MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

A Case for Non-Blocking Collectives in OpenSHMEM: Design, Implementation, and Performance Evaluation using MVAPICH2-X

Ammar Ahmad Awan, **Khaled Hamidouche**, Ching-Hsiang Chu, and Dhabaleswar K. Panda

Network-Based Computing Laboratory
Department of Computer Science and Engineering
The Ohio State University, Columbus, OH, U.S.A

Outline

- **Introduction**
- Problem Statement
- Contributions
  - Interface API Change
  - Design and Implementation
  - Micro-benchmarks
- Performance Evaluation
- Conclusion

Collective Communication

- An important and widely used communication pattern for both MPI and PGAS model
- Primitives available in both PGAS and MPI model
  - Reduce, Broadcast, Gather (Collect/Fcollect), Barrier etc.
- Collectives have been blocking
  - The context remains in the library till completion

Collectives in OpenSHMEM

- shmem_broadcast
- shmem_collect
- shmem_fcollect
- shmem_barrier
- shmem_reduce
  - Set of reductions using different data types

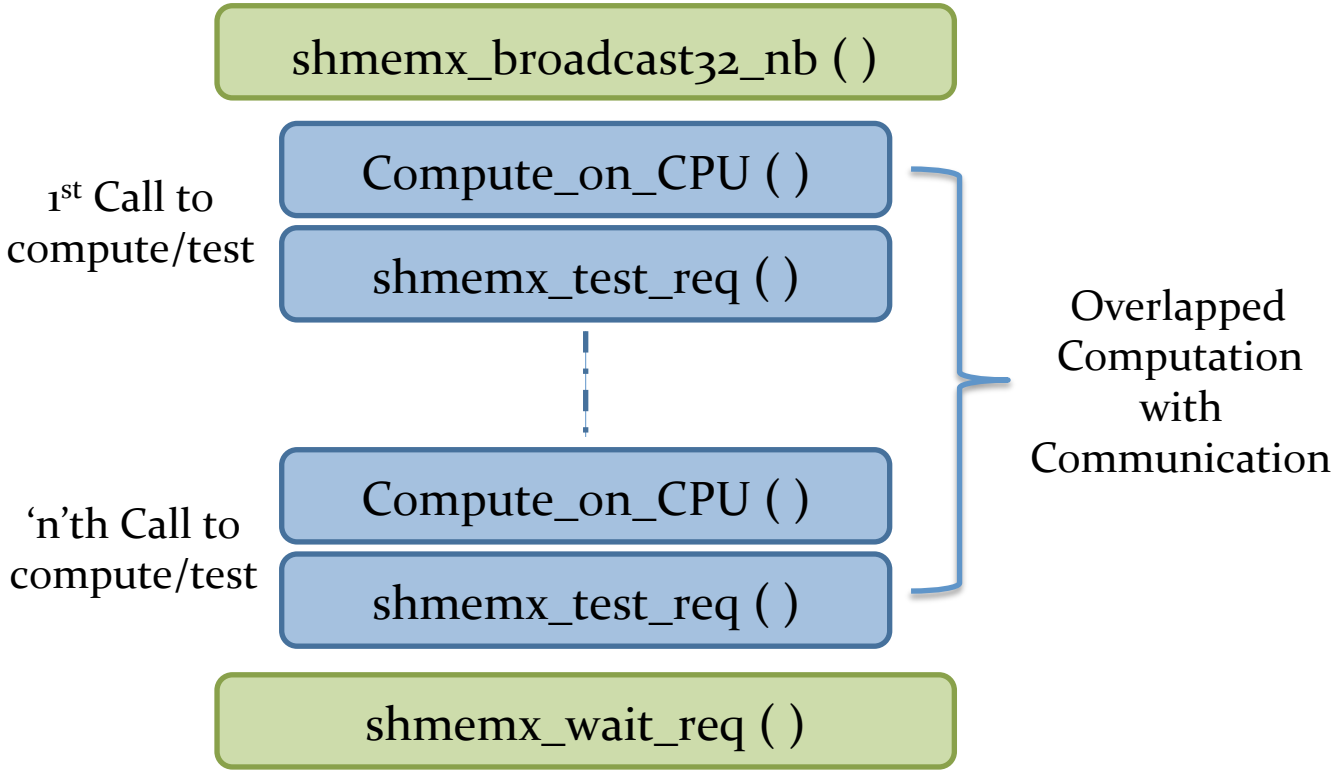
Non-Blocking Collectives (NBC)

