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

Example of an NBC operation

shmemx_test_req ()
Compute_on_CPU ()
shmemx_test_req ()
shmemx_wait_req ()

1st Call to compute/test

2nd Call to compute/test

Overlapped Computation with Communication
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?

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

Example (Broadcast Collective)

Example Code:

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

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

Wait and Test calls
• Introduction
• Problem Statement
• Contributions
  – Interface API Change
  – Design and Implementation
  – Micro-benchmarks
• Performance Evaluation
• Conclusion

Outline

MVAPICH

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

Outline

Overview: Design and implementation

MVAPICH2 Software

• High-Performance open-source MPI library for Infiniband, 10GigE/RoE, and NVLink converged enhanced Ethernet (NoC)
  – MVAPICH (Oct '11), MVAPICH2 (Oct '12 and Oct '13), Available since 2012
  – MVAPICH2 ‐ Window, Available since 2012
  – Support for OpenSHMEM NBC and OpenSHMEM PGAS. Available since 2015
  – Support for Virtualization (MVAPICH2). Available since 2015
  – Used by more than 1,250 organizations in 75 countries
  – More than 270,000 downloads from the OSI site directly
  – Empowering many Top500 student (25 ‘15 ranking)
  – 4th ranked 115,400 core‐second (Stampede at TACC)
  – 1st ranked 105,100 core‐second (Pleasant at SDSC)
  – Available with software stacks of many 10, HAL, and vendor vendors including Linux Distros (Redhat and SuSE)
  – http://mvapich.cse.ohio-state.edu

• System in Top500 in June '15, 2.0G throughput, 12.25 TFlops in Stampede at TACC (8th in Jun’13, 519,640 cores, 3.0G Power)

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 unified the Collective Algorithms for MPI and PGAS models
  – Takes advantage of all optimization in the MPI Collective word
  – Maps both OpenSHMEM and MPI collective operations to the same algorithms
  – Transparent and lightweight Communicator Creation
  – Communicator Cache

OMB has OpenSHMEM benchmarks already

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

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

Timeline of Pure comm

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

shmemx_broadcast32_nb()

While elapsed time < Compute

shmemx_test_req()

shmemx_wait_req()
### Blocking vs. Non-Blocking

**Reduce@512 proc.**

![Graph showing latency vs. message size for blocking and non-blocking]

**Medium Message**

- Non-Blocking Overall
- Non-Blocking Pure Comm.

**Small Message**

- Non-Blocking Overall
- Non-Blocking Pure Comm.

**NBC Overhead : Reduce**

- Similar trends with other collectives

**NBC Overhead : Broadcast**

- Almost zero overhead
- Slight overhead

### 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 optimise 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/