MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

Enabling Efficient Use of UPC and OpenSHMEM PGAS Models on GPU Clusters

Presented at GTC ’15

MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

Presented by

Dhabaleswar K. (DK) Panda

The Ohio State University

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

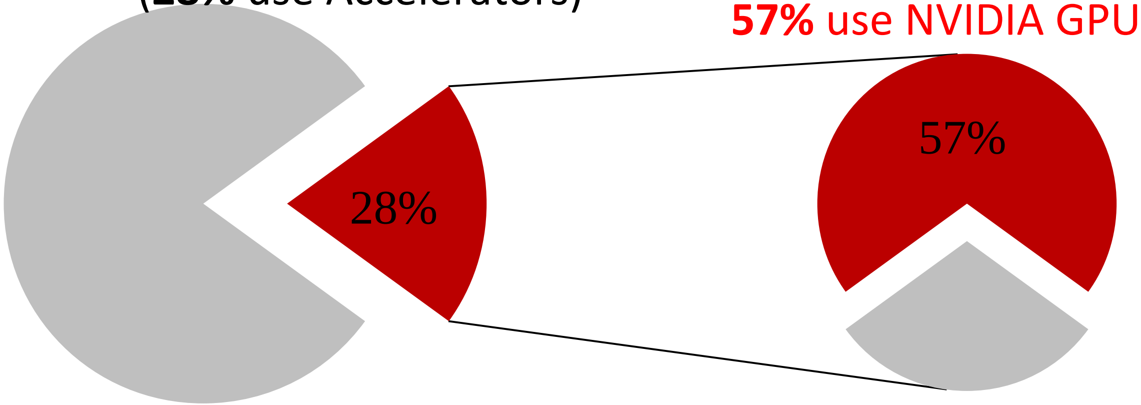
http://www.cse.ohio-state.edu/~panda

Accelerator Era

- Accelerators are becoming common in high-end system architectures

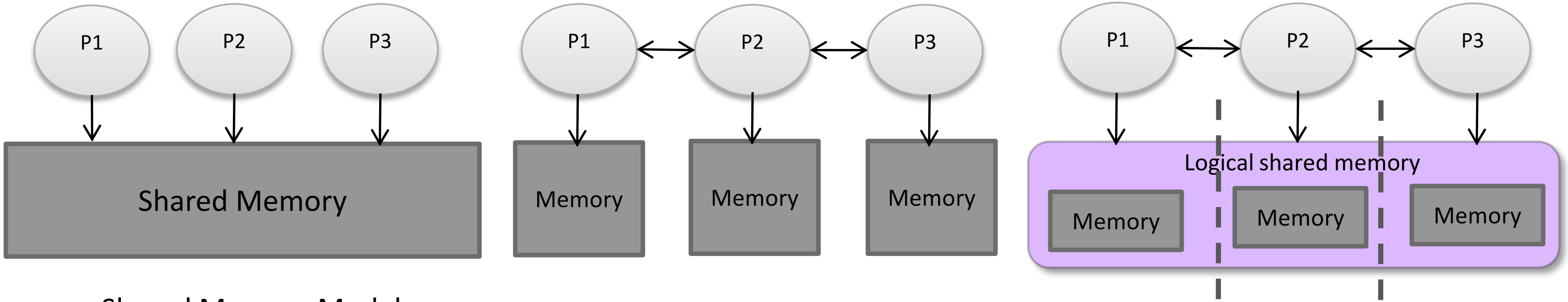
Top 100 – Nov 2014
(**28%** use Accelerators)

57% use NVIDIA GPUs



- Increasing number of workloads are being ported to take advantage of NVIDIA GPUs
- As they scale to large GPU clusters with high compute density – higher the synchronization and communication overheads – higher the penalty
- Critical to minimize these overheads to achieve maximum performance

Parallel Programming Models Overview



Shared Memory Model

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.
- Each model has strengths and drawbacks - suite different problems or applications

Outline

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

Partitioned Global Address Space (PGAS) Models

- PGAS models, an attractive alternative to traditional message passing

  - Simple shared memory abstractions

  - Lightweight one-sided communication

  - Flexible synchronization

- Different approaches to PGAS

  - Libraries
    - OpenSHMEM
    - Global Arrays
    - Chapel

  - Languages
    - Unified Parallel C (UPC)
    - Co-Array Fortran (CAF)
    - X10

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 applications 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