- Have been used since 2007 in Message Passing Interface (MPI)
- Recently, made part of the MPI-3 standard
- NBC performance is good *
  - Latency is good with acceptable overhead posed by NBC operations
  - Overlap is the new parameter – maximizing it enables independent computation to proceed in background

*H. Subramoni, A. A. Awan, K. Hamidouche, D. Pekurovsky, A. Venkatesh, S. Chakraborty, K Tomko, and D.K. Panda. Designing Non-Blocking Personalized Collectives with Near Perfect Overlap for RDMA-Enabled Clusters . In Proceeding of the International Supercomputing Conference (ISC) '15, Frankfurt, Germany, July 2015.

*A. A. Awan, K. Hamidouche, A. Venkatesh, J. Perkins, H. Subramoni, and D. K. Panda.
GPU-Aware Design, Implementation, and Evaluation of Non-blocking Collective Bench- marks (Accepted). In *Proceedings of the 22nd European MPI Users' Group Meeting*, EuroMPI '15, Bordeaux, France, 2015. ACM.

Example of an NBC operation



Outline

- Introduction
- **Problem Statement**
- Contributions
  - Interface API Change
  - Design and Implementation
  - Micro-benchmarks
- Performance Evaluation
- Conclusion

Questions/Challenges

- Can we design OpenSHMEM NBC support with minimal changes to the current blocking interface?
- Can we implement the OpenSHMEM NBC interface in a high performance and extensible fashion?
- Can a set of micro-benchmarks be proposed that evaluate performance of any standard implementation of OpenSHMEM NBC operations?

Outline

- Introduction
- Problem Statement
- **Contributions**
  - **Interface API Change**
  - Design and Implementation
  - Micro-benchmarks
- Performance Evaluation
- Conclusion

Our Suggested Changes

- Earlier Proposal of NBC for OpenSHMEM by Poole et al. *
  - In addition to the NBC, ***explicit creation of Active Sets*** was also proposed
- We propose changes to the API presented by Poole et al.
  - Use the same API as blocking collectives
  - Add an additional "request" argument to the end of function signature
  - Handle Active Sets inside the runtime (like communicator creation in MPI)
  - Keep the triplet – *PE_start, logPE_stride, PE_end*
  - Any applications written with blocking collectives can easily be ported by adding the **request** argument and the **wait** call

*StephenPoole,PavelShamis,AaronWelch,SwaroopPophale,ManjunathGorentlaVenkata, Oscar Hernandez, Gregory Koenig, Tony Curtis, and Chung-Hsing Hsu. Openshmem extensions and a vision for its future direction. OpenSHMEM 2014, Annapolis, MD, USA, 2014.

Example (Broadcast Collective)

```
/* Blocking Broadcast - Current Implementation */
void
shmem_broadcast32 (void *target, const void *source,
                   size_t nelems, int PE_root, int PE_start,
                   int logPE_stride, int PE_size, long *pSync);


/* Non-Blo                                  Proposed in this paper */
void
shmemx_broadcast32_nb (void *target, const void *source,
                       size_t nelems, int PE_root, int PE_start,
                       int logPE_stride, int PE_size, long *pSync,
                       int *request);


/* Non-Blocking Broadcast - Proposed by Poole et al. */
void
shmem_broadcast32_nb (void *target, const void *source,
                      size_t nelems, struct shmem_aset aset,
                      long *pSync, shmem_request_handle_t request);
```

_nb prefix

x (extension)

