MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

NETWORK-BASED COMPUTING LABORATORY

Enabling Performance Efficient Runtime Support for Hybrid MPI+UPC++ Programming Models

Jahanzeb M. Hashmi, Khaled Hamidouche,

Dhabaleswar K. (DK) Panda

Network-Based Computing Laboratory

Department of Computer Science and Engineering

The Ohio State University, USA

Outline

- Introduction
- Motivation and Challenges
- Contributions
- Proposed Design
- Evaluation
- Conclusion and Future Work

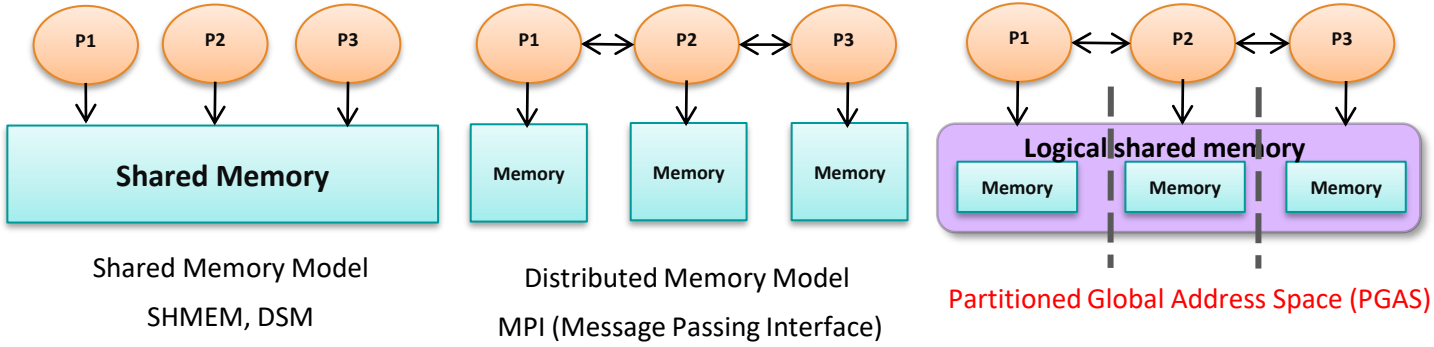
Outline

- Introduction
- Motivation and Challenges
- Contributions
- Proposed Design
- Evaluation
- Conclusion and Future Work

Introduction

- MPI is the de-facto programming model for scientific parallel applications
  - Offers attractive features for High Performance Computing (HPC) applications
  - MPI Libraries (such as MVAPICH2, Open MPI, Intel MPI) have been optimized to the hilt
- Partitioned Global Address Space (PGAS) models are Emerging
  - Global view of data, One sided operations, better programmability
  - Well suited for irregular and dynamic applications
- Hybrid MPI+PGAS approach
  - A new direction for parallel application design aimed at Exascale

Partitioned Global Address Space (PGAS) Models



- Key abstraction
  - Shared memory abstraction over distributed system images
- Library-level solutions
  - OpenSHMEM
  - Global Arrays
  - UPC++
- Language-level solutions
  - UPC
  - Coarray Fortran (CAF)
  - …

Unified Parallel C++ (UPC++): A Library Based PGAS Programming Model

- A new library based PGAS model using C++
  - C++11 features i.e., templates, lambda functions
  - Task based programming constructs
  - Asynchronous remote execution
- GASNet as the communication middleware
  - Several conduits for communication including MPI conduit (GASNet-MPI) and native InfiniBand (IB) conduit (GASNet-IBV)
- Supports hybrid MPI+UPC++ paradigm
  - Hybrid approach uses GASNet MPI conduit