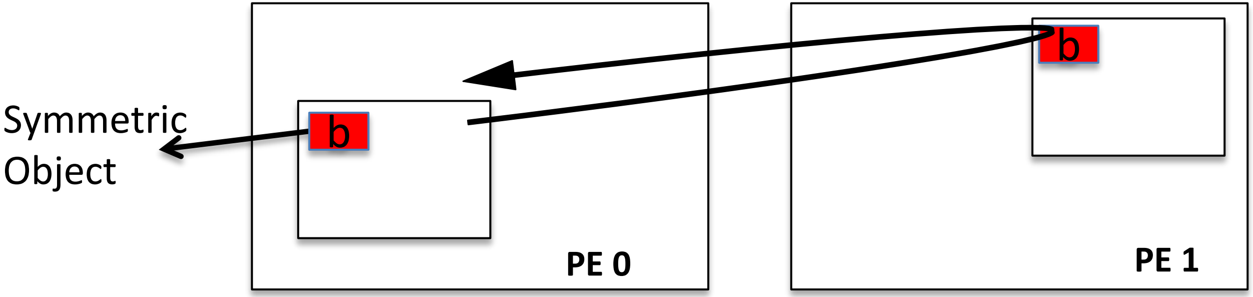
SGI SHMEM is the baseline

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)
```

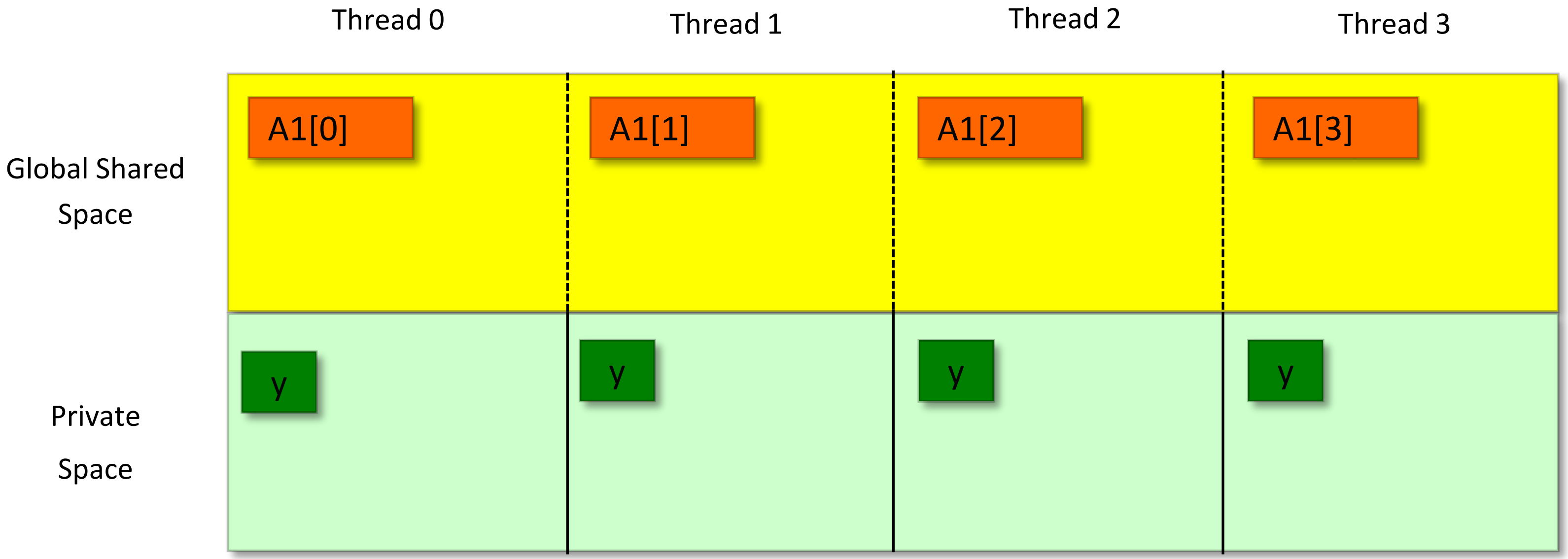
```
int main (int c, char *v[]) {
   int *b;

   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



Thread 0 Thread 1 Thread 2 Thread 3

Global Shared Space

A1[0] A1[1] A1[2] A1[3]

Private Space

y y y y

- 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
- Example:

```
shared int A1[THREADS];   //shared variable
int main() {
        int y;            //private variable
        A1[0] = 0;       //local access
        A1[1] = 1;       //remote access
}
```

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

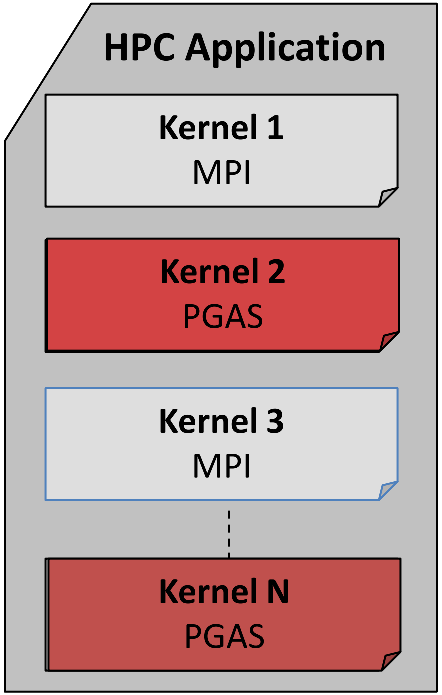
- Re-writing complete applications can be a huge effort
- Port critical kernels to the desired model instead