Wait and Test calls

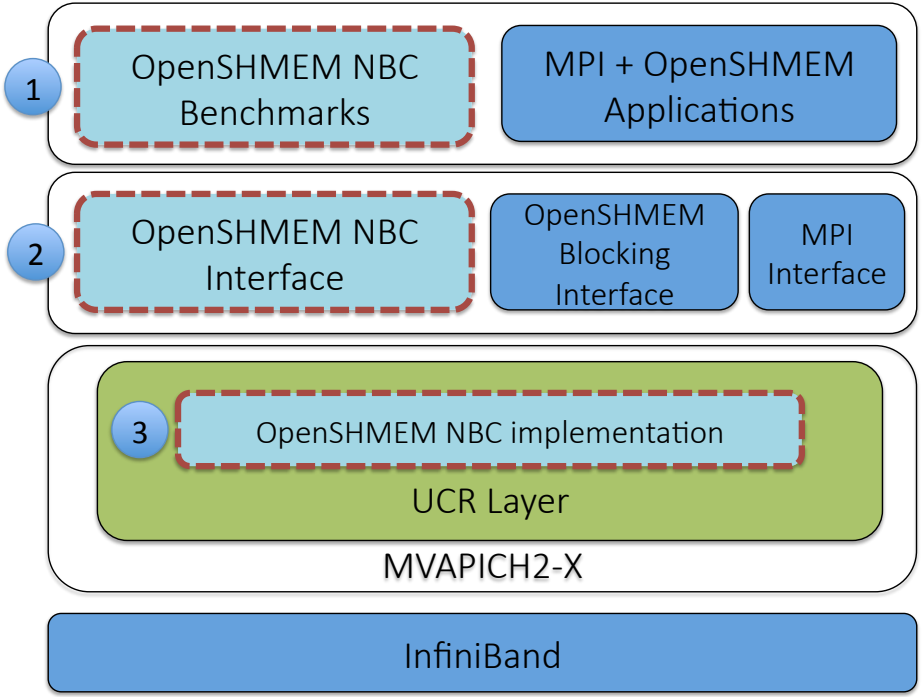
```
/* Test */
int
shmemx_test_req (int *request, int *flag, void *status);

/* Wait */
int
shmemx_wait_req (int *request, void *status);
```

Outline

- Introduction
- Problem Statement
- **Contributions**
  - Interface API Change
  - **Design and Implementation**
  - Micro-benchmarks
- Performance Evaluation
- Conclusion

Overview : Design and Implementation



High Performance Design

- Unified Communication Runtime in MVAPICH2-X
- The UCR layer combines the best of both MPI and PGAS model by
  - combining runtimes instead of
  - combining programming models

MVAPICH2 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 (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014**
  - **Support for Virtualization (MVAPICH2-Virt), Available since 2015**
  - **Used by more than 2,425 organizations in 75 countries**
  - **More than 279,000 downloads from the OSU site directly**
  - Empowering many TOP500 clusters (Jun '15 ranking)
    - 8th ranked 519,640-core cluster (Stampede) at TACC
    - 11th ranked 185,344-core cluster (Pleiades) at NASA
    - 22nd ranked 76,032-core cluster (Tsubame 2.5) at Tokyo Institute of Technology 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
- System-X from Virginia Tech (3rd in Nov 2003, 2,200 processors, 12.25 TFlops) -> Stampede at TACC (8th in Jun'15, 519,640 cores, 5.168 Plops)

Implementation at UCR level

- We have implemented OpenSHMEM NBC inside UCR
- Other implementations can implement it in any way (over GASNet and/or any comm. runtime)
- UCR unifies the Collective Algorithms for MPI and PGAS models
  - Takes advantage of all optimizations in the MPI Collective word
  - Maps both OpenSHMEM and MPI collective operations to the same algorithms
  - Transparent and lightweight Communicator Creation
  - Communicator Cache

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

Outline

- Introduction
- Problem Statement
- **Contributions**
  - Interface API Change
  - Design and Implementation
  - **Micro-benchmarks**
- Performance Evaluation
- Conclusion

OSU Micro-Benchmarks (OMB)

- OMB has OpenSHMEM benchmarks already
- In this paper, we propose
  - OpenSHMEM NBC benchmarks
  - Support for all Non-Blocking Collectives

Extending OMB : OpenSHMEM NBC

- We need extensions to the OMB for evaluating OpenSHMEM NBC
- Need to evaluate new parameters like
  - Overlap Percentage
  - Time for test calls
  - And more…
- Will help:
  - Evaluate implementations of OpenSHMEM NBC in a fair manner
  - Users redesigning their applications

Performance Evaluation Metrics

1. **Pure Comm. Latency** - Latency of an NBC when we call the collective immediately followed by shmemx_wait_req() call

2. **Overall Latency** - Latency of an NBC operation when we call the collective, followed by independent computation and specified number of test calls, followed by a shmemx_wait_req() call

3. **Blocking-Avg Latency** - Average latency of a Blocking Collective operation

4. **Compute Time** - Time taken by the dummy compute (independent overlapped computation) function.