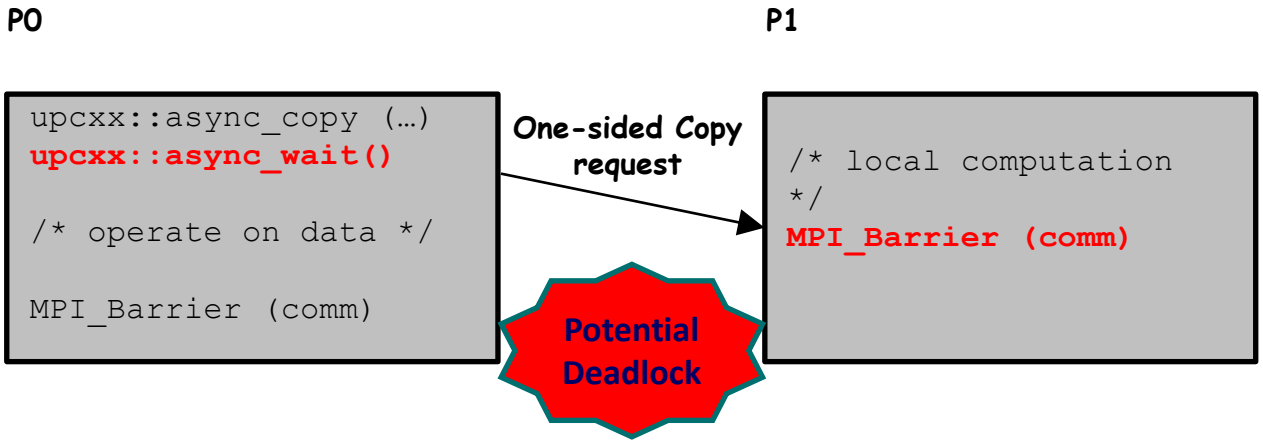
Outline

- Introduction
- Motivation and Challenges
- Contributions
- Proposed Design
- Evaluation
- Conclusion and Future Work

Motivation

- Performance limitations of GASNet based conduits for UPC++ communication
  - *MPI conduit* supports hybrid programming but has bad point-to-point performance
  - *IB Verbs conduit* shows native point-to-point performance but bad collectives performance.
    - Doesn't support hybrid MPI+UPC++ programming
- Hybrid MPI+UPC++ design using MPI conduit has potential for deadlocks
  - Lack of runtime resource consolidation

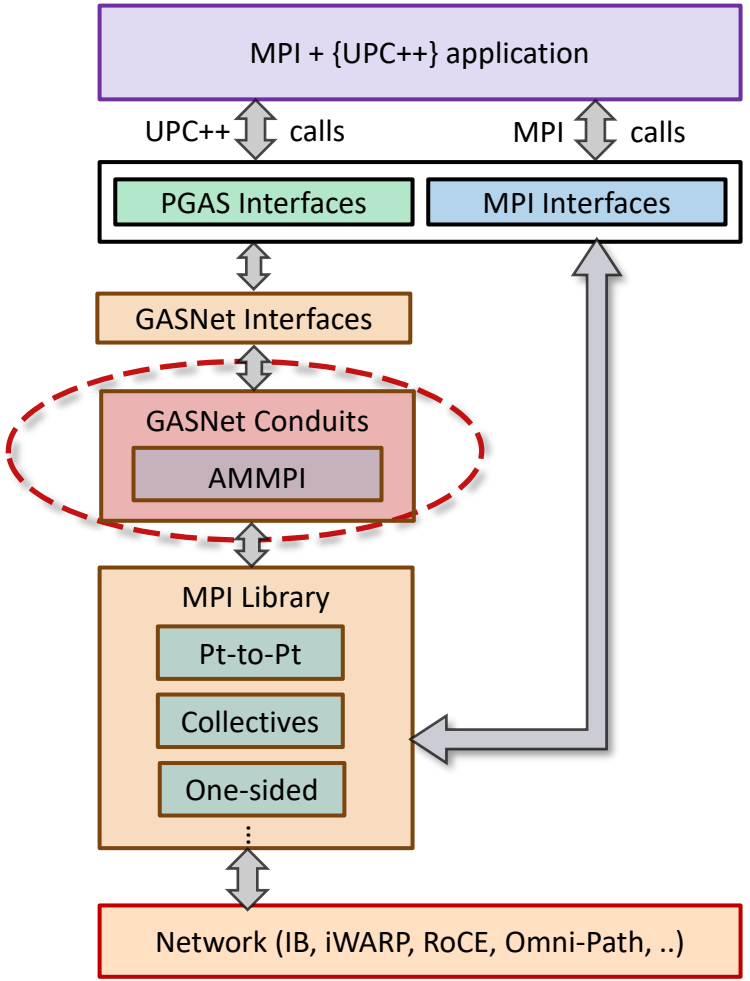
Example of a Deadlock Scenario



A potential deadlock scenario in hybrid MPI+UPC++ application design. `P0` waiting for `upcxx::async_wait` to finish while `P1` is waiting for `P0` to reach `MPI_Barrier`

Challenges

- Existing approaches uses GASNet based communication conduits
- The conduit based approach uses Active Message MPI (AMMPI)
  - Additional overhead
  - Unoptimized Collective algorithms using point-to-point operations
- How to avoid AMMPI overhead and achieve best Pt-to-pt and Collectives performance?