Hybrid (MPI+PGAS) Programming

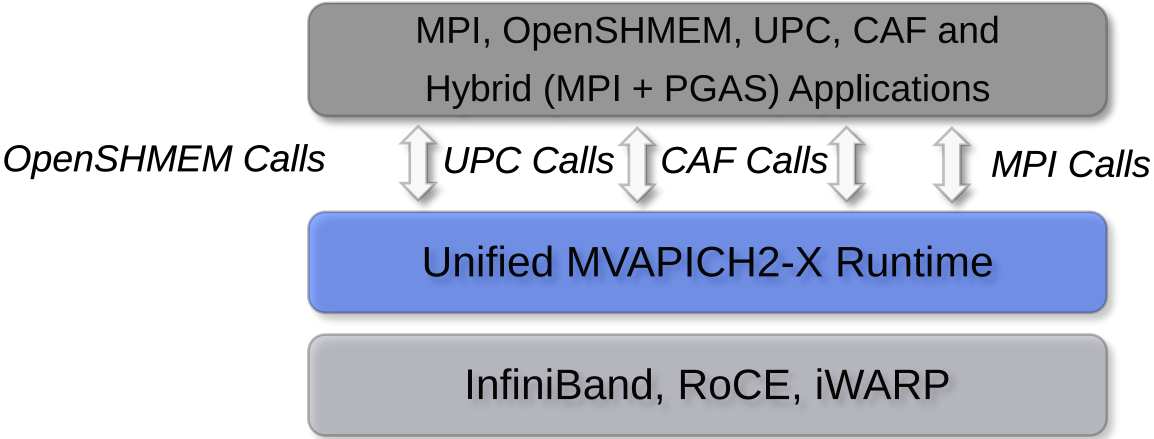
- Application sub-kernels can be re-written in MPI/PGAS based on communication characteristics

- Benefits:
  - Best of Distributed Computing Model
  - Best of Shared Memory Computing Model

- Exascale Roadmap*:
  - “Hybrid Programming is a practical way to program exascale systems”



MVAPICH2-X for Hybrid MPI + PGAS Applications



MPI, OpenSHMEM, UPC, CAF and Hybrid (MPI + PGAS) Applications

OpenSHMEM Calls *UPC Calls* *CAF Calls* *MPI Calls*

Unified MVAPICH2-X Runtime

InfiniBand, RoCE, iWARP

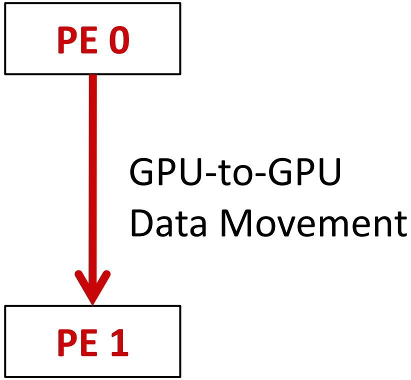
- Unified communication runtime for MPI, UPC, OpenSHMEM,CAF available from MVAPICH2-X 1.9 : (09/07/2012) http://mvapich.cse.ohio-state.edu
- Feature Highlights
  - Supports MPI(+OpenMP), OpenSHMEM, UPC, CAF, MPI(+OpenMP) + OpenSHMEM, MPI(+OpenMP) + UPC, MPI(+OpenMP) + CAF
  - MPI-3 compliant, OpenSHMEM v1.0 standard compliant, UPC v1.2 standard compliant, CAF 2015 standard compliant
  - Scalable Inter-node and intra-node communication – point-to-point and collectives
- Effort underway for support on NVIDIA GPU clusters

Outline

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

Limitations of PGAS models for GPU Computing

- PGAS memory models does not support disjoint memory address spaces - case with GPU clusters
- OpenSHMEM case
- Copies severely limit the performance



