MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

Overview of the MVAPICH Project: Latest Status and Future Roadmap

MVAPICH2 User Group (MUG) Meeting

by

Dhabaleswar K. (DK) Panda

The Ohio State University

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

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

Drivers of Modern HPC Cluster Architectures

Multi-core Processors

High Performance Interconnects - InfiniBand
<1usec latency, 100Gbps Bandwidth>

Accelerators / Coprocessors
high compute density, high performance/watt
>1 TFlop DP on a chip

SSD, NVMe-SSD, NVRAM

- Multi-core/many-core technologies
- Remote Direct Memory Access (RDMA)-enabled networking (InfiniBand and RoCE)
- Solid State Drives (SSDs), Non-Volatile Random-Access Memory (NVRAM), NVMe-SSD
- Accelerators (NVIDIA GPGPUs and Intel Xeon Phi)

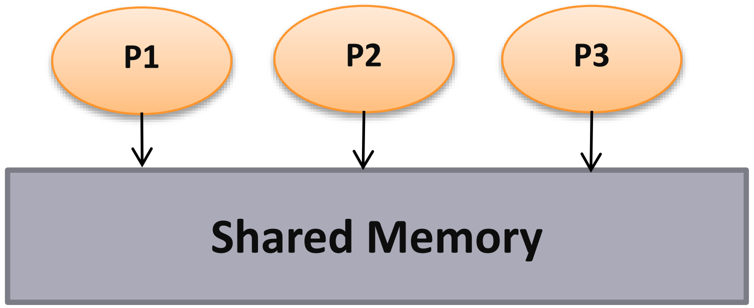
Tianhe – 2

Titan

Stampede

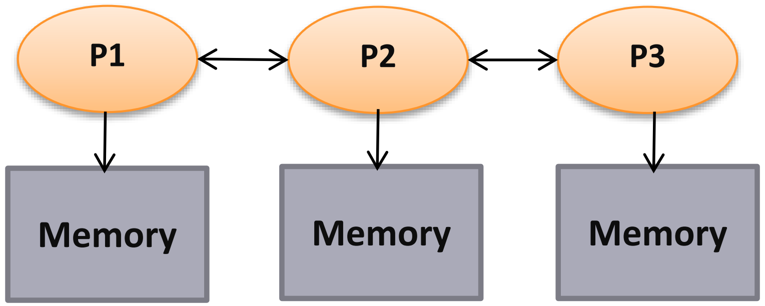
Tianhe – 1A

Parallel Programming Models Overview



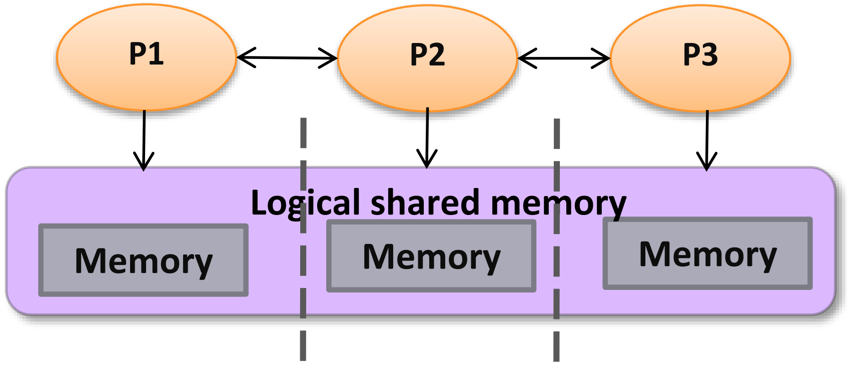
Shared Memory Model

SHMEM, DSM



Distributed Memory Model

MPI (Message Passing Interface)

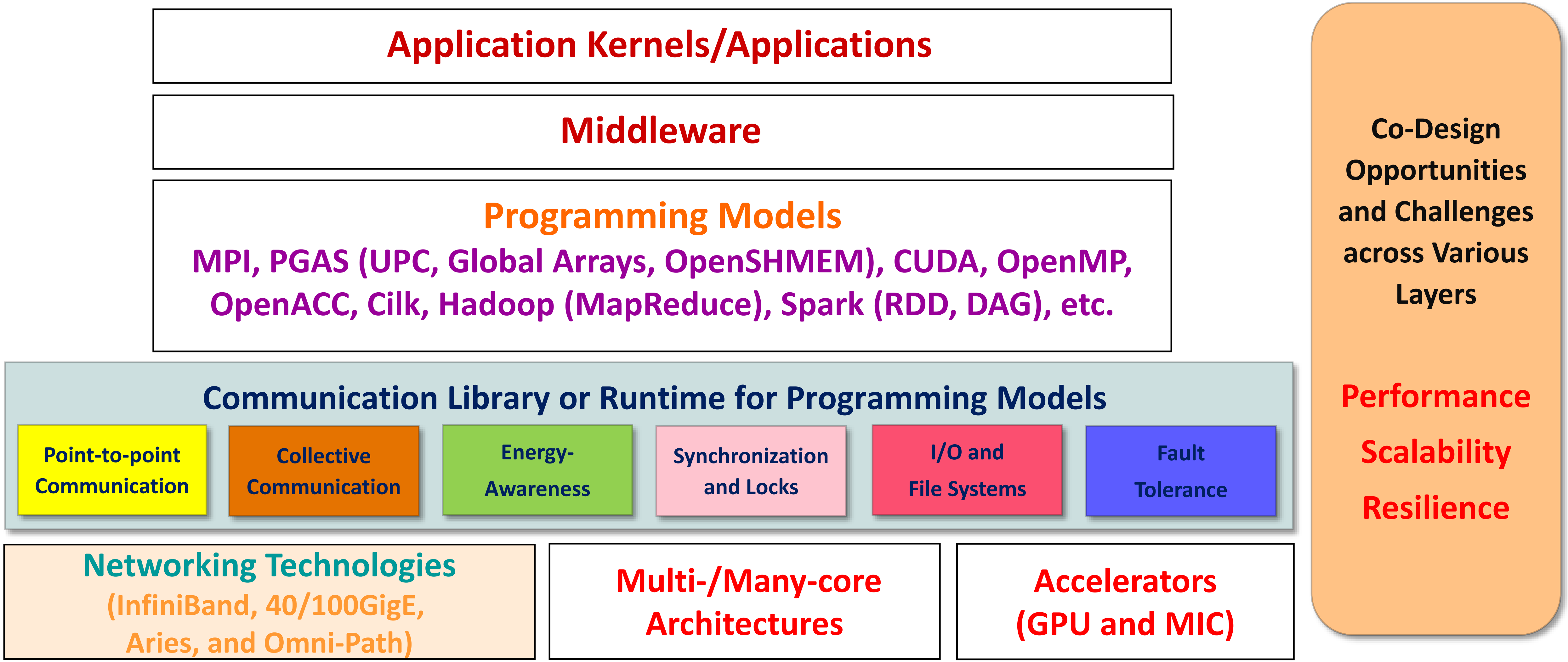


Partitioned Global Address Space (PGAS)

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

Supporting Programming Models for Multi-Petaflop and Exaflop Systems: Challenges



Application Kernels/Applications

Middleware

Programming Models

MPI, PGAS (UPC, Global Arrays, OpenSHMEM), CUDA, OpenMP, OpenACC, Cilk, Hadoop (MapReduce), Spark (RDD, DAG), etc.

Communication Library or Runtime for Programming Models

Point-to-point Communication

Collective Communication

Energy-Awareness

Synchronization and Locks

I/O and File Systems

Fault Tolerance

Networking Technologies (InfiniBand, 40/100GigE, Aries, and Omni-Path)

Multi-/Many-core Architectures

Accelerators (GPU and MIC)

Co-Design Opportunities and Challenges across Various Layers

Performance

Scalability

Resilience

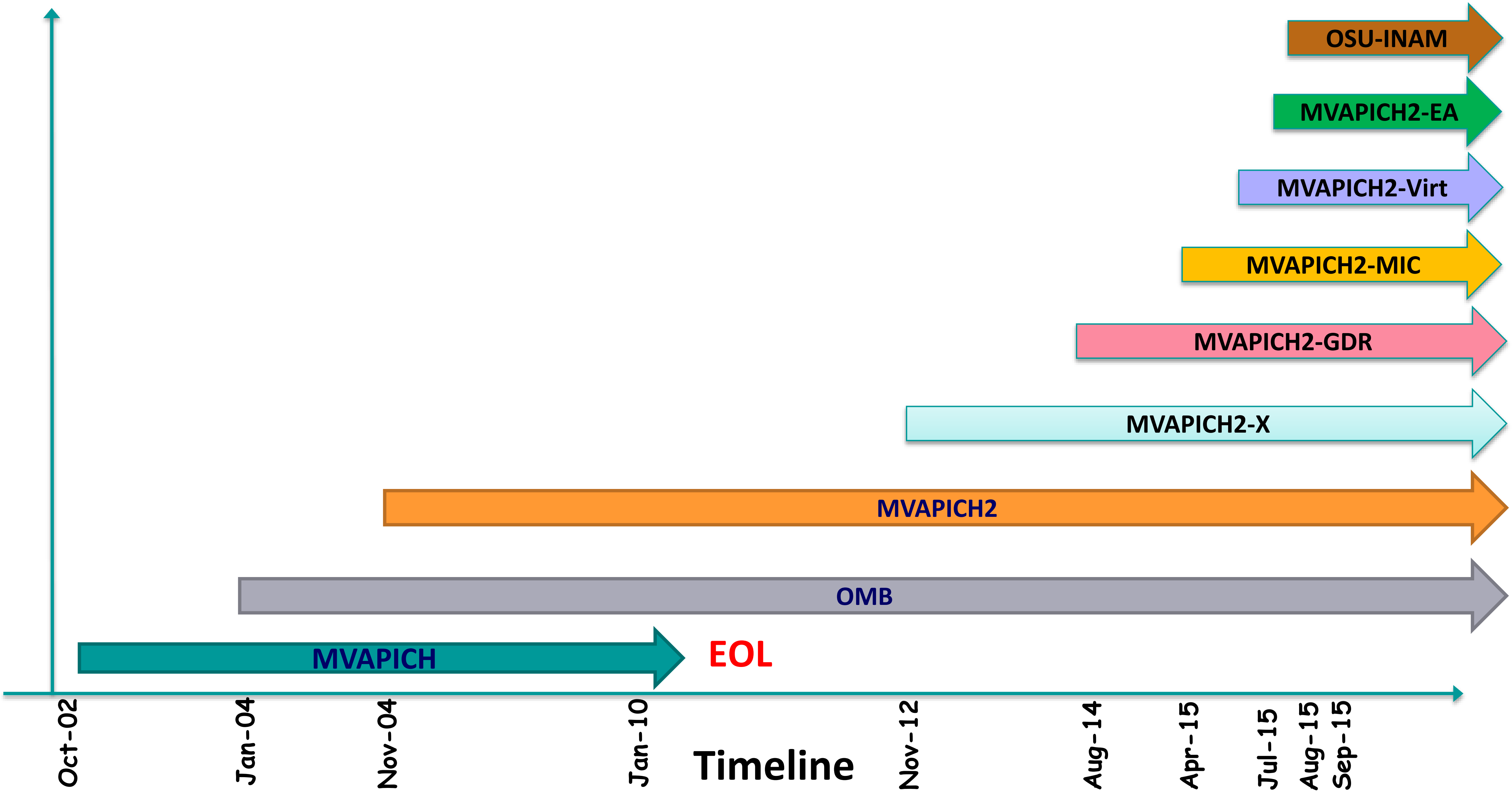
Designing (MPI+X) for Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
- Scalable Collective communication
  - Offloaded
  - Non-blocking
  - Topology-aware
- Balancing intra-node and inter-node communication for next generation multi-/many-core (128-1024 cores/node)
  - Multiple end-points per node
- Support for efficient multi-threading
- Integrated Support for GPGPUs and Accelerators
- Fault-tolerance/resiliency
- QoS support for communication and I/O
- Support for Hybrid MPI+PGAS programming
  - MPI + OpenMP, MPI + UPC, MPI + OpenSHMEM, CAF, MPI + UPC++…
- Virtualization
- Energy-Awareness