Problem Statement

Can we provide an efficient communication runtime support for pure UPC++ and hybrid MPI+UPC++ programming models that exploits modern features of UPC++ and MPI while achieving the best performance?

Outline

- Introduction
- Motivation and Challenges
- Contributions
- Proposed Design
- Evaluation
- Conclusion and Future Work

Contributions

- Efficient support for pure UPC++ and hybrid MPI+UPC++ models using MVAPICH2-X
- Performance *and* feature centric solution
  - Native pt-to-pt and collectives performance
  - Hybrid MPI+UPC++ supporting MPI3 Non-blocking Collectives (NBC)
- Proposed pure UPC++ microbenchmarks
  - upcxx_{Bcast, alltoall, allgather, scatter, reduce, gather}
  - Available since OSU Microbenchmark (OMB) v5.3
- Re-designing scientific applications for hybrid MPI+UPC++ to demonstrate the benefits
  - 2D Heat equation using Gauss-seidel kernel
  - LULESH2.0: 3-D shock hydrodynamic simulation

Outline

- Introduction
- Motivation and Challenges
- Contributions
- Proposed Design
- Evaluation
- Conclusion and Future Work

Overview of the MVAPICH2 Project

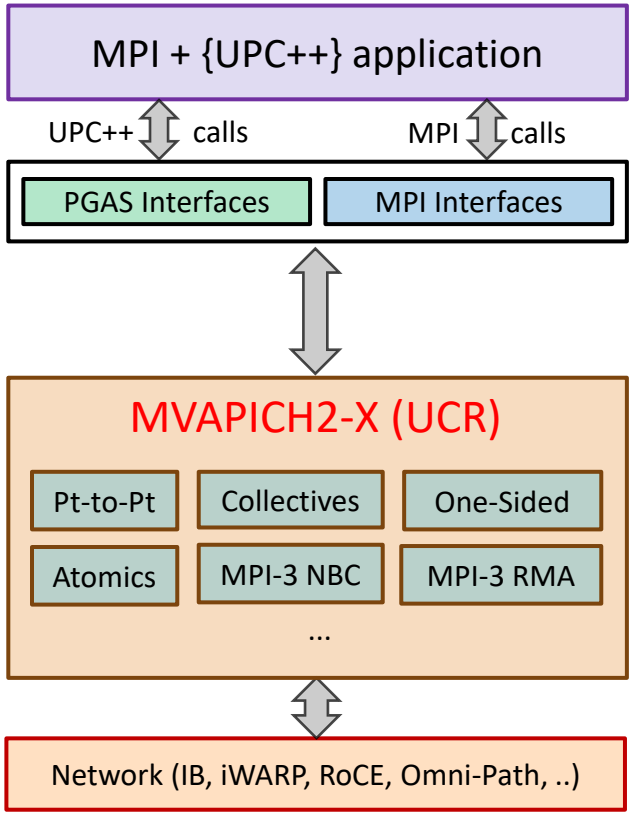
- High Performance open-source MPI Library for InfiniBand, Omni-Path, Ethernet/iWARP, and RDMA over Converged Ethernet (RoCE)
  - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.0), Started in 2001, First version available in 2002
  - **MVAPICH2-X (MPI + PGAS), Available since 2011**
  - Support for GPGPUs (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
  - Support for InfiniBand Network Analysis and Monitoring (OSU INAM) since 2015
  - **Used by more than 2,700 organizations in 83 countries**
  - **More than 404,000 (> 0.4 million) downloads from the OSU site directly**
  - Empowering many TOP500 clusters (Nov '16 ranking)
    - **1st ranked 10,649,640-core cluster (Sunway TaihuLight) at NSC, Wuxi, China**
    - 13th ranked 241,108-core cluster (Pleiades) at NASA
    - 17th ranked 519,640-core cluster (Stampede) at TACC
    - 40th 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) -> Sunway TaihuLight at NSC, Wuxi, China (1st in Nov'16, 10,649,640 cores, 93 PFlops)