5. **Test Time** - Time taken by shmemx_test_req() calls

6. **NBC Overhead** - This is the difference in performance of collective when its Pure Comm. latency is compared with Overall latency

Structure of the Benchmark

Timing of Pure comm

Estimate compute time using Pure comm and test time
- Initiate the compute function

shmemx_broadcast32_nb ( )

Compute_on_CPU
(Matrix Mul)

shmemx_test_req ( )

While elapsed time < Compute

shmemx_wait_req ( )

Outline

- Introduction
- Problem Statement
- Contributions
  - Interface API Change
  - Design and Implementation
  - Micro-benchmarks
- **Performance Evaluation**
- Conclusion

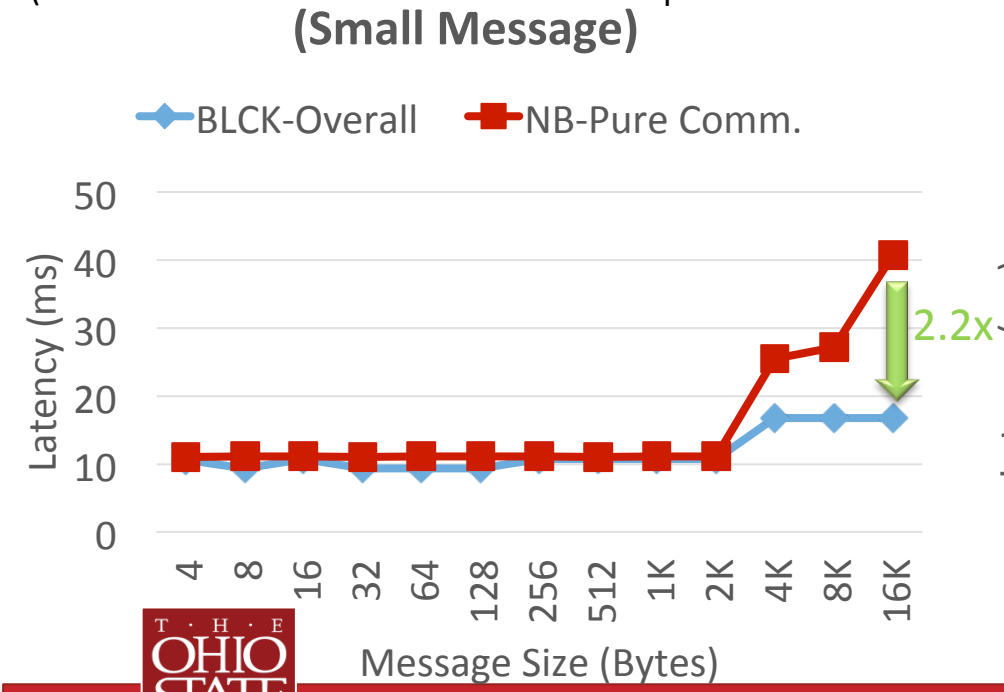
Experimental Setup

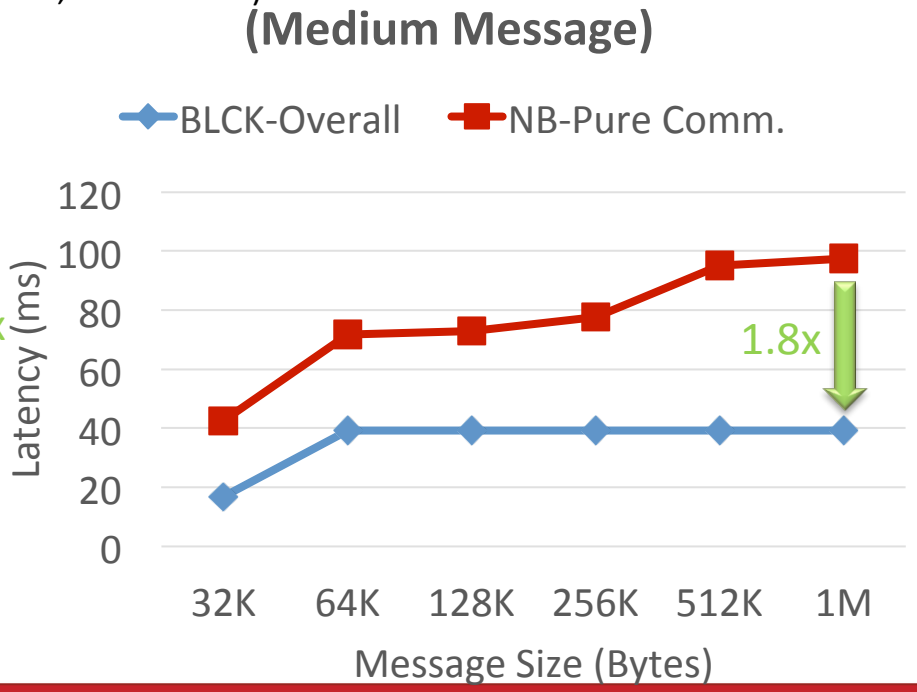
- RI cluster at The Ohio State University for all the experiments
- 160 compute nodes
- Each node:
  - 2 Intel E5630 processors 2.53 GHz
  - 8 cores per node
  - 12 GB of RAM per node
  - RHEL 6.5 (Kernel 2.6.32-431) operating system
  - Mellanox ConnectX-2 QDR InfiniBand cards (40 Gbps)
- Nodes connected with 1 Mellanox MTS3610 196-port InfiniBand switch configured as full fat-tree network