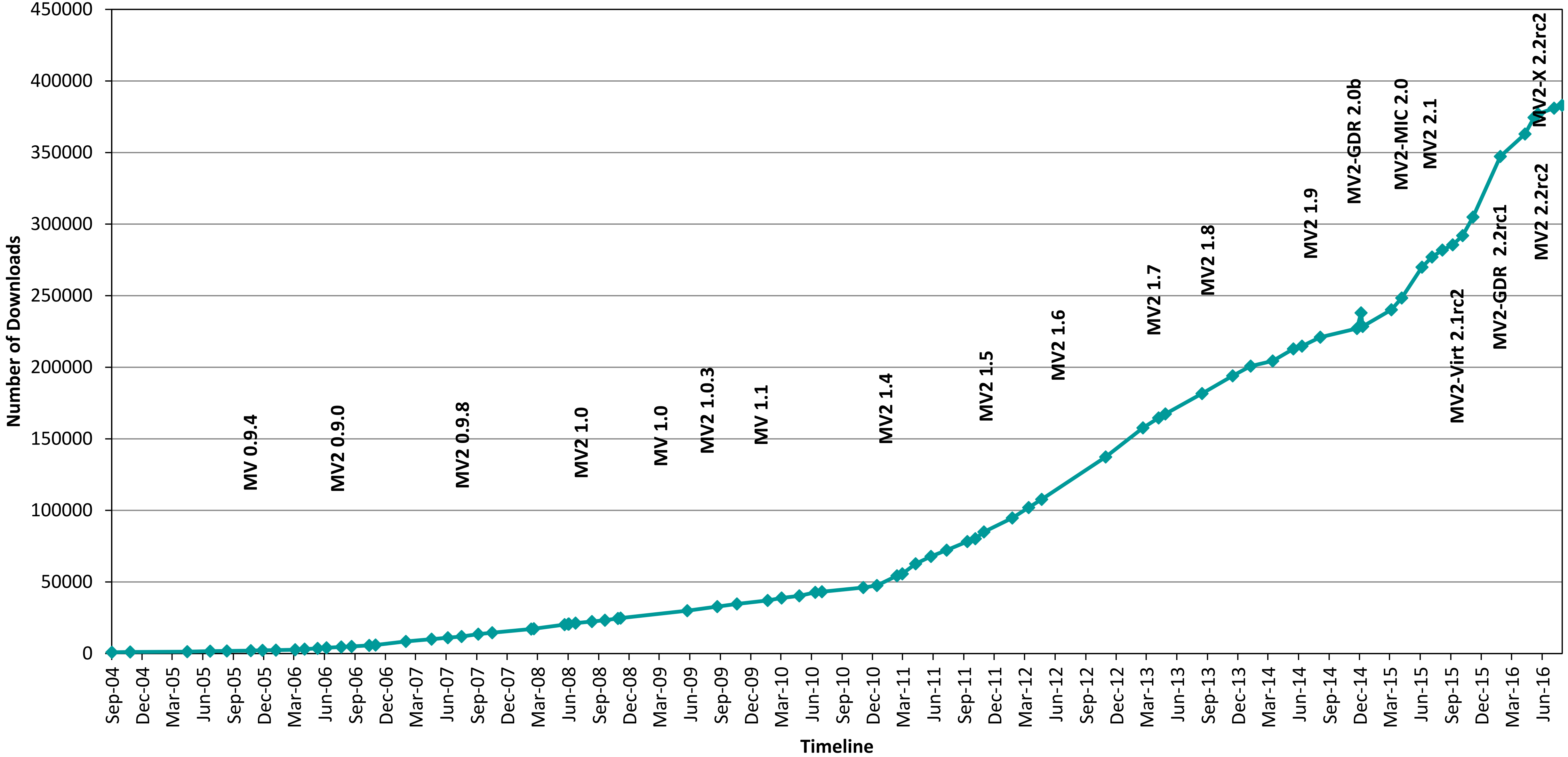
Overview of the MVAPICH2 Project

- 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 2,650 organizations in 81 countries**
  - **More than 383,000 (> 0.38 million) downloads from the OSU site directly**
  - Empowering many TOP500 clusters (Jun '16 ranking)
    - 12th ranked 519,640-core cluster (Stampede) at TACC
    - 15th ranked 185,344-core cluster (Pleiades) at NASA
    - 31st 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 (12th in Jun'16, 462,462 cores, 5.168 Plops)

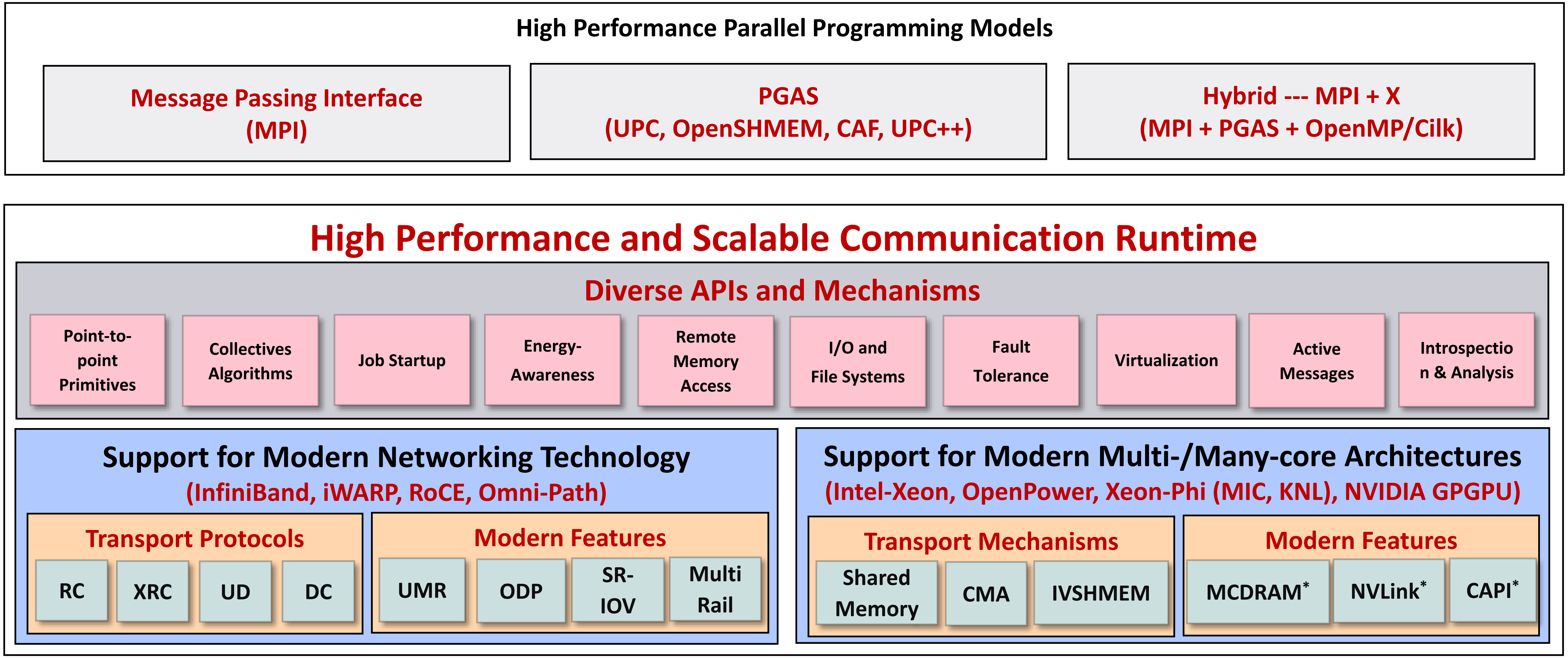
MVAPICH Project Timeline



MVAPICH/MVAPICH2 Release Timeline and Downloads



Architecture of MVAPICH2 Software Family



High Performance Parallel Programming Models

Message Passing Interface (MPI)

PGAS (UPC, OpenSHMEM, CAF, UPC++)

Hybrid --- MPI + X (MPI + PGAS + OpenMP/Cilk)

High Performance and Scalable Communication Runtime

Diverse APIs and Mechanisms

Point-to-point Primitives

Collectives Algorithms

Job Startup

Energy-Awareness

Remote Memory Access

I/O and File Systems

Fault Tolerance

Virtualization

Active Messages

Introspection & Analysis

Support for Modern Networking Technology (InfiniBand, iWARP, RoCE, Omni-Path)

Transport Protocols

RC

XRC

UD

DC

Modern Features

UMR

ODP

SR-IOV

Multi Rail

Support for Modern Multi-/Many-core Architectures (Intel-Xeon, OpenPower, Xeon-Phi (MIC, KNL), NVIDIA GPGPU)

Transport Mechanisms

Shared Memory

CMA

IVSHMEM

Modern Features

MCDRAM*

NVLink*

CAPI*

* Upcoming

Strong Procedure for Design, Development and Release

- Research is done for exploring new designs
- Designs are first presented to conference/journal publications
- Best performing designs are incorporated into the codebase
- Rigorous Q&A procedure before making a release
  - Exhaustive unit testing
  - Various test procedures on diverse range of platforms and interconnects
  - Performance tuning
  - Applications-based evaluation
  - Evaluation on large-scale systems
- Even alpha and beta versions go through the above testing