Unified Communication Runtime (UCR) in MVAPICH2-X

- The common communication layer in MVAPICH2-X
  - Lower-level but easy-to-use primitives supporting the common one-sided, two-sided, collective communication and synchronization semantics
  - Optimization for collective communication, shared memory communication, etc.
- UCR-based MVAPICH2-X Conduit for GASNet
  - A complete implementation of GASNet core APIs as well as collective extended APIs
  - Has supported the UPC, OpenSHMEM, and CAF implementation in MVAPICH2-X

The Overview of Proposed Design

- Unified initialization of MPI and UPC++ runtimes
- Consolidation of runtime resources for MPI and UPC++ using UCR to avoid deadlocks
- UPC++ asynchronous data-movement operations
  - Upcxx::async_copy to ucr_put and ucr_get one-sided operations
  - Avoid overheads and deliver native IB performance

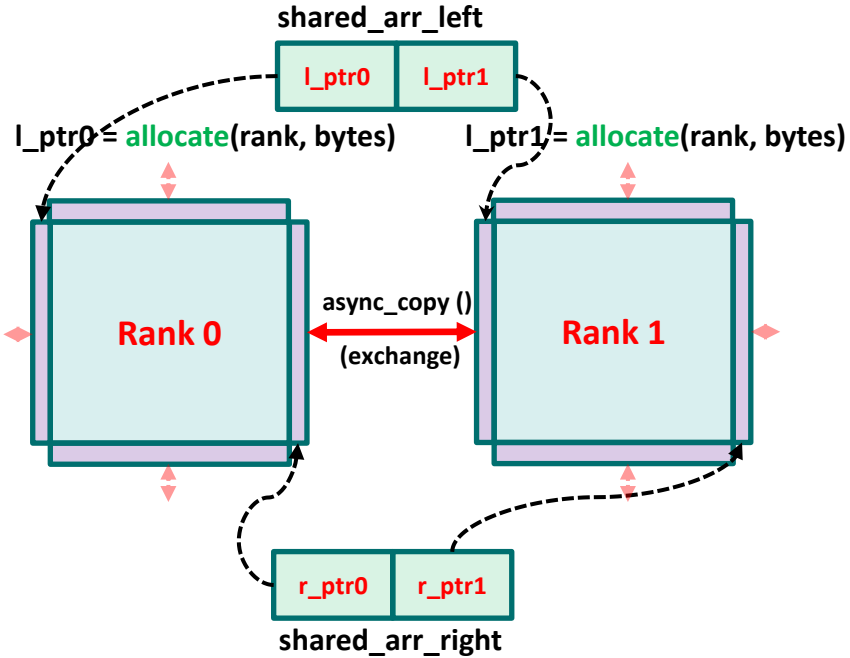


Active Message Events in UCR

- Direct mapping of UPC++ progression to UCR progress
  - Upcxx::advance() to ucr_probe etc…
- Introduced support for internal active message events in UCR using RDMA semantics
  - Results in native IB performance
- Mapping UPC++ active message events to UCR events
  - Alleviate active message based progression overheads
- Achieve efficient asynchronous remote method execution

Application Re-design (UPC++ 2D-Heat)

- UPC++ implementation of 2D heat conduction using Gauss-seidal kernel
- Near-neighbor communication with adjacent exchanges
- Shared arrays of global_ptrs for each dimension containing local data for exchange
- Asynchronous data transfers overlapped with computation



*(similarly for top and bottom)

```
shared_array<global_ptr<float>,1> shared_arr_left  (ranks())
shared_array<global_ptr<float>,1> shared_arr_right (ranks())
```

Application Re-design with MPI-3 NBC (LULESH2.0)

- LULESH profiling shows about 10-14% of application time is spent in collective communication phase (all_reduce)
- Re-design LULESH for hybrid MPI+UPC++ with non-blocking collective to achieve overlap
- Initial time-step calculation happens outside the convergence loop
- After state variables are updated, compute next time-step and issue a non-blocking collective (NBC) operation
- MPI_Iallgather followed by local reduction to mimic MPI_allreduce
- The NBC is offloaded to network adapter using CORE-Direct feature provided by Mellanox

