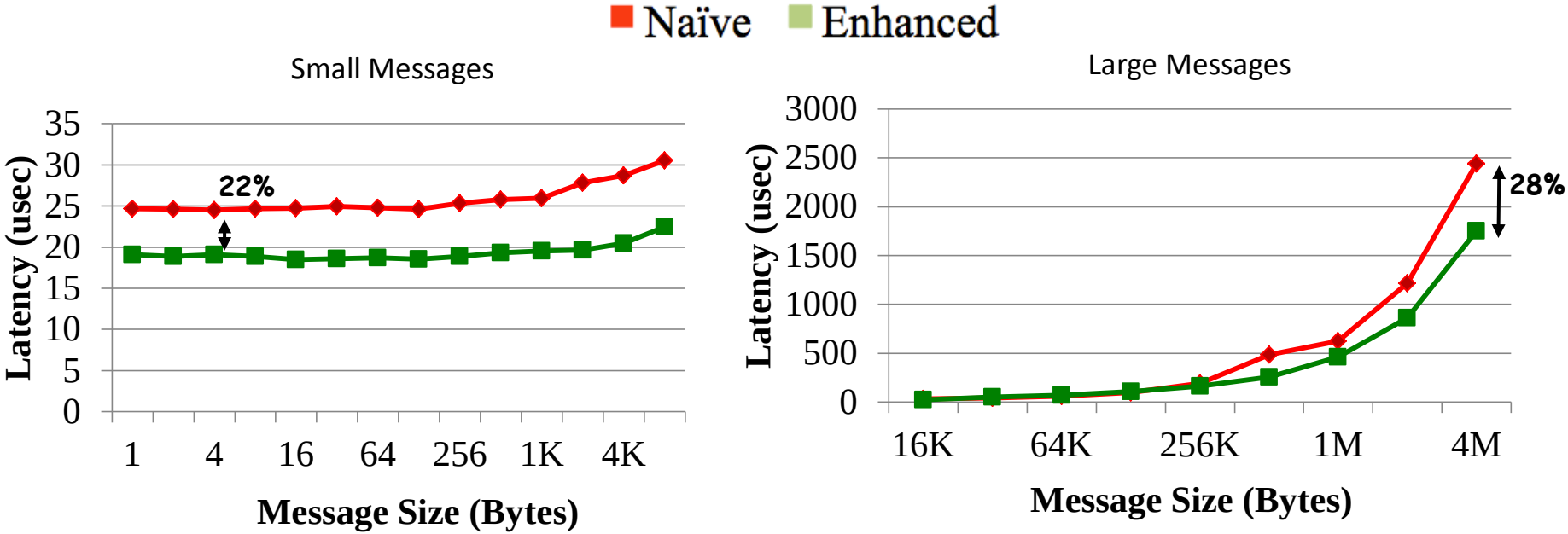
Shmem_putmem Intra-node Communication (GDR Enhancement)





- GDR for small and medium message sizes
- IPC for large message to avoid PCIe bottlenecks
- Hybrid design brings best of both
- 2.42 us Put D-H latency for 4 Bytes (2.6X improvement) and 3.92 us latency for 4 KBytes
- 3.6X improvement for Get operation
- Similar results with other configurations (D-D, H-D and D-H)

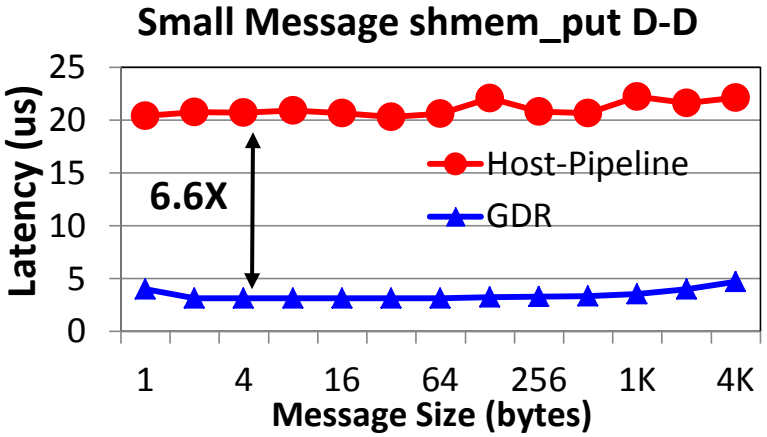
Shmem_putmem Inter-node Communication (Pipeline)