Blocking vs. Non-Blocking

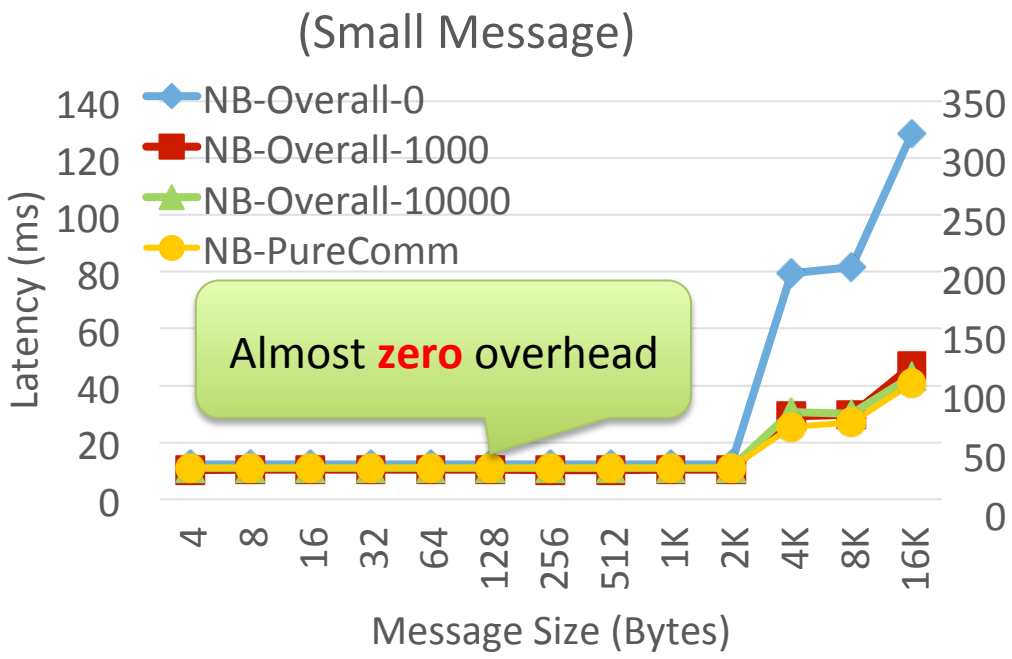
Reduce@512 proc.

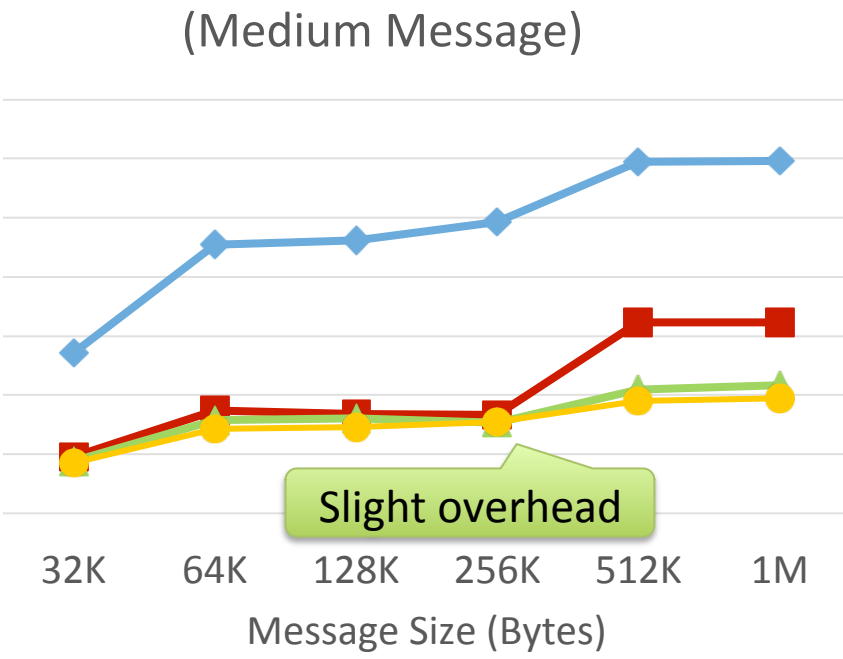
(similar trend can be observed for other operations such as fcollect, broadcast…)