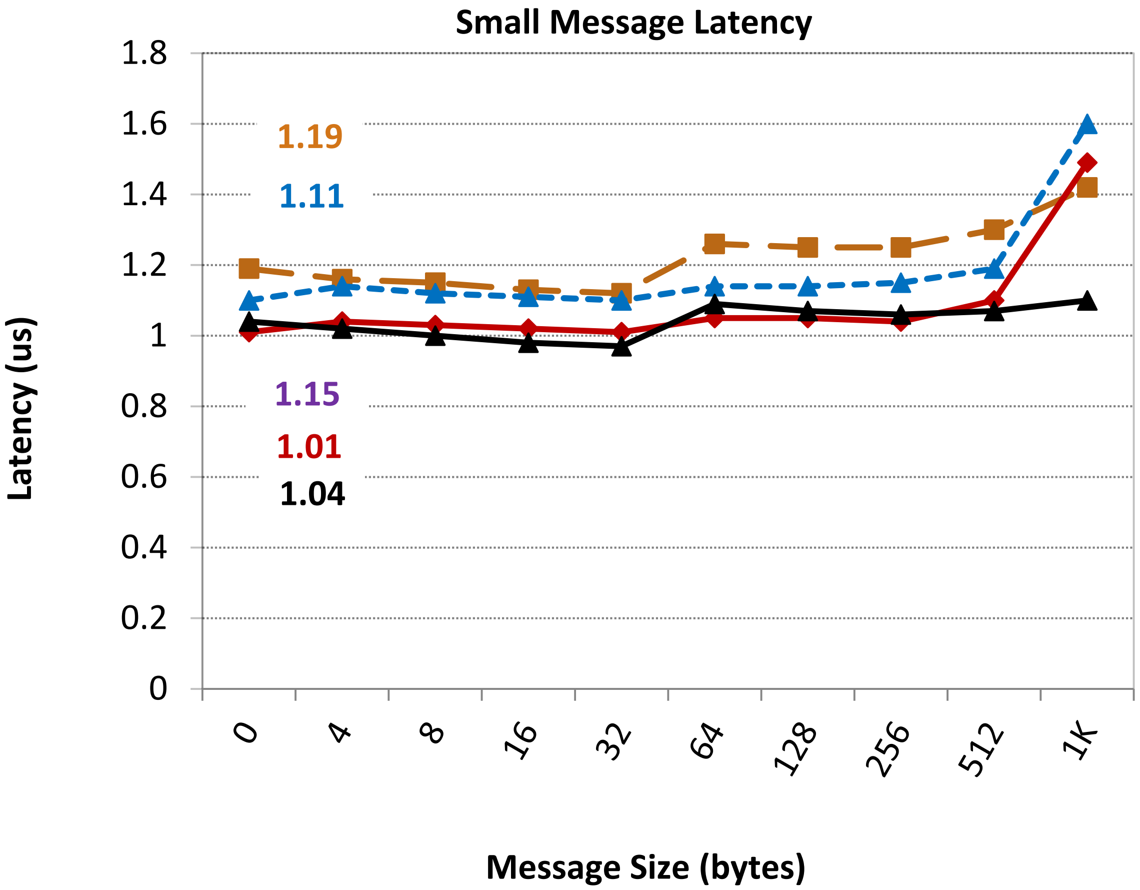
MVAPICH2 Software Family

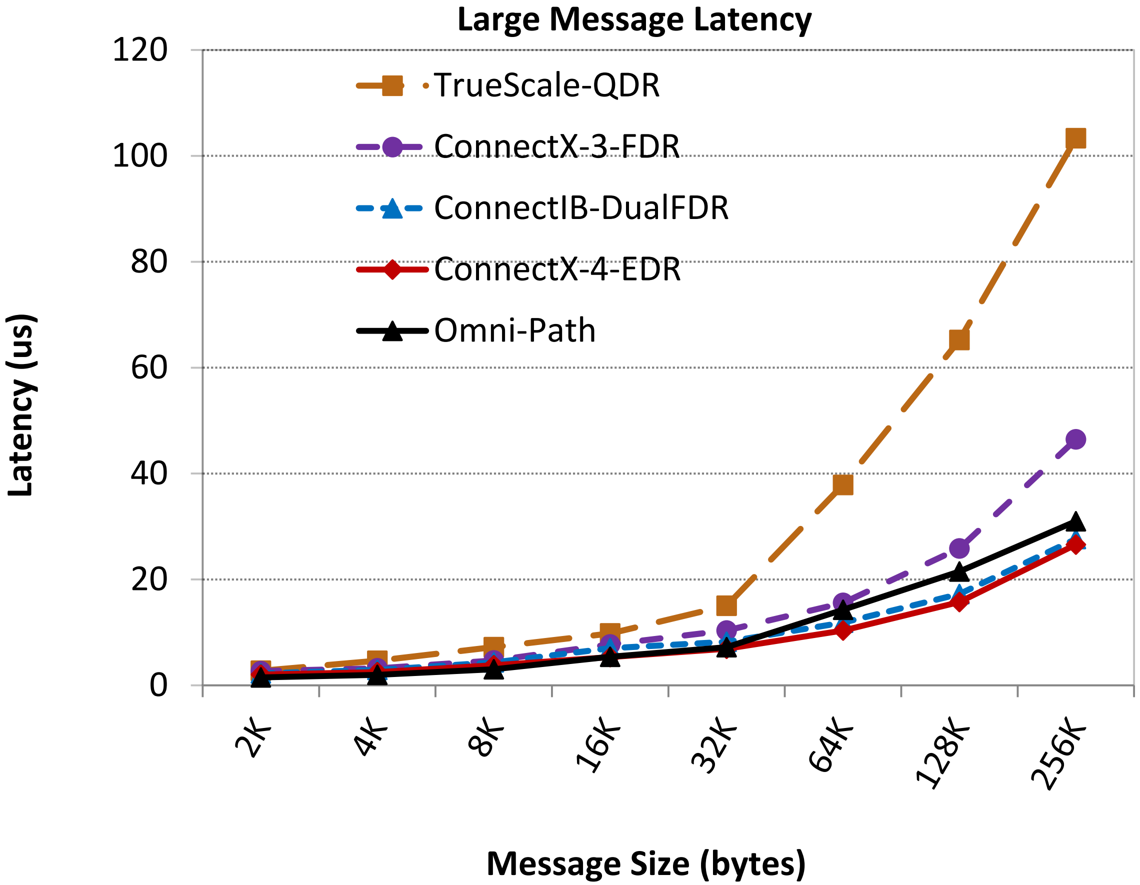
Requirements	Library
MPI with IB, iWARP and RoCE	MVAPICH2
Advanced MPI, OSU INAM, PGAS and MPI+PGAS with IB and RoCE	MVAPICH2-X
MPI with IB & GPU	MVAPICH2-GDR
MPI with IB & MIC	MVAPICH2-MIC
HPC Cloud with MPI & IB	MVAPICH2-Virt
Energy-aware MPI with IB, iWARP and RoCE	MVAPICH2-EA
MPI Energy Monitoring Tool	OEMT
InfiniBand Network Analysis and Monitoring	OSU INAM
Microbenchmarks for Measuring MPI and PGAS Performance	OMB

MVAPICH2 2.2rc2

- Released on 08/08/2016
- Major Features and Enhancements
  - Based on MPICH-3.1.4
  - Enhanced performance for MPI_Comm_split through new bitonic algorithm
  - Enable graceful fallback to Shared Memory if LiMIC2 or CMA transfer fails
  - Enable support for multiple MPI initializations
  - Remove verbs dependency when building the PSM and PSM2 channels
  - Allow processes to request MPI_THREAD_MULTIPLE when socket or NUMA node level affinity is specified
  - Point-to-point and collective performance optimization for Intel Knights Landing
  - Automatic detection and tuning for InfiniBand EDR HCAs
  - Collective tuning for Opal@LLNL, Bridges@PSC, and Stampede-1.5@TACC
  - Tuning and architecture detection for Intel Broadwell processors
  - Warn user to reconfigure library if rank type is not large enough to represent all ranks in job
  - Unify process affinity support in Gen2, PSM and PSM2 channels

One-way Latency: MPI over IB with MVAPICH2





TrueScale-QDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB switch

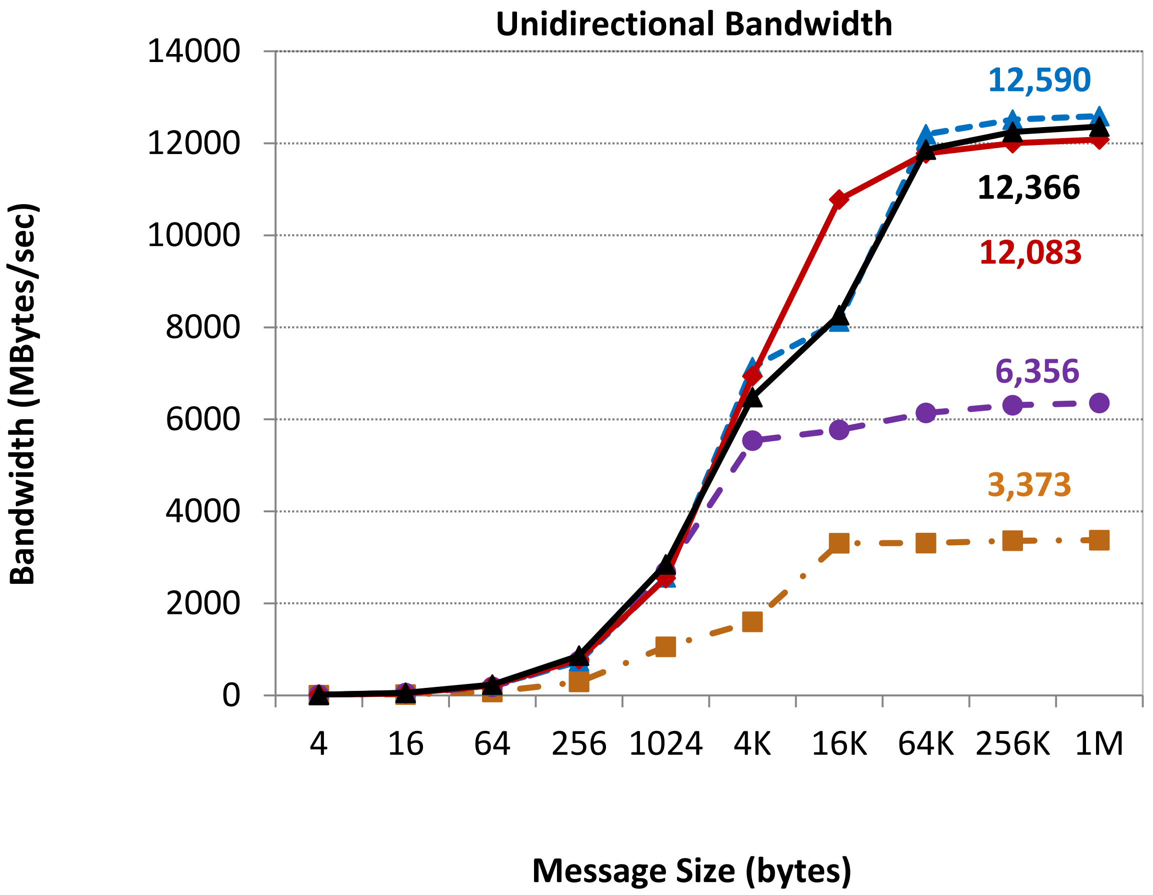
ConnectX-3-FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch