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, . . . )
```

- Synchronization negates the benefits of one-sided communication
- Similar limitations in UPC

Outline

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

Global Address Space with Host and Device Memory

heap_on_device();

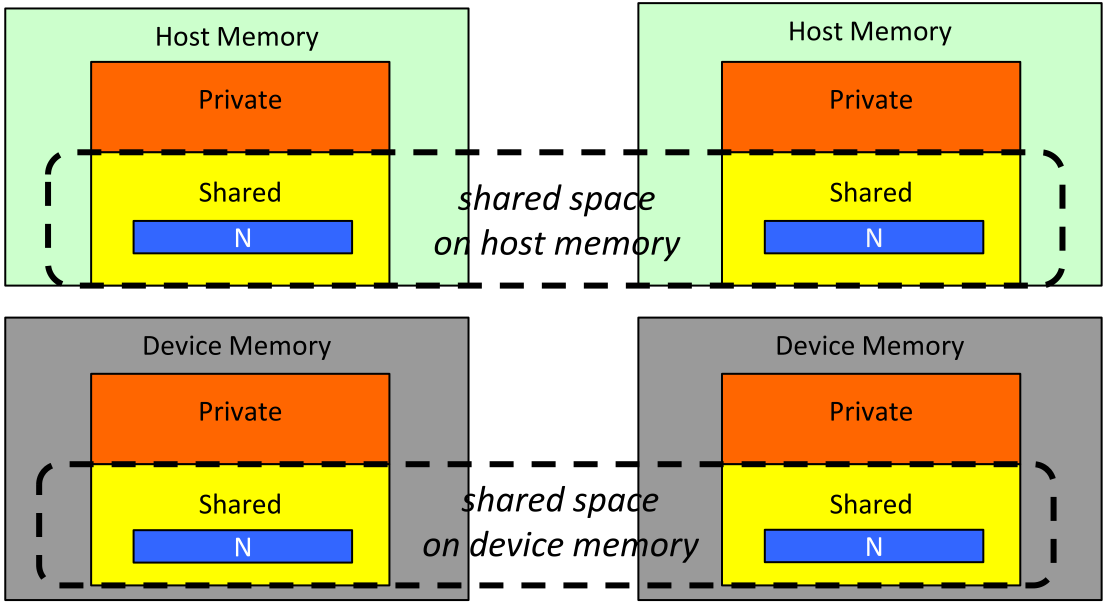
*/*allocated on device*/*

dev_buf = shmalloc (sizeof(int));

heap_on_host();

*/*allocated on host*/*

host_buf = shmalloc (sizeof(int));



Extended APIs:

heap_on_device/heap_on_host

a way to indicate location on heap

Can be similar for dynamically allocated memory in UPC

CUDA-aware OpenSHMEM and UPC runtimes

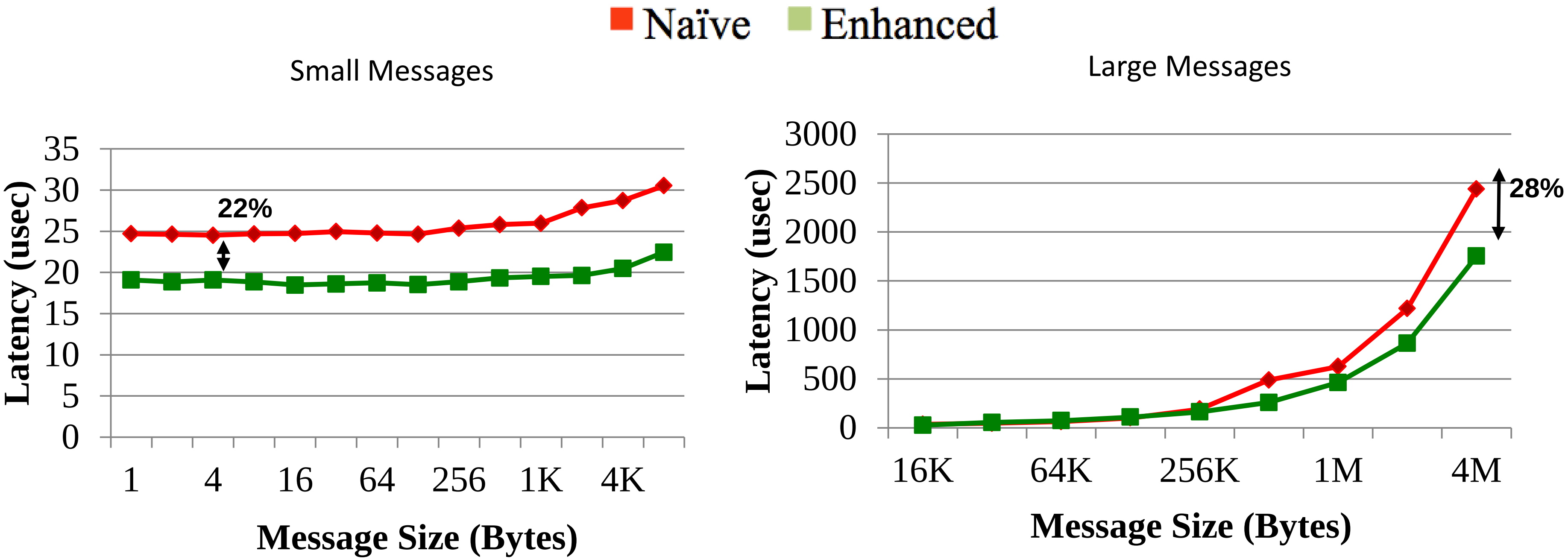
- After device memory becomes part of the global shared space:
  - Accessible through standard UPC/OpenSHMEM communication APIs