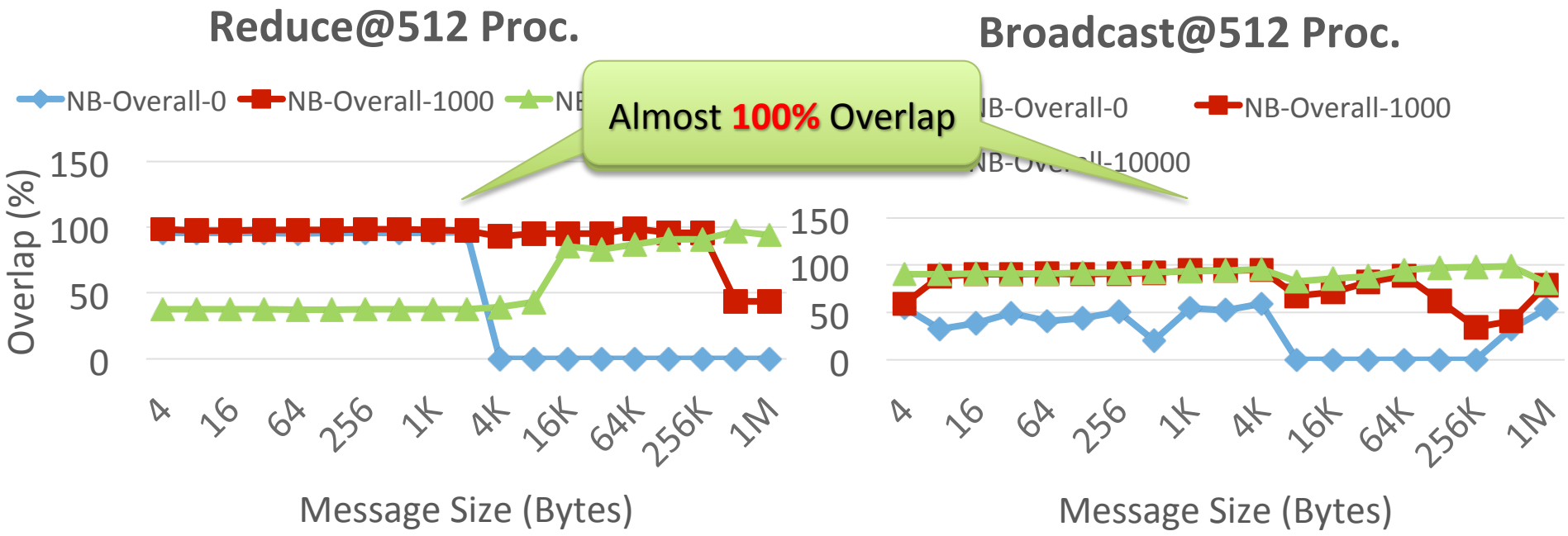
NBC Overhead : Reduce





Similar trends with other collectives

Overlap



Finding the appropriate number of Test is application dependent

Outline

- Introduction
- Problem Statement
- Contributions
  - Interface API Change
  - Design and Implementation
  - Micro-benchmarks
- Performance Evaluation
- **Conclusion**

Conclusion and Future Work

- Described the design and implementation details of OpenSHMEM NBC operations using MVAPICH2-X
- Designed and implemented new NBC Micro-benchmarks for evaluating OpenSHMEM NBC (part of OMB suite)
- Presented a comprehensive performance evaluation based on parameters like latency, overlap, NBC overhead, and effect of number of tests calls
- Design and optimize NBC algorithms for OpenSHMEM
- Application level redesign and study

Thank You!

Ammar Ahmad Awan, **Khaled Hamidouche**,
Ching-Hsiang Chu, and Dhabaleswar K. Panda

{awan.10, hamidouche.2, chu.368, panda.2} @osu.edu

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

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