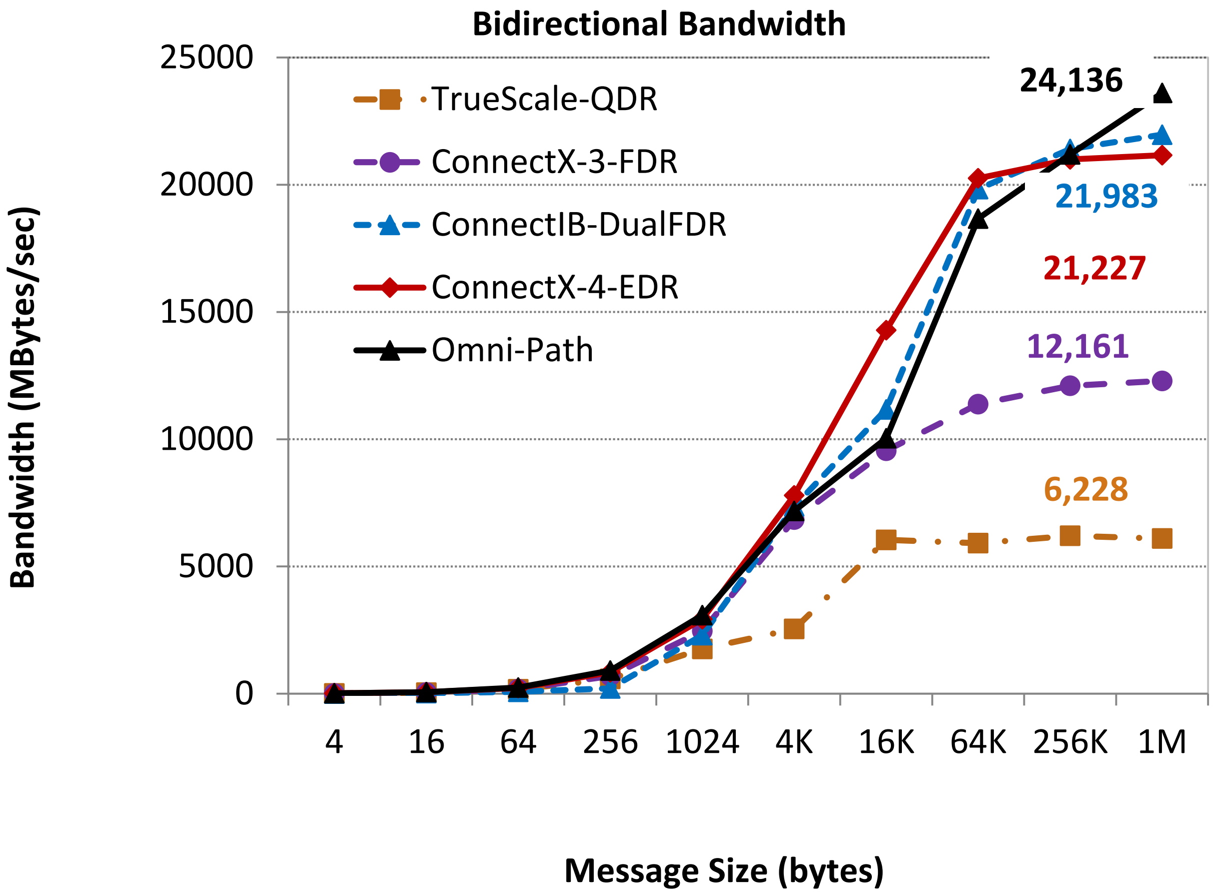
ConnectIB-Dual FDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB switch

ConnectX-4-EDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB Switch

Omni-Path - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with Omni-Path switch

Bandwidth: MPI over IB with MVAPICH2





TrueScale-QDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB switch

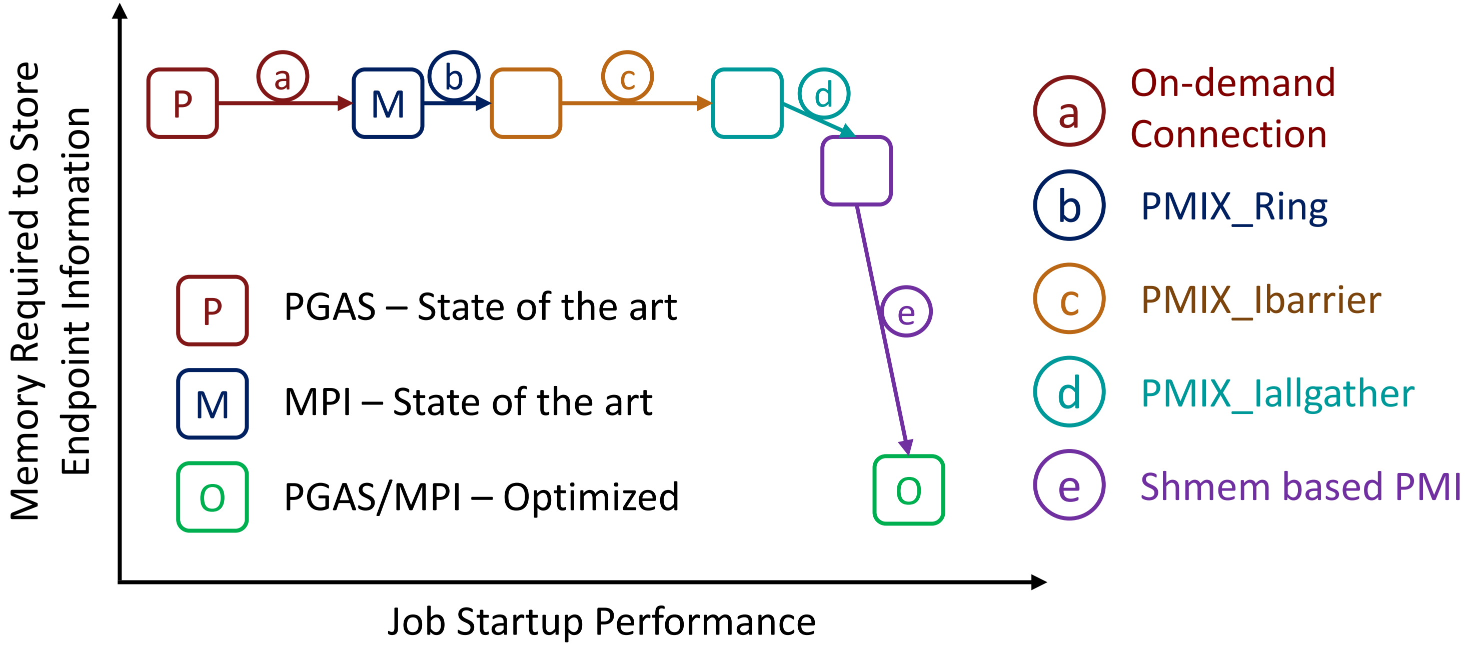
ConnectX-3-FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch

ConnectIB-Dual FDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB switch

ConnectX-4-EDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 IB switch

Omni-Path - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with Omni-Path switch

Towards High Performance and Scalable Startup at Exascale



- Near-constant MPI and OpenSHMEM initialization time at any process count
- 10x and 30x improvement in startup time of MPI and OpenSHMEM respectively at 16,384 processes
- Memory consumption reduced for remote endpoint information by O(processes per node)
- 1GB Memory saved per node with 1M processes and 16 processes per node

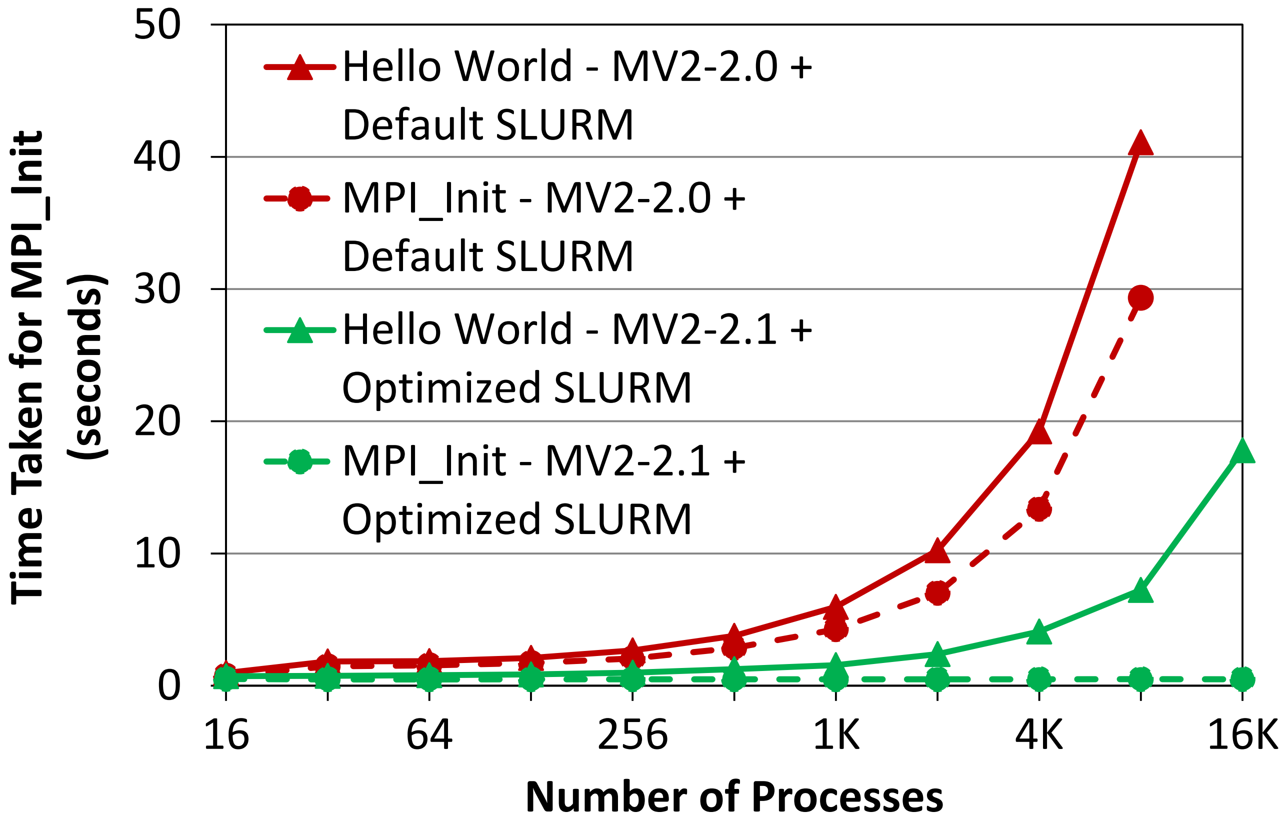
ⓐ **On-demand Connection Management for OpenSHMEM and OpenSHMEM+MPI.** S. Chakraborty, H. Subramoni, J. Perkins, A. A. Awan, and D K Panda, 20th International Workshop on High-level Parallel Programming Models and Supportive Environments (HIPS ’15)

ⓑ **PMI Extensions for Scalable MPI Startup.** S. Chakraborty, H. Subramoni, A. Moody, J. Perkins, M. Arnold, and D K Panda, Proceedings of the 21st European MPI Users' Group Meeting (EuroMPI/Asia ’14)

ⓒ ⓓ **Non-blocking PMI Extensions for Fast MPI Startup.** S. Chakraborty, H. Subramoni, A. Moody, A. Venkatesh, J. Perkins, and D K Panda, 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid ’15)

ⓔ **SHMEMPMI – Shared Memory based PMI for Improved Performance and Scalability.** S. Chakraborty, H. Subramoni, J. Perkins, and D K Panda, 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid ’16) , *Accepted for Publication*

Non-blocking Process Management Interface (PMI) Primitives for Scalable MPI Startup



- Address exchange over PMI is the major bottleneck in job startup
- Non-blocking PMI exchange hides this cost by overlapping it with application initialization and computation
- New PMI operation PMIX_Allgather for improved symmetric data transfer
- Near-constant MPI_Init at any scale
- MPI_Init is 59 times faster at 8,192 processes (512 nodes)
- Hello World (MPI_Init + MPI_Finalize) takes 5.7 times less time at 8,192 processes

Available since MVAPICH2-2.1 and as patch for SLURM-15.08.8 and SLURM-16.05.1

More Details in Student Poster Presentation

A new Initiative with LLNL about Avalaunch (Adam Moody’s presentation)