  - Data movement transparently handled by the runtime
  - Preserves one-sided semantics at the application level
- Efficient designs to handle communication
  - Inter-node 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

Outline

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

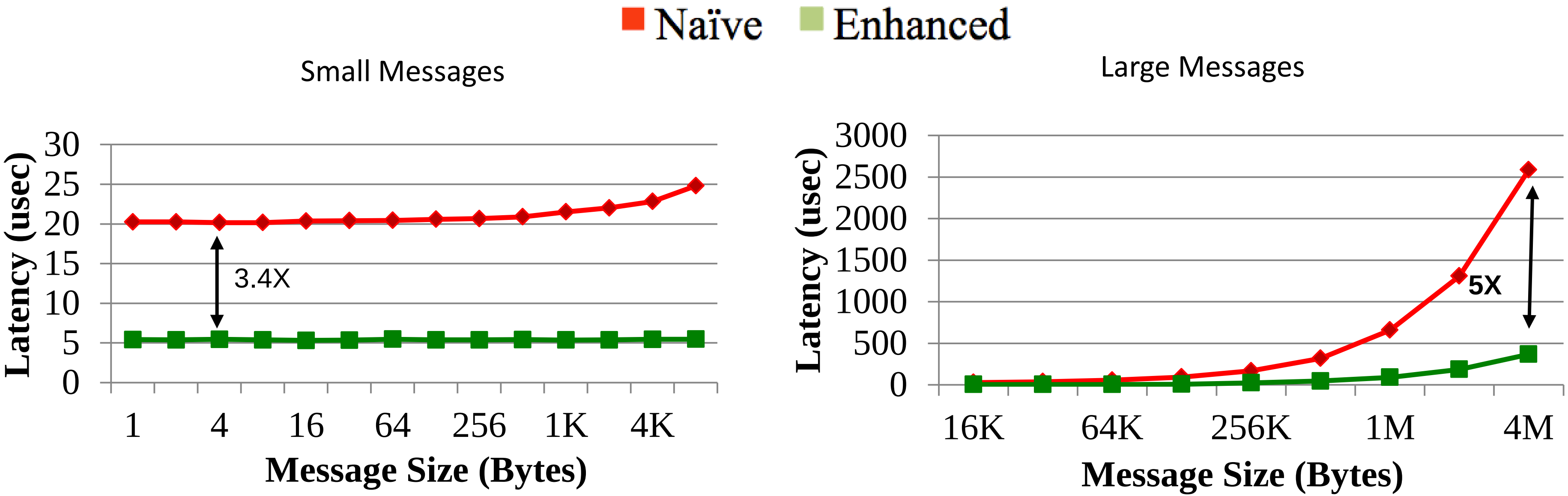
Shmem_putmem Inter-node Communication



- 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)

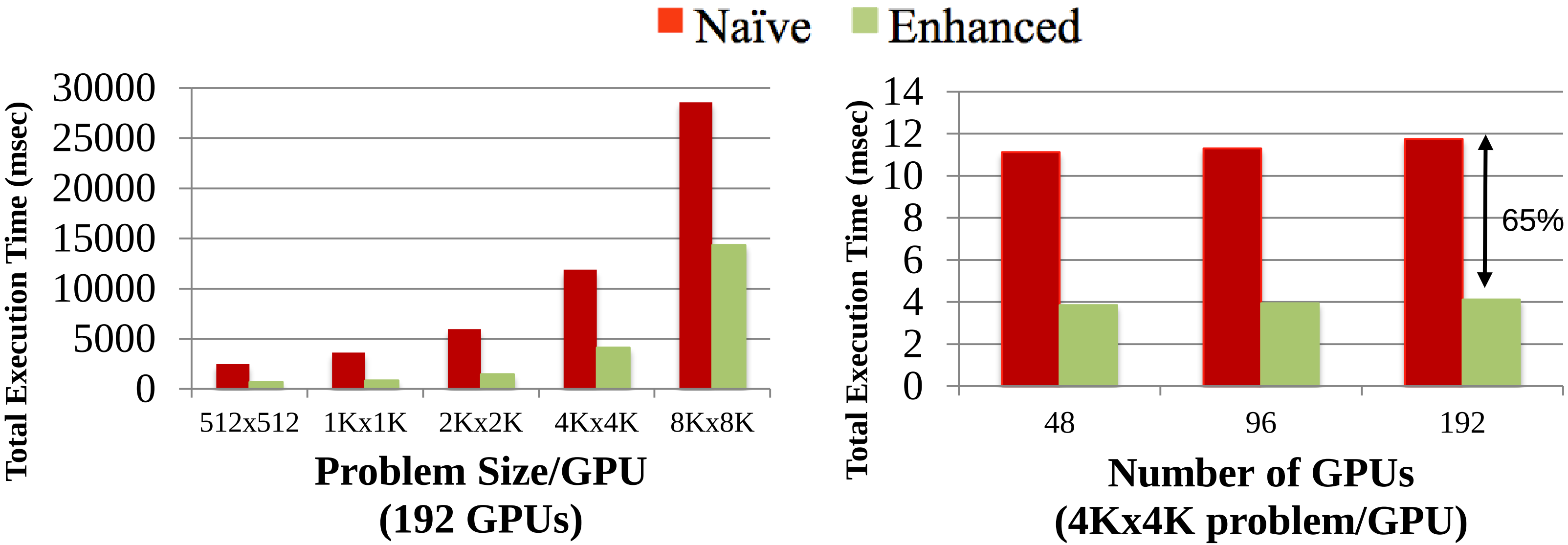
Shmem_putmem Intra-node Communication



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

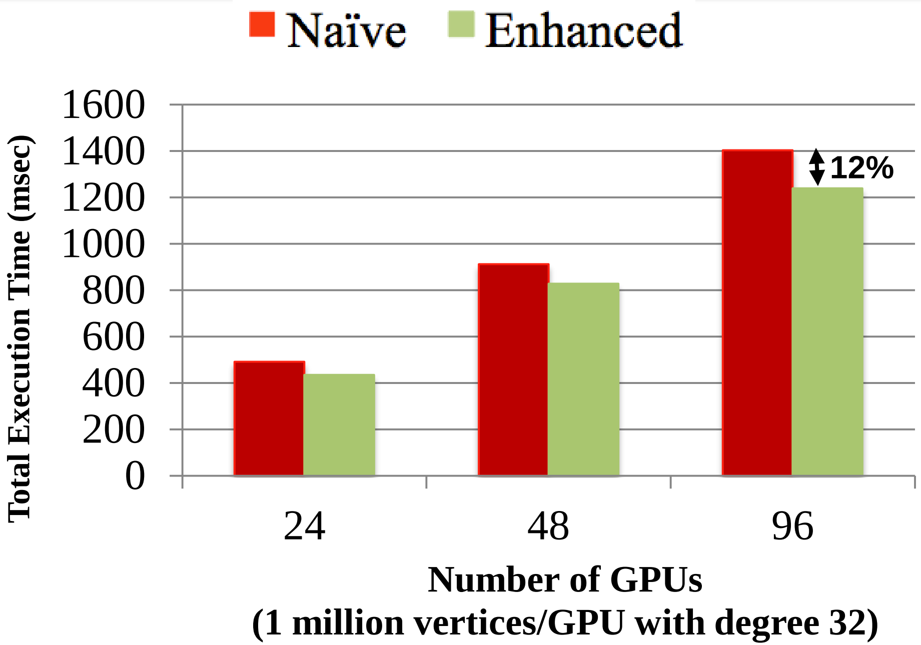
Based on MVAPICH2X-2.0b + Extensions
Intel WestmereEP node with 8 cores
2 NVIDIA Tesla K20c GPUs, Mellanox QDR HCA
CUDA 6.0RC1

Application Kernel Evaluation: Stencil2D



- Modified SHOC Stencil2D kernel to use OpenSHMEM for cluster level parallelism