Outline

- Introduction
- Motivation and Challenges
- Contributions
- Proposed Design
- Evaluation
- Conclusion and Future Work

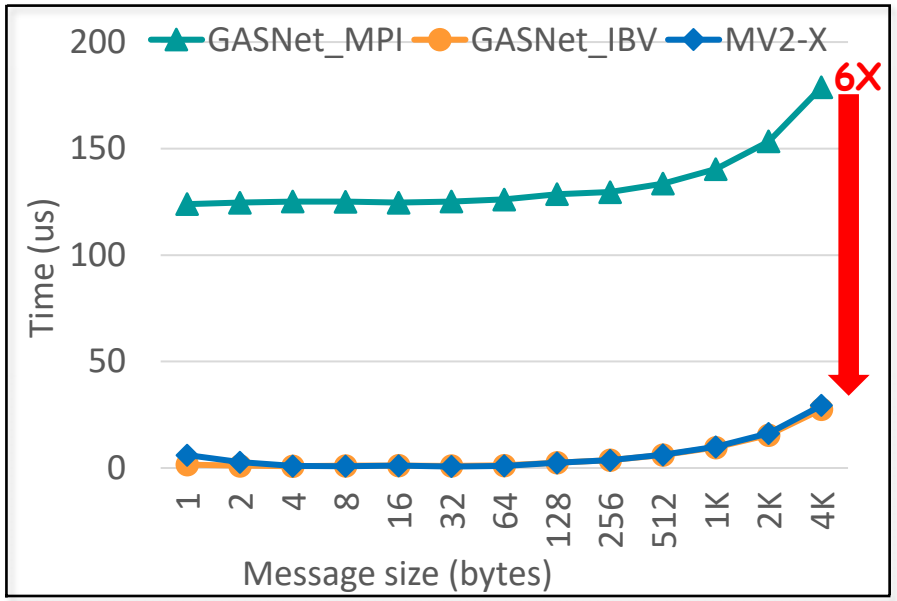
Evaluation Methodology

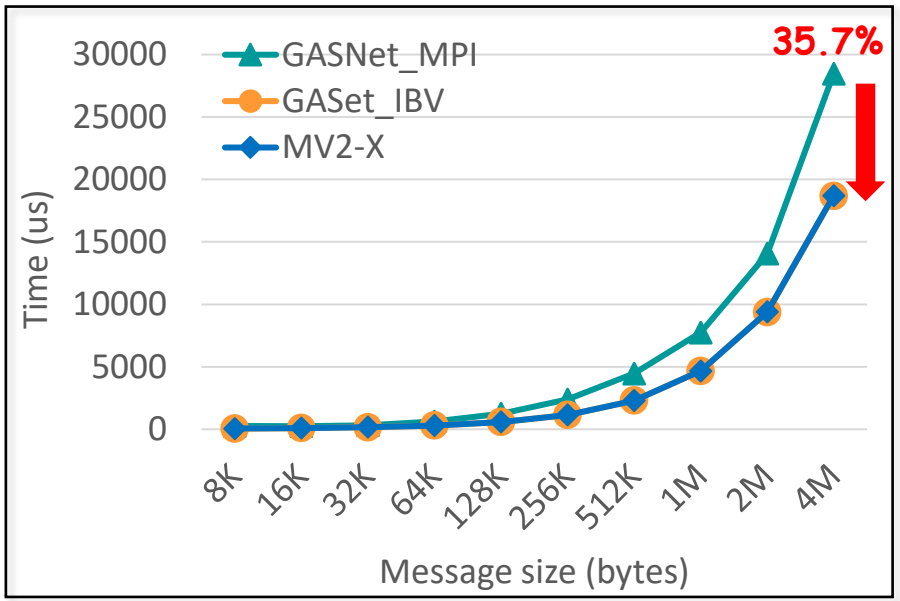
- Microbenchmark Evaluation
  - Proposed UPC++ microbenchmarks in OMB
  - Comparison with existing runtimes e.g., GASNet-MPI and GASNet-IBV
- Pure UPC++ application evaluation
  - Own implementation of 2D-Heat equation in UPC++
  - Comparison of proposed design with GASNet-MPI and GASNet-IBV
- Hybrid MPI+UPC++ with NBC
  - Redesigned LULESH2.0 application for communication overlap
  - Demonstrate the ‘extra’ benefits achieved with overlap
  - Comparison of ‘Pure MPI’, ‘Pure UPC++’, ‘Hybrid MPI+UPC++’, and Hybrid with NBC
  - All implementations use MVAPICH2-X as the communication runtime