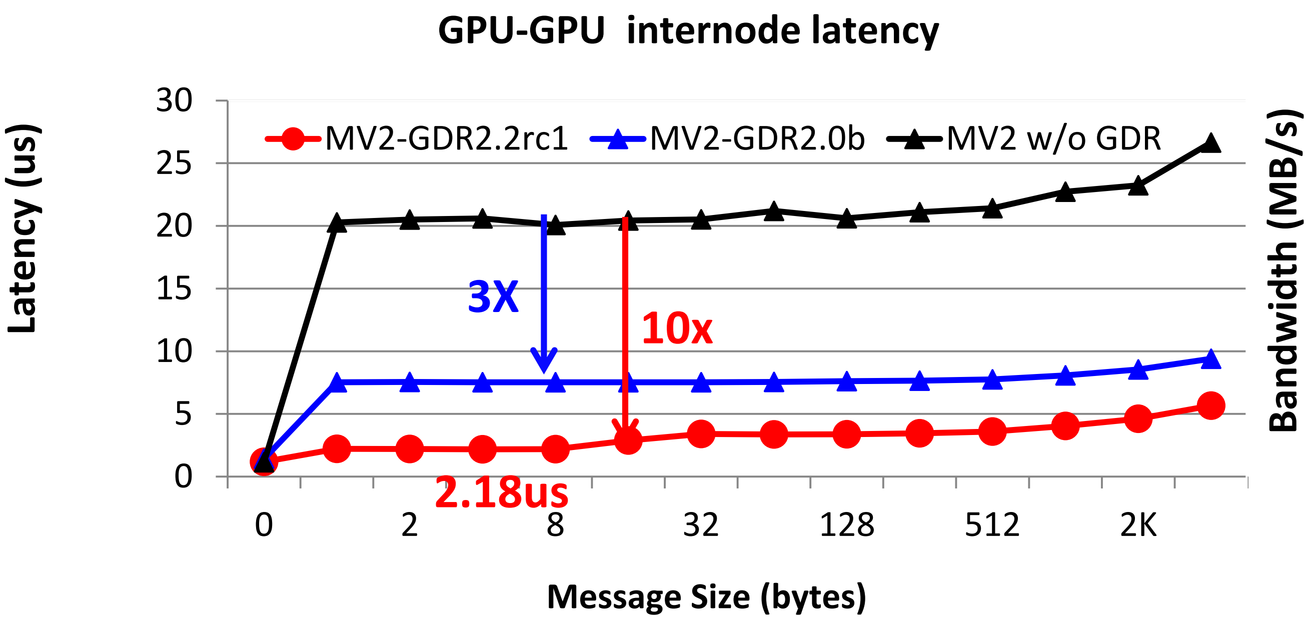
CUDA-Aware MPI: MVAPICH2-GDR 1.8-2.2 Releases

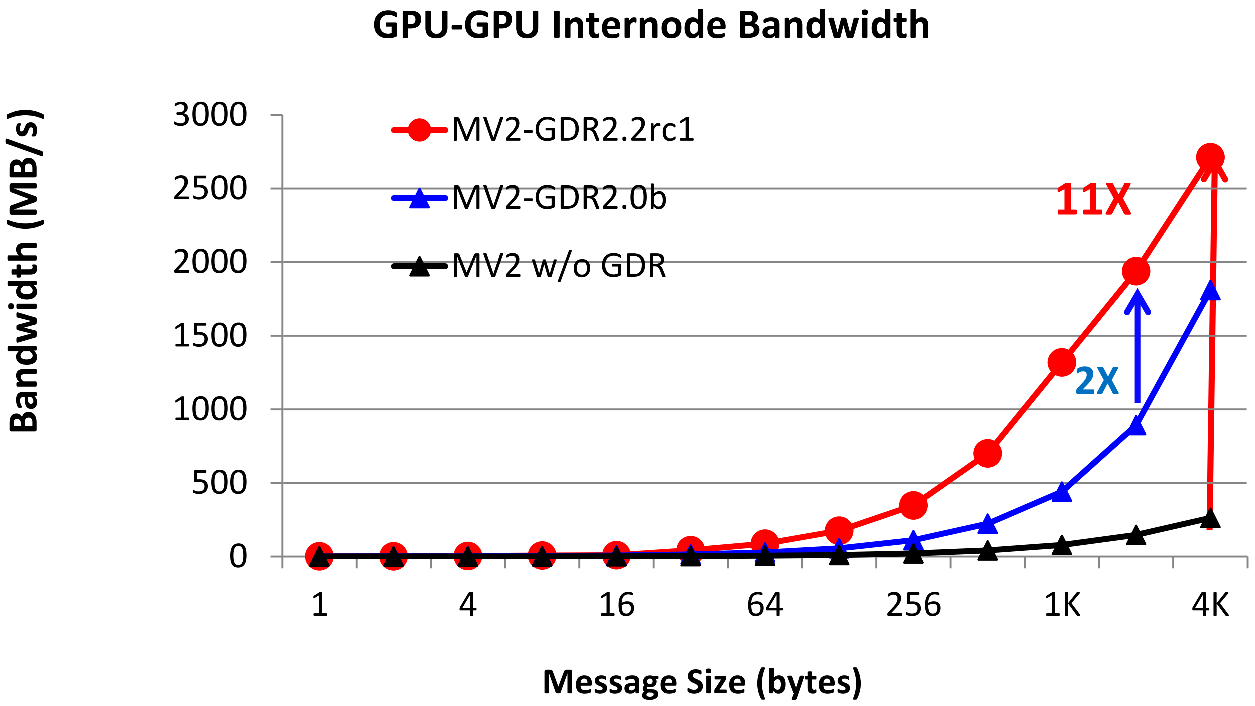
- Support for MPI communication from NVIDIA GPU device memory
- High performance RDMA-based inter-node point-to-point communication (GPU-GPU, GPU-Host and Host-GPU)
- High performance intra-node point-to-point communication for multi-GPU adapters/node (GPU-GPU, GPU-Host and Host-GPU)
- Taking advantage of CUDA IPC (available since CUDA 4.1) in intra-node communication for multiple GPU adapters/node
- Optimized and tuned collectives for GPU device buffers
- MPI datatype support for point-to-point and collective communication from GPU device buffers

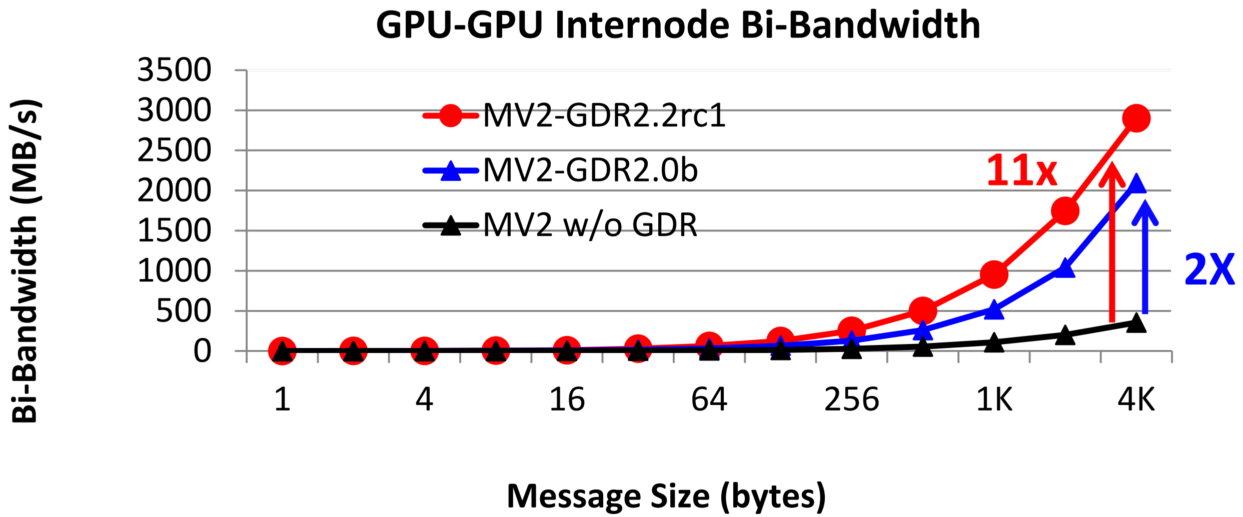
MVAPICH2-GDR 2.2rc1

- Released on 05/28/2016
- Major Features and Enhancements
  - Based on MPICH-3.1.4
  - Based on MVAPICH2 2.2rc1
  - Support for high-performance non-blocking send operations from GPU buffers
  - Enhancing Intra-node CUDA-Aware Managed Memory communication using a new CUDA-IPC-based design
  - Adding support for RDMA_CM communication
  - Introducing support for RoCE-V1 and RoCE-V2
  - Introducing GPU-based tuning framework for Bcast and Gather operations

Performance of MVAPICH2-GPU with GPU-Direct RDMA (GDR)