- The enhanced version shows 65% improvement on 192 GPUs with 4Kx4K problem size/GPU
- Using OpenSHMEM for GPU-GPU communication allows runtime to optimize non-contiguous transfers

Application Kernel Evaluation: BFS



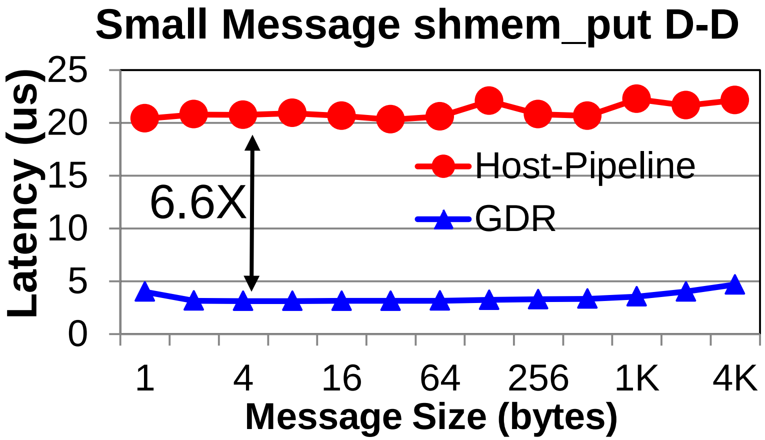
- Extended SHOC BFS kernel to run on a GPU cluster using a level-synchronized algorithm and OpenSHMEM

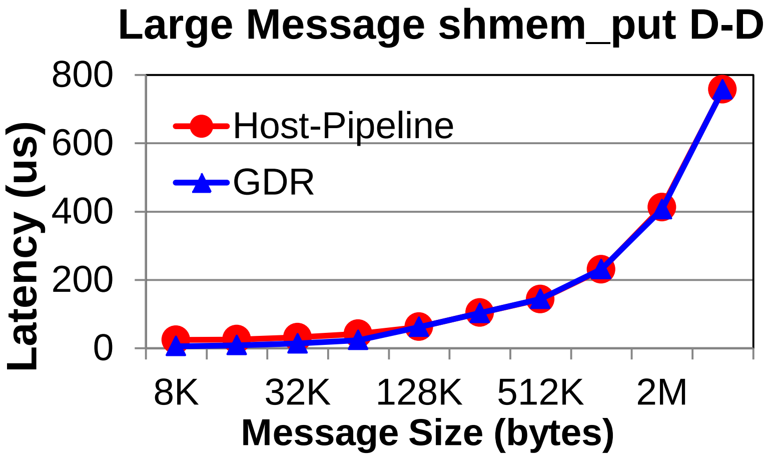
- The enhanced version shows up to 12% improvement on 96 GPUs, a consistent improvement in performance as we scale from 24 to 96 GPUs.

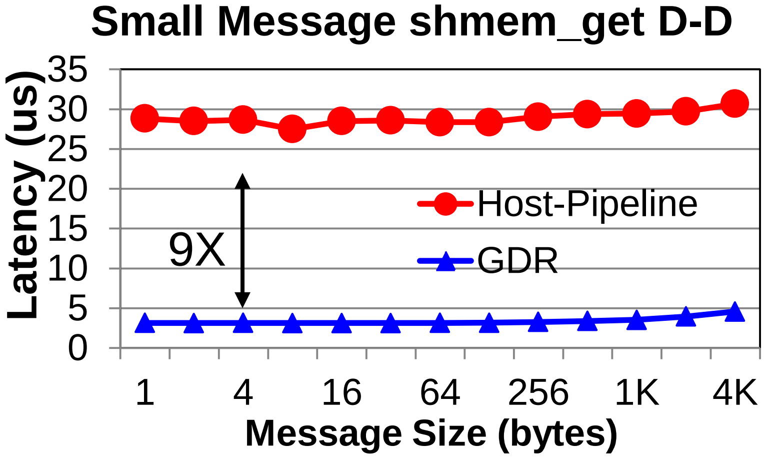
Outline

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

OpenSHMEM: Inter-node Evaluation

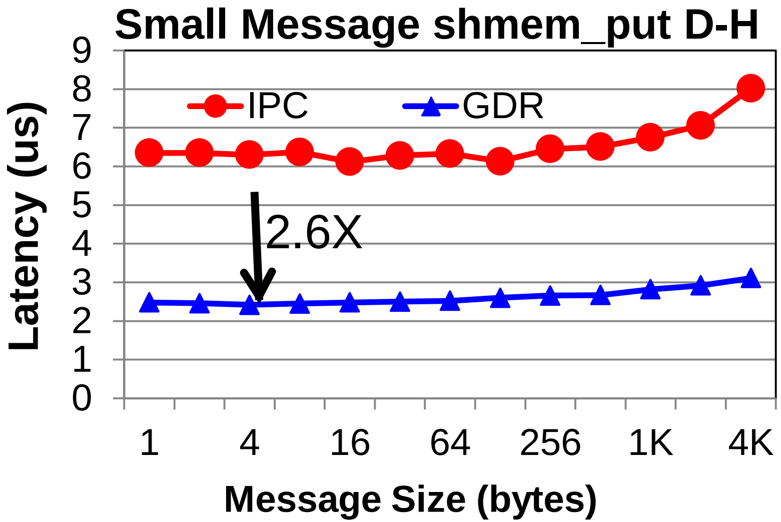


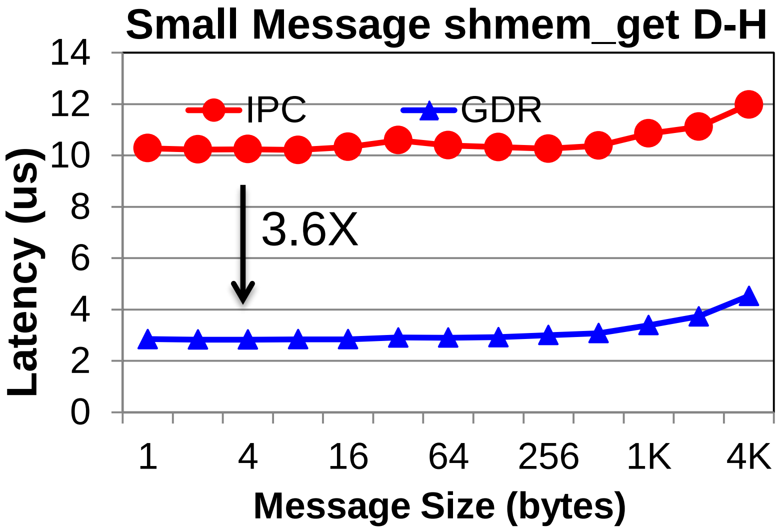




- GDR for small/medium message sizes
- Host-staging for large message to avoid PCIe bottlenecks