Experiment Setup

- RI Cluster @ CSE, OSU (128 nodes)
  - Xeon dual 8 core sockets (2.67GHz) with 12GB RAM
  - Mellanox QDR Connect-X HCAs (32 Gbps data rate)
- RI2 Cluster @ CSE, OSU (40 nodes)
  - Xeon E5-2680 v4 (2.40 GHz) with 128 GB RAM
  - Mellanox EDR Connect-X HCAs (100 Gbps data rate) with CORE-Direct offload support
- Software stack
  - RHEL 6.3 with Mellanox OFED v2.2-1.0.0
  - MVAPICH2-X 2.2rc1

Microbenchmark Evaluations (Point-to-Point)

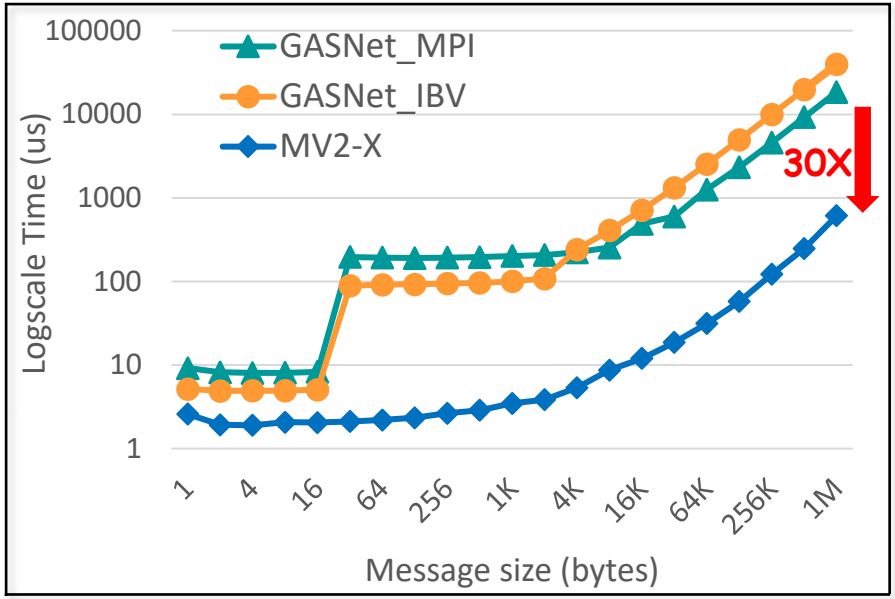




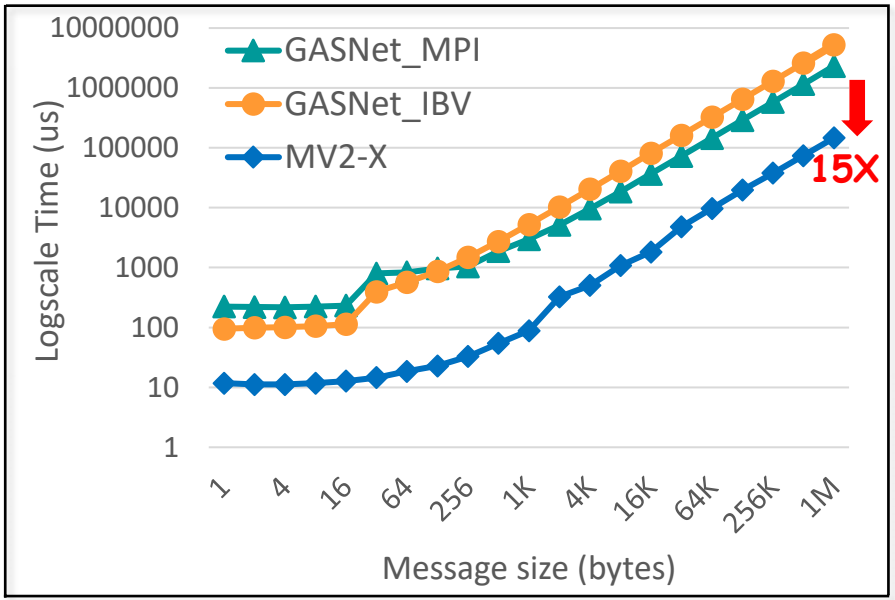
UPC++ async_copy (put) on 128 processes small and large message ranges