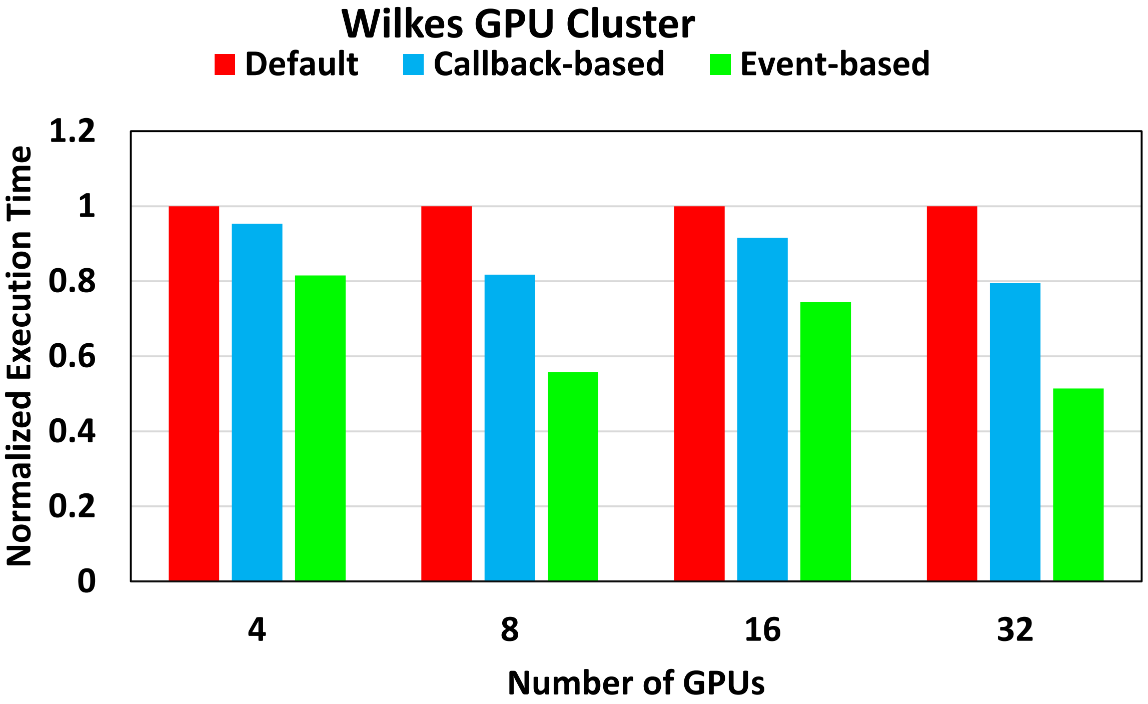


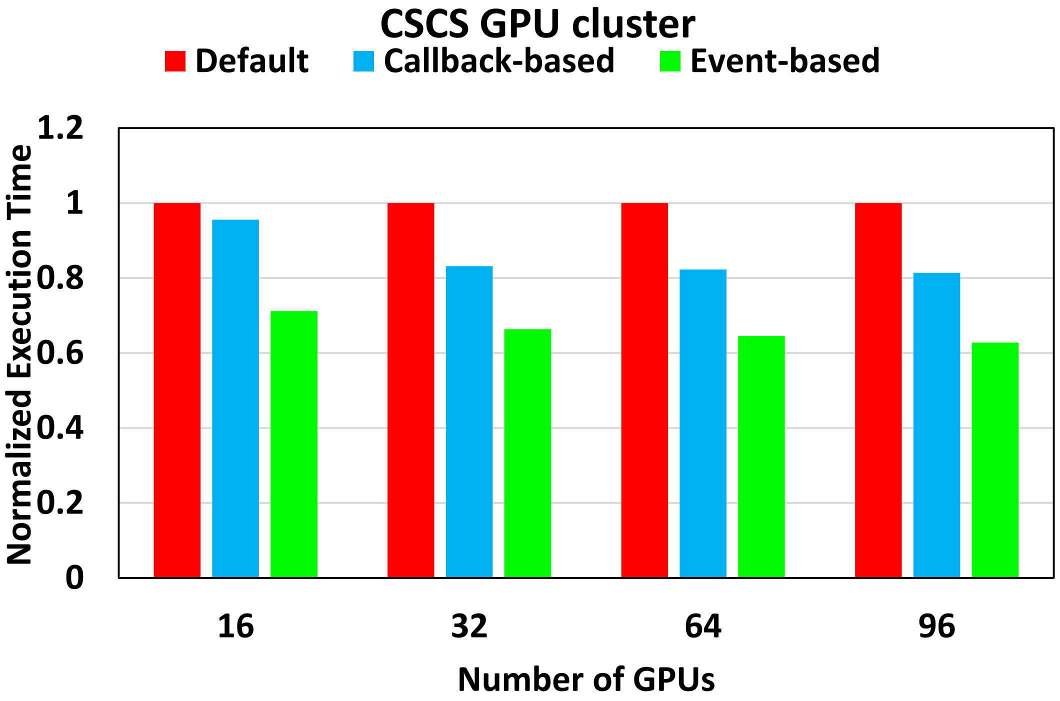




MVAPICH2-GDR-2.2rc1
Intel Ivy Bridge (E5-2680 v2) node - 20 cores
NVIDIA Tesla K40c GPU
Mellanox Connect-X4 EDR HCA
CUDA 7.5
Mellanox OFED 3.0 with GPU-Direct-RDMA

Application-Level Evaluation (HaloExchange - Cosmo)





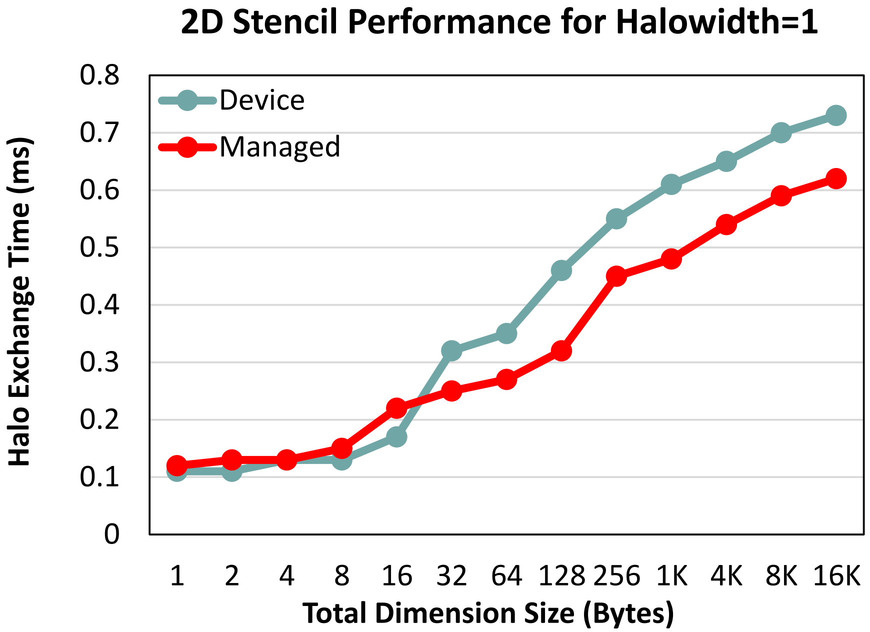
- 2X improvement on 32 GPUs nodes
- 30% improvement on 96 GPU nodes (8 GPUs/node)

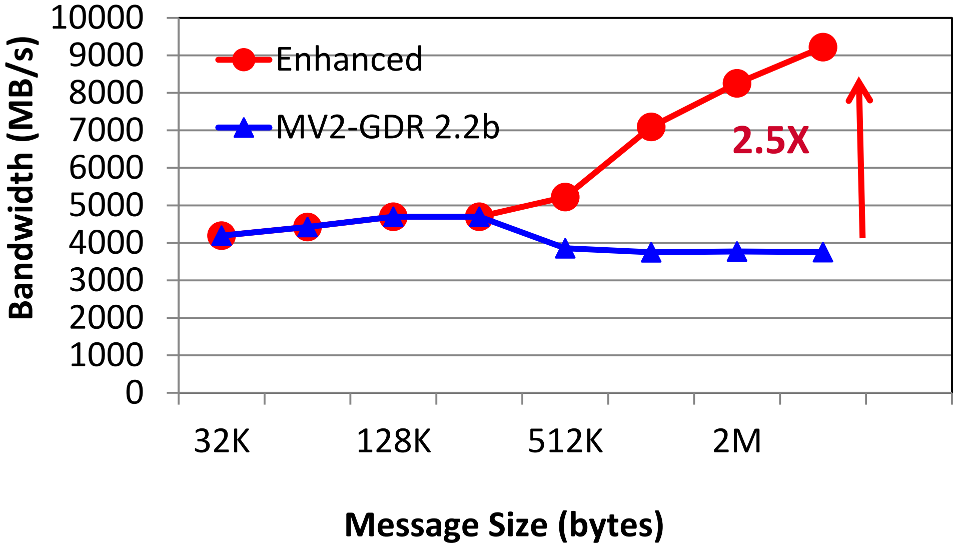
On-going collaboration with CSCS and MeteoSwiss in co-designing MV2-GDR and Cosmo Application

C. Chu, K. Hamidouche, A. Venkatesh, D. Banerjee , H. Subramoni, and D. K. Panda, Exploiting Maximal Overlap for Non-Contiguous Data Movement Processing on Modern GPU-enabled Systems, IPDPS'16

Enhanced Support for GPU Managed Memory

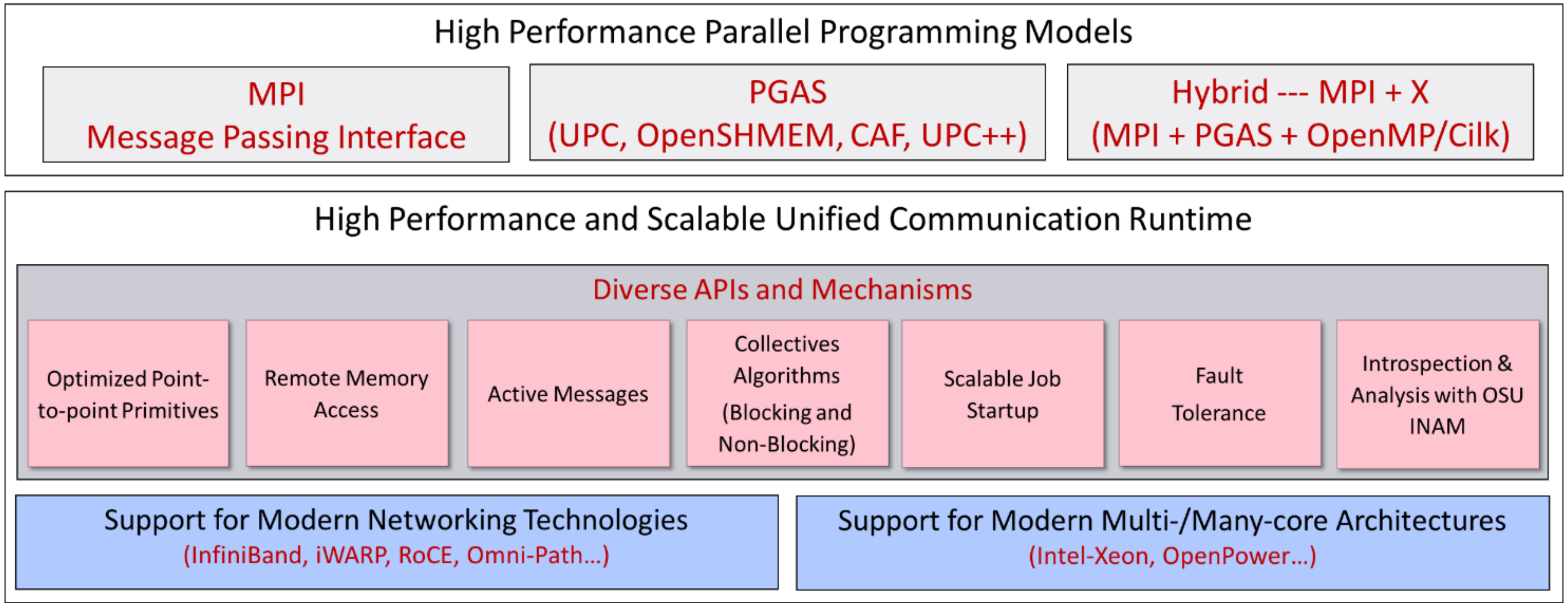
- CUDA Managed => no memory pin down
  - No IPC support for intranode communication
  - No GDR support for Internode communication
- Significant productivity benefits due to abstraction of explicit allocation and *cudaMemcpy()*
- Initial and basic support in MVAPICH2-GDR
  - For both intra- and inter-nodes use "pipeline through" host memory
- Enhance intranode managed memory to use IPC
  - Double buffering pair-wise IPC-based scheme
  - Brings IPC performance to Managed memory
  - High performance and high productivity
  - 2.5 X improvement in bandwidth
- OMB extended to evaluate the performance of point-to-point and collective communications using managed buffers





D. S. Banerjee, K Hamidouche, and D. K Panda, Designing High Performance Communication Runtime for GPUManaged Memory: Early Experiences, GPGPU-9 Workshop, to be held in conjunction with PPoPP '16

MVAPICH2-X for Hybrid MPI + PGAS Applications


High Performance Parallel Programming Models
MPI
Message Passing Interface
PGAS
(UPC, OpenSHMEM, CAF, UPC++)
Hybrid --- MPI + X
(MPI + PGAS + OpenMP/Cilk)
High Performance and Scalable Unified Communication Runtime
Diverse APIs and Mechanisms
Optimized Point-to-point Primitives
Remote Memory Access
Active Messages
Collectives Algorithms (Blocking and Non-Blocking)
Scalable Job Startup
Fault Tolerance
Introspection & Analysis with OSU INAM
Support for Modern Networking Technologies
(InfiniBand, iWARP, RoCE, Omni-Path…)
Support for Modern Multi-/Many-core Architectures
(Intel-Xeon, OpenPower…)