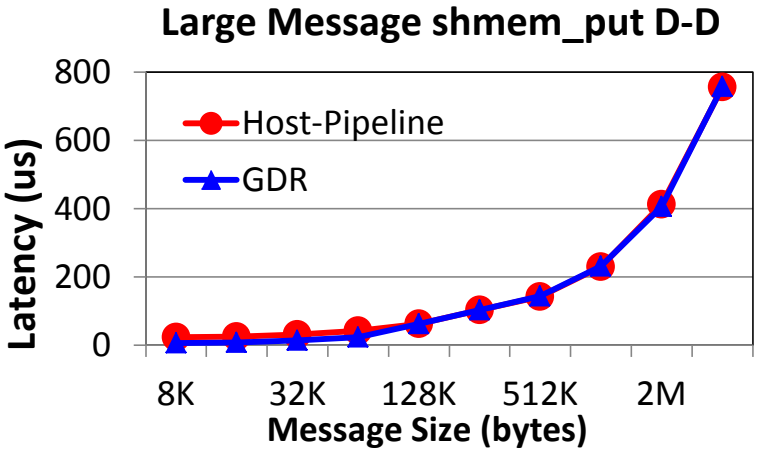


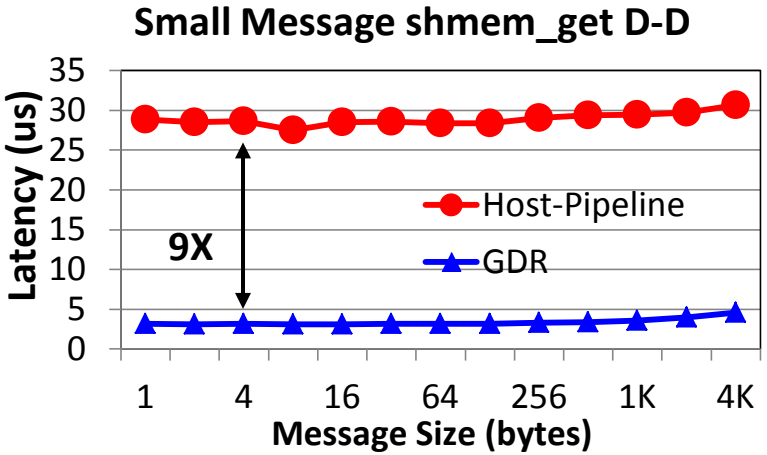
- Small messages benefit from selective CUDA registration – 22% for 4Byte messages
- Large messages benefit from pipelined overlap – 28% for 4MByte messages

S. Potluri, D. Bureddy, H. Wang, H. Subramoni and D. K. Panda, Extending OpenSHMEM for GPU Computing, Int'l Parallel and Distributed Processing Symposium (IPDPS '13)

Shmem_putmem Inter-node Communication (GDR and Proxy Enhancement)