- UCR design achieves native InfiniBand (IB) level point-to-point performance for small and large message sizes

Microbenchmark Evaluations (Collectives)



UPC++ broadcast on 128 processes

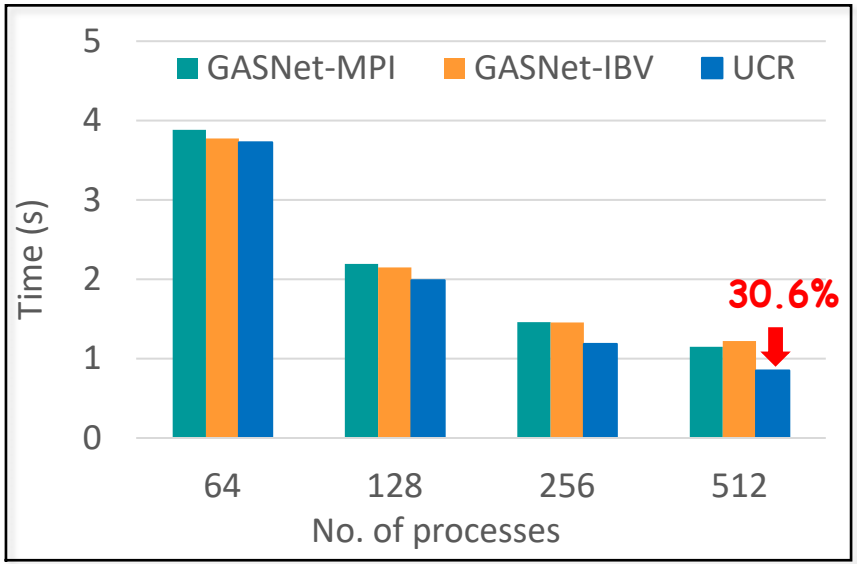


UPC++ allgather on 128 processes

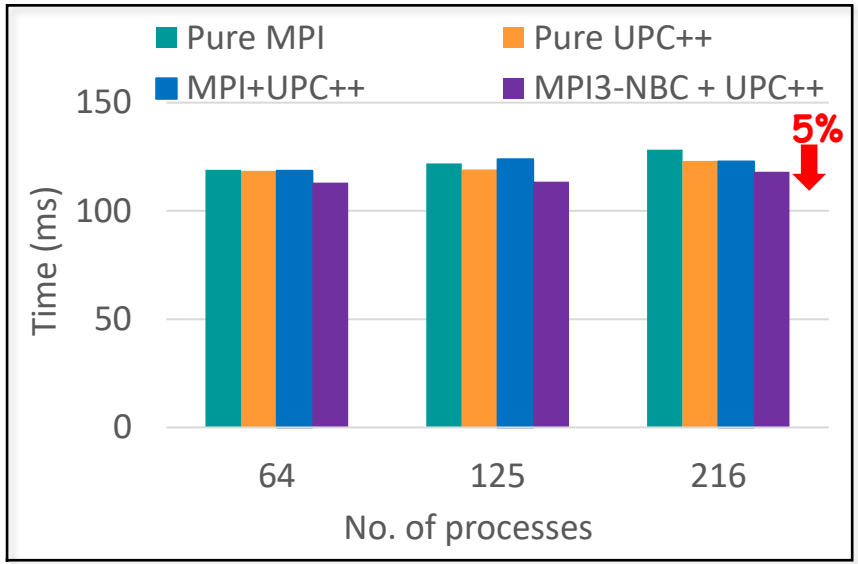
- Better performance in collective operations for all message ranges
- 30X and 15X improvement over GASNet-MPI conduit in *bcast* and *allgather* collectives respectively on large message sizes