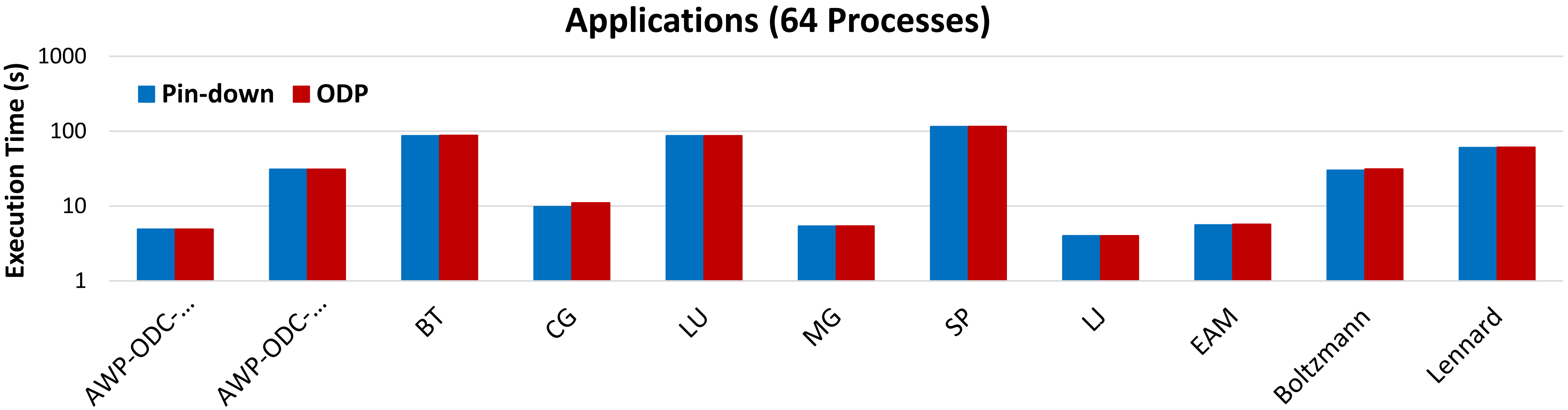
- Current Model – Separate Runtimes for OpenSHMEM/UPC/UPC++/CAF and MPI
  - Possible deadlock if both runtimes are not progressed
  - Consumes more network resource
- Unified communication runtime for MPI, UPC, UPC++, OpenSHMEM, CAF
  - Available with since 2012 (starting with MVAPICH2-X 1.9)
  - http://mvapich.cse.ohio-state.edu

MVAPICH2-X 2.2rc2

- Released on 08/08/2016
- Major Features and Enhancements
  - MPI Features
    - Based on MVAPICH2 2.2rc2 (OFA-IB-CH3 interface)
    - Efficient support for On Demand Paging (ODP) feature of Mellanox for point-to-point and RMA operations
    - Support for Intel Knights Landing architecture
  - UPC Features
    - Support for Intel Knights Landing architecture
  - UPC++ Features
    - Support for Intel Knights Landing architecture
  - OpenSHMEM Features
    - Support for Intel Knights Landing architecture
  - CAF Features
    - Support for Intel Knights Landing architecture
  - Hybrid Program Features
    - Support Intel Knights Landing architecture for hybrid MPI+PGAS applications
  - Unified Runtime Features
    - Based on MVAPICH2 2.2rc2 (OFA-IB-CH3 interface). All the runtime features enabled by default in OFA-IB-CH3 and OFA-IB-RoCE interface of MVAPICH2 2.2rc2 are available in MVAPICH2-X 2.2rc2

On-Demand Paging (ODP)

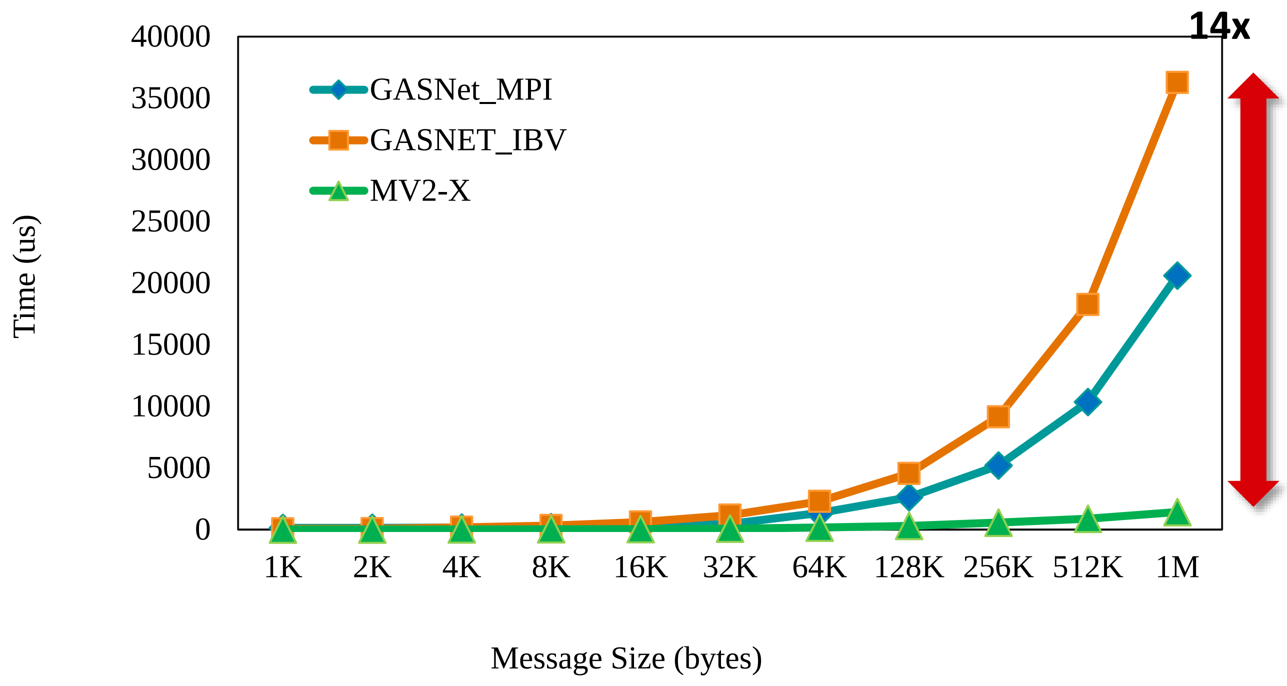
- Introduced by Mellanox to avoid pinning the pages of registered memory regions
- ODP-aware runtime could reduce the size of pin-down buffers while maintaining performance



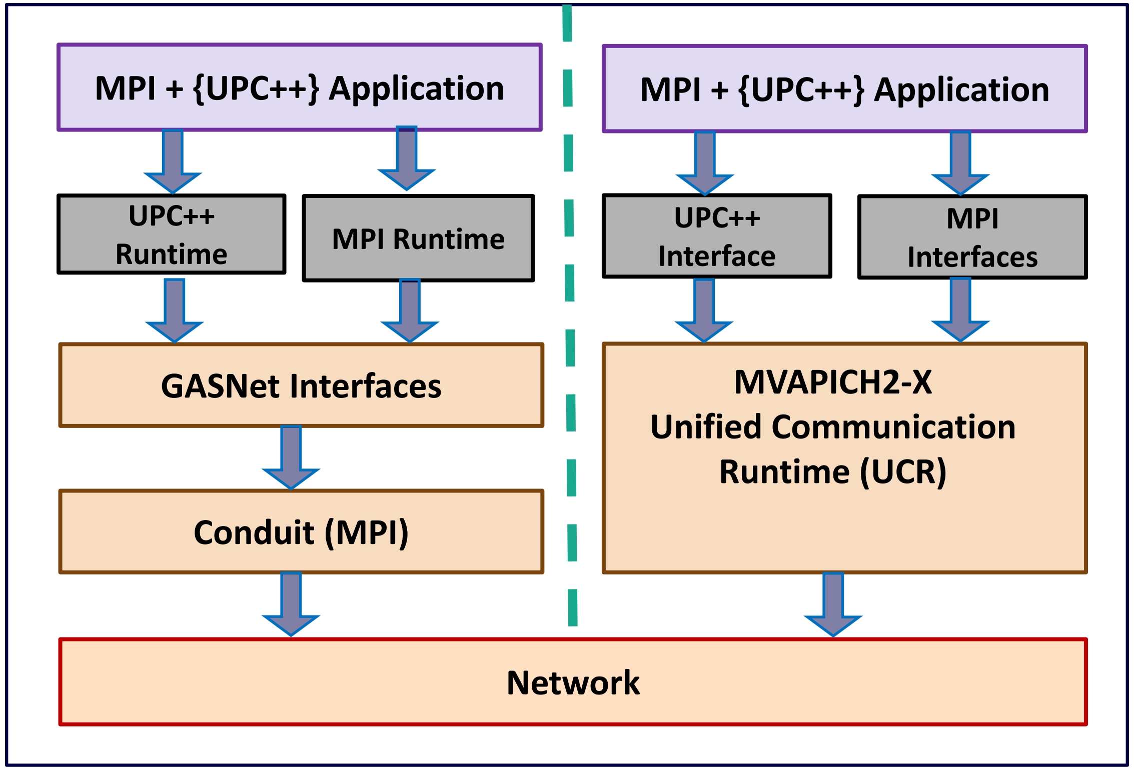
M. Li, K. Hamidouche, X. Lu, H. Subramoni, J. Zhang, and D. K. Panda, “Designing MPI Library with On-Demand Paging (ODP) of InfiniBand: Challenges and Benefits”, SC, 2016

More Details in Student Poster Presentation

UPC++ Support in MVAPICH2-X



Inter-node Broadcast (64 nodes 1:ppn)


MPI + {UPC++} Application
UPC++ Runtime
MPI Runtime
GASNet Interfaces
Conduit (MPI)
MPI + {UPC++} Application
UPC++ Interface
MPI Interfaces
MVAPICH2-X Unified Communication Runtime (UCR)
Network