- Hybrid design brings best of both
- 3.13 us Put latency for 4B (6.6X improvement) and 4.7 us latency for 4KB
- 9X improvement for Get latency of 4B

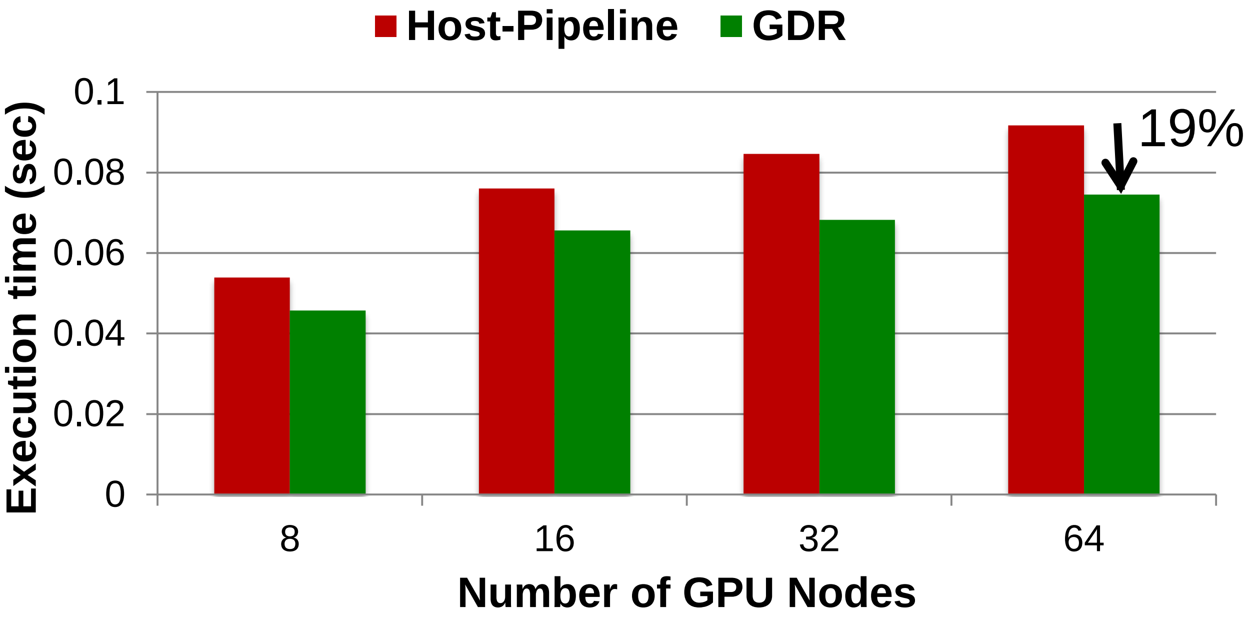
OpenSHMEM: Intra-node Evaluation





- 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)

OpenSHMEM: Stencil3D Application Kernel Evaluation



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

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. (under review)

Conclusions

- PGAS models offers lightweight synchronization and one-sided communication semantics
- Low-overhead synchronization is suited for GPU architecture
- Extensions to the PGAS memory model to efficiently support CUDA-Aware PGAS models.
- High efficient GDR-based designs for OpenSHMEM
- Plan on exploiting the GDR-based designs for UPC
- Enhanced designs are planned to be incorporated into MVAPICH2-X

Two More Talks

- Learn more on how to combine and take advantage of Multicast and GPUDirect RDMA simultaneously
    - S5507 – High Performance Broadcast with GPUDirect RDMA and InfiniBand Hardware Multicast for Streaming Application
    - Thursday, 03/19 (Today)
    - Time: 14:00 -- 14:25
    - Room 210 D
- Learn about recent advances and upcoming features in CUDA-aware MVAPICH2-GPU library
    - S5461 - Latest Advances in MVAPICH2 MPI Library for NVIDIA GPU Clusters with InfiniBand
    - Thursday, 03/19 (Today)
    - Time: 17:00 -- 17:50
    - Room 212 B

Thanks! Questions?

Contact

panda@cse.ohio-state.edu

MVAPICH

Network Based Computing Laboratory

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

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