Application Level Evaluations

UPC++ 2D-Heat equation using Gauss-seidel kernel on 512x512 matrix



Total time of one iteration of LULESH2.0 on different implementations (all using UCR design)



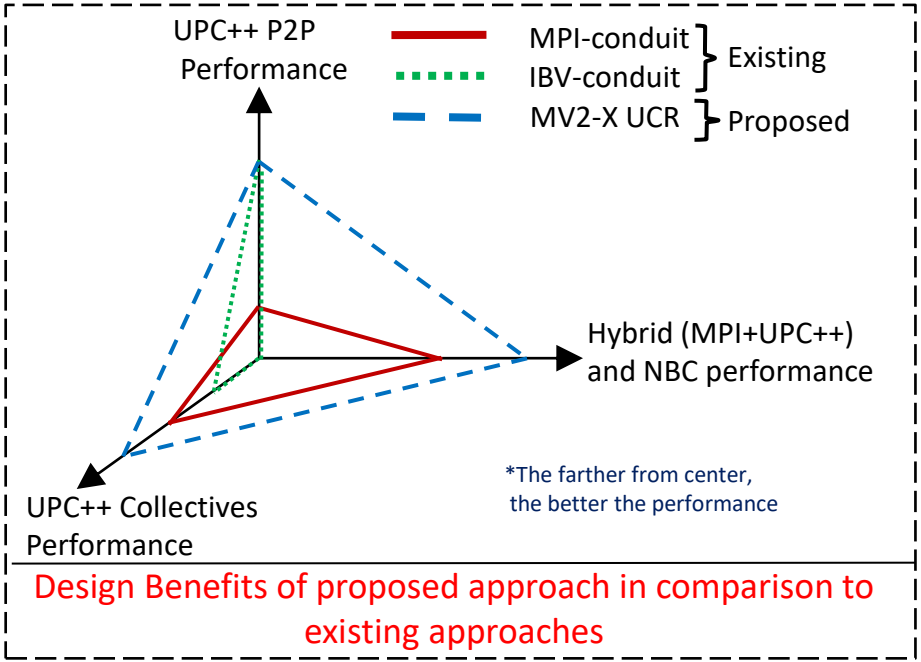
- 30.6% improvement over GASNet-IBV for Gauss-seidel kernel
- 5% improvement per iteration of LULESH2.0 due to MPI3 NBC using CORE-Direct (MVAPICH2-X UCR is used for all the implementations)

Outline

- Introduction
- Motivation and Challenges
- Contributions
- Proposed Design
- Evaluation
- Conclusion and Future Work

Conclusion

- A lightweight, modular design for unified runtime support in UPC++
- Exploits full features of hybrid MPI+UPC++ programming models with best performance
- Resource consolidation of runtimes in hybrid MPI+UPC++ programming to avoid deadlock scenarios
- Up to 6X and 30X improvement in Pt-to-pt and collective benchmarks, respectively



Design Benefits of proposed approach in comparison to existing approaches

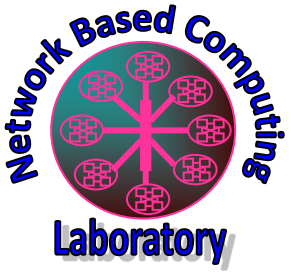
- Up to 30% improvement in 2D-Heat equation using Gauss-seidel kernel
- Redesigned hybrid LULESH2.0 with MPI-3 NBC and CORE-Direct offload shows up to 5% improvement in total execution time

Release and Future Work

- The proposed design and UPC++ microbenchmarks have been publicly released and are available in MVAPICH2-X software package
  - Can be downloaded from http://mvapich.cse.ohio-state.edu
- The next steps for advanced support of UPC++ in MVAPICH2-X:
  - Further optimize the collective communication subroutines
  - Enable the upcoming team-level collective communication semantics
  - Add efficient support for the upcoming remote atomic operations in UPC++
- Optimizing applications for hybrid MPI+UPC++ programming models to achieve better overlap

Thank You!

{hashmi.29, hamidouche.2, panda.2}@osu.edu



Network-Based Computing Laboratory

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

MVAPICH Web Page

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