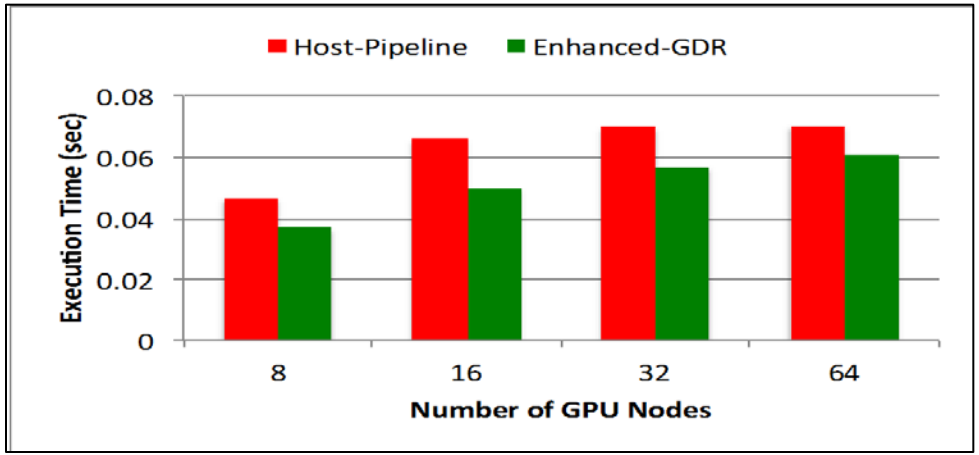




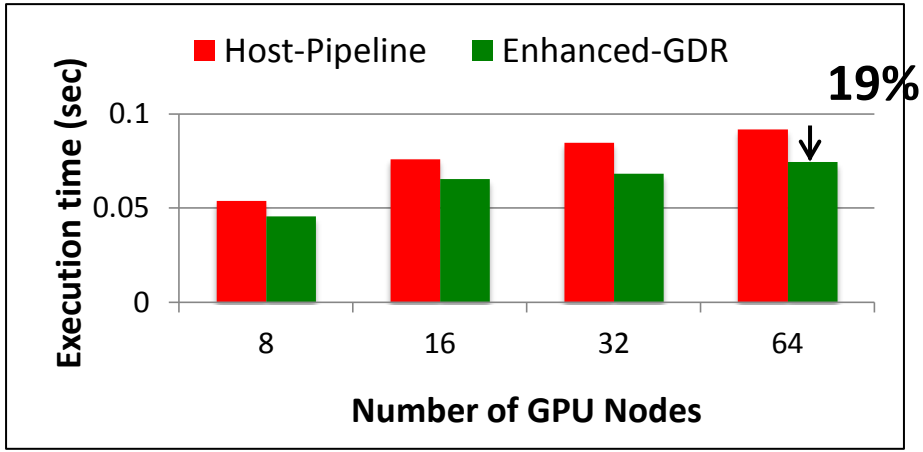


- GDR for small/medium message sizes
- Host-staging for large message to avoid PCIe bottlenecks
- Hybrid design brings best of both
- 3.13 us Put latency for 4B (6.6X improvement) and 4.7 us latency for 4KB
- 9X improvement for Get latency of 4B

Application Evaluation: 2DStencil



2DStencil 1Kx1K

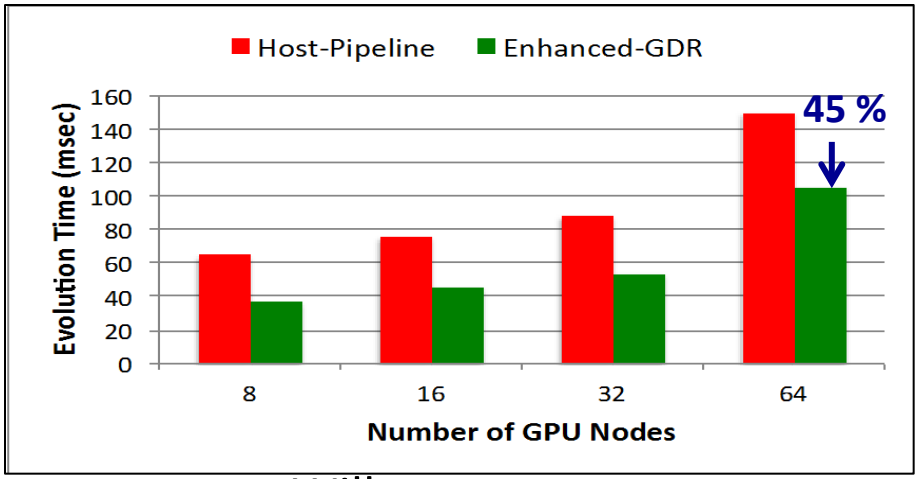


2DStencil 2Kx2K

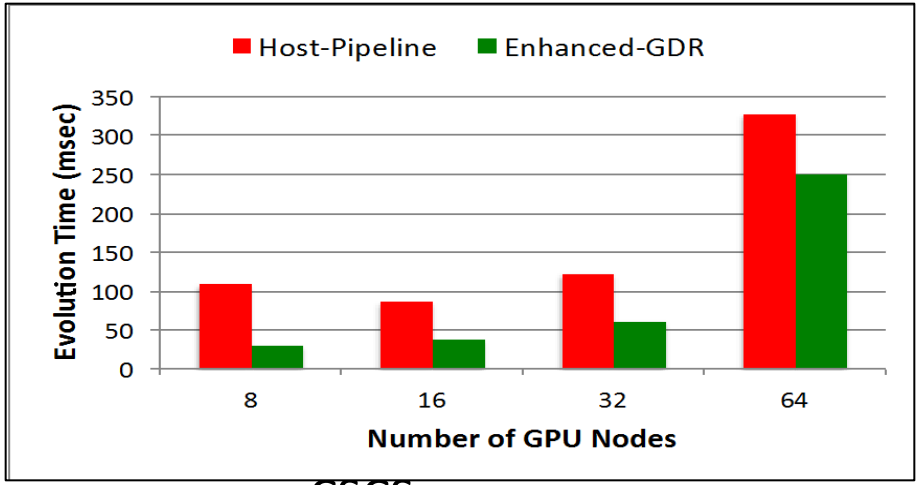
- Platform: Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- New designs achieve 20% and 19% improvements on 32 and 64 GPU nodes for 2Kx2K input size

K . Hamidouche, A. Venkatesh, A. Awan, H. Subramoni, C. Ching and D. K. Panda, Exploiting GPUDirect RDMA in Designing High Performance OpenSHMEM for GPU Clusters. IEEE Cluster 2015

Application Evaluation: GPULBM



Wilkes



CSCS

- Redesign the application
  - CUDA-Aware MPI : Send/Recv=> hybrid CUDA-Aware MPI+OpenSHMEM
  - cudaMalloc =>shmalloc(size,1);
  - MPI_Send/recv => shmem_put + fence
  - 45% benefits

- Platform: Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- Platform: CSCS (CS-Storm based): Haswell + NVIDA K80+ IB FDR

K. Hamidouche, A. Venkatesh, A. Awan, H. Subramoni, C. Ching and D. K. Panda, Exploiting GPUDirect RDMA in Designing High Performance OpenSHMEM for GPU Clusters. Elsevier PARCO Journal

Outline

- Overview of PGAS models (UPC and OpenSHMEM)
- Limitations in PGAS models for GPU computing
- Proposed Designs and Alternatives
- Performance Evaluation
- Conclusions

Conclusions

- PGAS models offer lightweight synchronization and one-sided communication semantics
- Low-overhead synchronization is suited for GPU architectures
- Extensions to the PGAS memory model needed to efficiently support CUDA-Aware PGAS models
- High-performance GDR-based design for OpenSHMEM is proposed
- Plan on exploiting the GDR-based designs for UPC and UPC++
- Enhanced designs are planned to be incorporated into MVAPICH2-X

International Workshop on Communication Architectures at Extreme Scale (ExaComm)

ExaComm 2015 was held with Int'l Supercomputing Conference (ISC '15), at Frankfurt, Germany, on Thursday, July 16th, 2015

One Keynote Talk: John M. Shalf, CTO, LBL/NERSC
Four Invited Talks: Dror Goldenberg (Mellanox); Martin Schulz (LLNL);
Cyriel Minkenberg (IBM-Zurich); Arthur (Barney) Maccabe (ORNL)
Panel: Ron Brightwell (Sandia)
Two Research Papers

ExaComm 2016 will be held in conjunction with ISC '16

http://web.cse.ohio-state.edu/~subramon/ExaComm16/exacomm16.html

Technical Paper Submission Deadline: Friday, April 15, 2016

Funding Acknowledgments

Funding Support by

Equipment Support by

Personnel Acknowledgments

Current Students

- A. Augustine (M.S.)
- A. Awan (Ph.D.)
- S. Chakraborthy (Ph.D.)
- C.-H. Chu (Ph.D.)
- N. Islam (Ph.D.)
- M. Li (Ph.D.)
- K. Kulkarni (M.S.)
- M. Rahman (Ph.D.)
- D. Shankar (Ph.D.)
- A. Venkatesh (Ph.D.)
- J. Zhang (Ph.D.)

Current Research Scientists

- H. Subramoni
- X. Lu

Current Senior Research Associate

- K. Hamidouche

Current Post-Doc

- J. Lin
- D. Banerjee

Current Programmer

- J. Perkins

Current Research Specialist

- M. Arnold

Past Students

- 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.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- 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.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- R. Rajachandrasekar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)

Past Post-Docs

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

Past Research Scientist

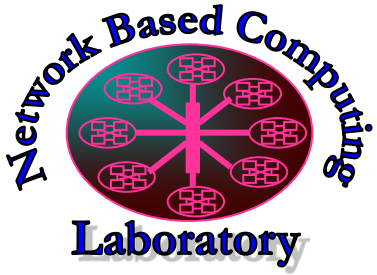
- S. Sur

Past Programmers

- D. Bureddy

Thank You!

panda@cse.ohio-state.edu



Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

The MVAPICH2 Project
http://mvapich.cse.ohio-state.edu/