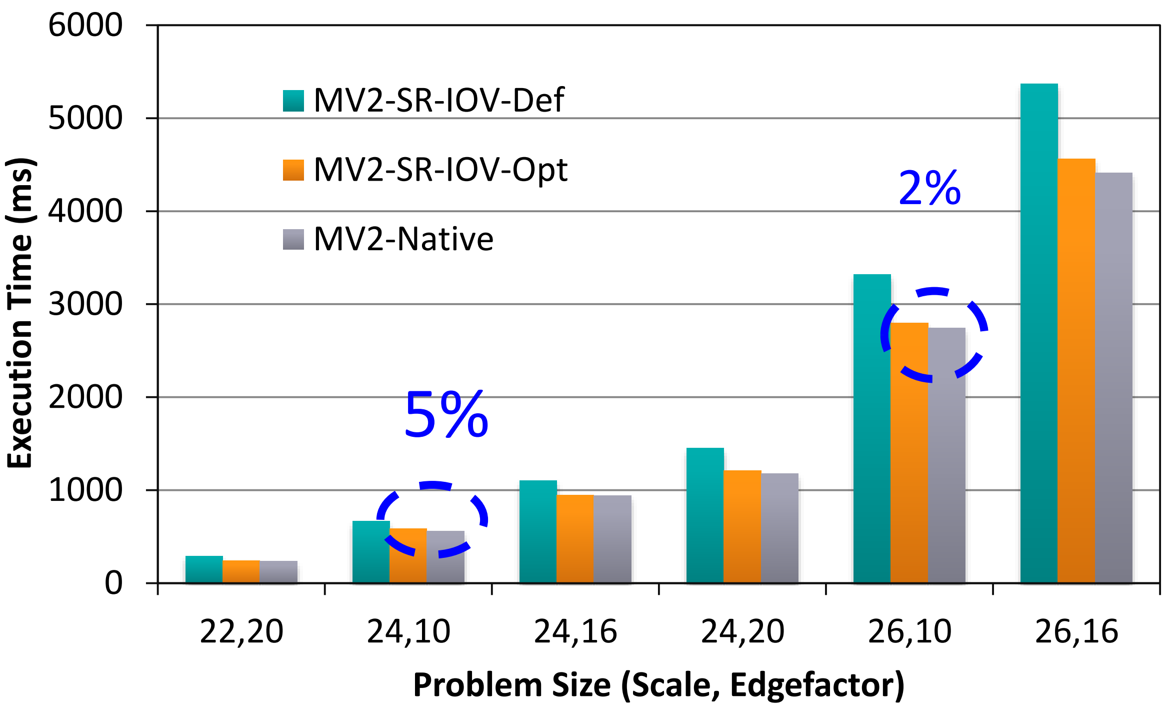
- Full and native support for hybrid MPI + UPC++ applications
- Better performance compared to IBV and MPI conduits
- OSU Micro-benchmarks (OMB) support for UPC++
- Available since MVAPICH2-X (2.2rc1)

More Details in Student Poster Presentation

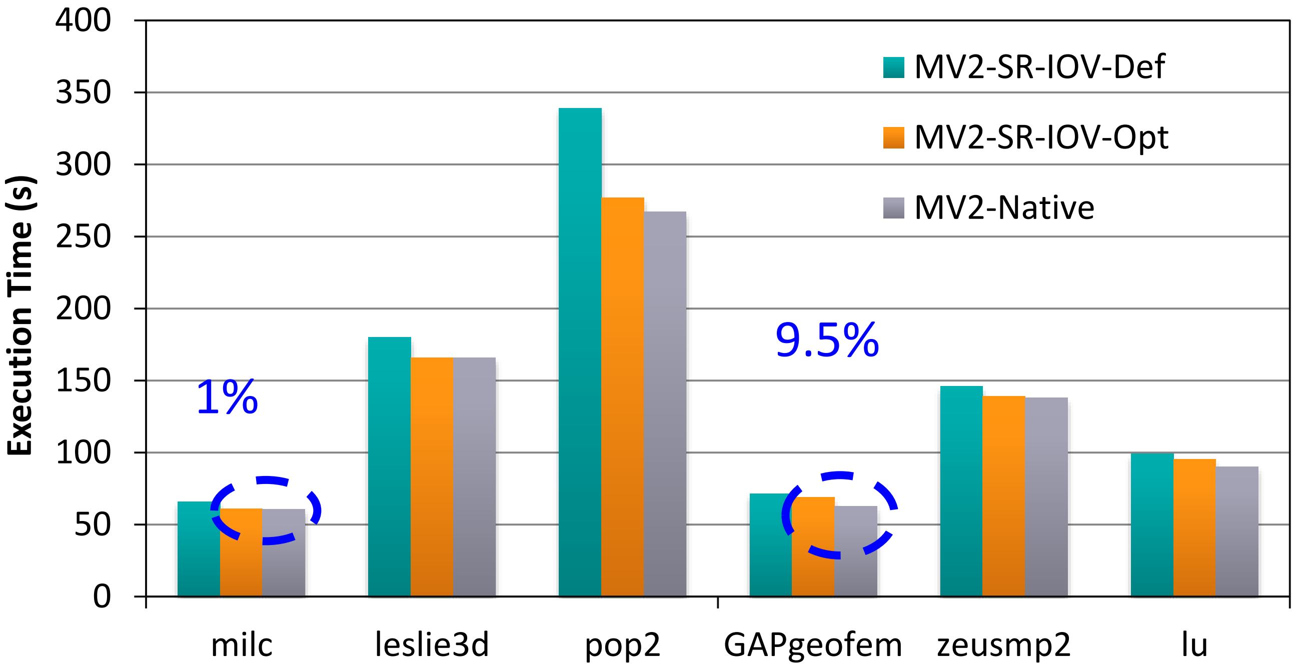
MVAPICH2-Virt 2.2rc1

- Released on 07/12/2016
- Major Features and Enhancements
  - Based on MVAPICH2 2.2rc1
  - High-performance and locality-aware MPI communication with IPC-SHM and CMA for containers
  - Support for locality auto-detection in containers
  - Automatic communication channel selection among IPC-SHM, CMA, and HCA
  - Support for easy configuration through runtime parameters
  - Tested with
    - Docker 1.9.1 and 1.10.3
    - Mellanox InfiniBand adapters (ConnectX-3 (56Gbps))

Application-Level Performance on Chameleon (SR-IOV Support)



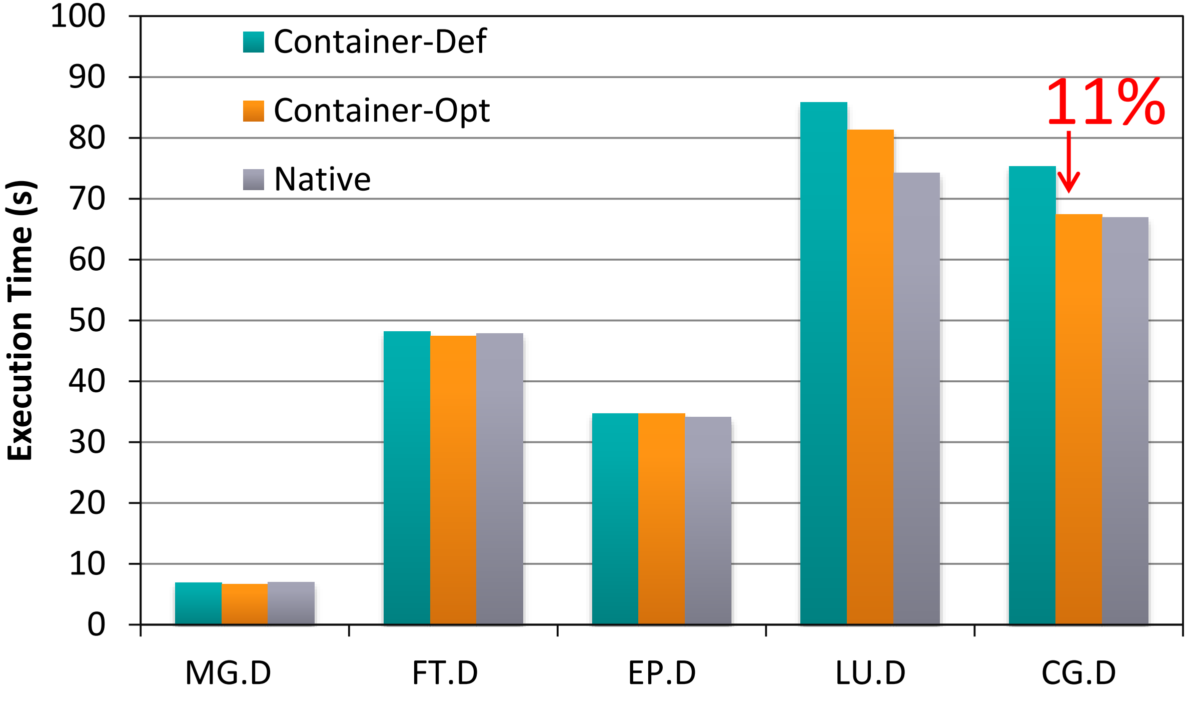
Graph500



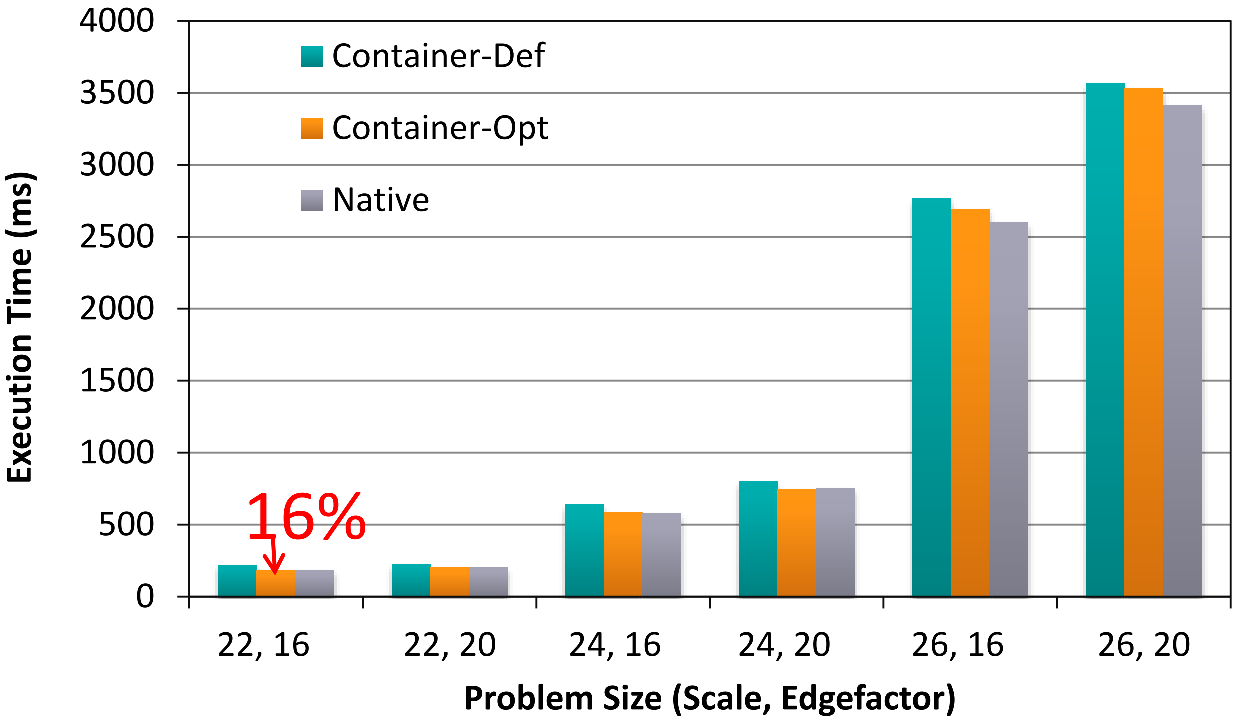
SPEC MPI2007

- 32 VMs, 6 Core/VM
- Compared to Native, 2-5% overhead for Graph500 with 128 Procs
- Compared to Native, 1-9.5% overhead for SPEC MPI2007 with 128 Procs

Application-Level Performance on Chameleon (Containers Support)



NAS



Graph 500

- 64 Containers across 16 nodes, pining 4 Cores per Container
- Compared to Container-Def, up to 11% and 16% of execution time reduction for NAS and Graph 500
- Compared to Native, less than 9 % and 4% overhead for NAS and Graph 500

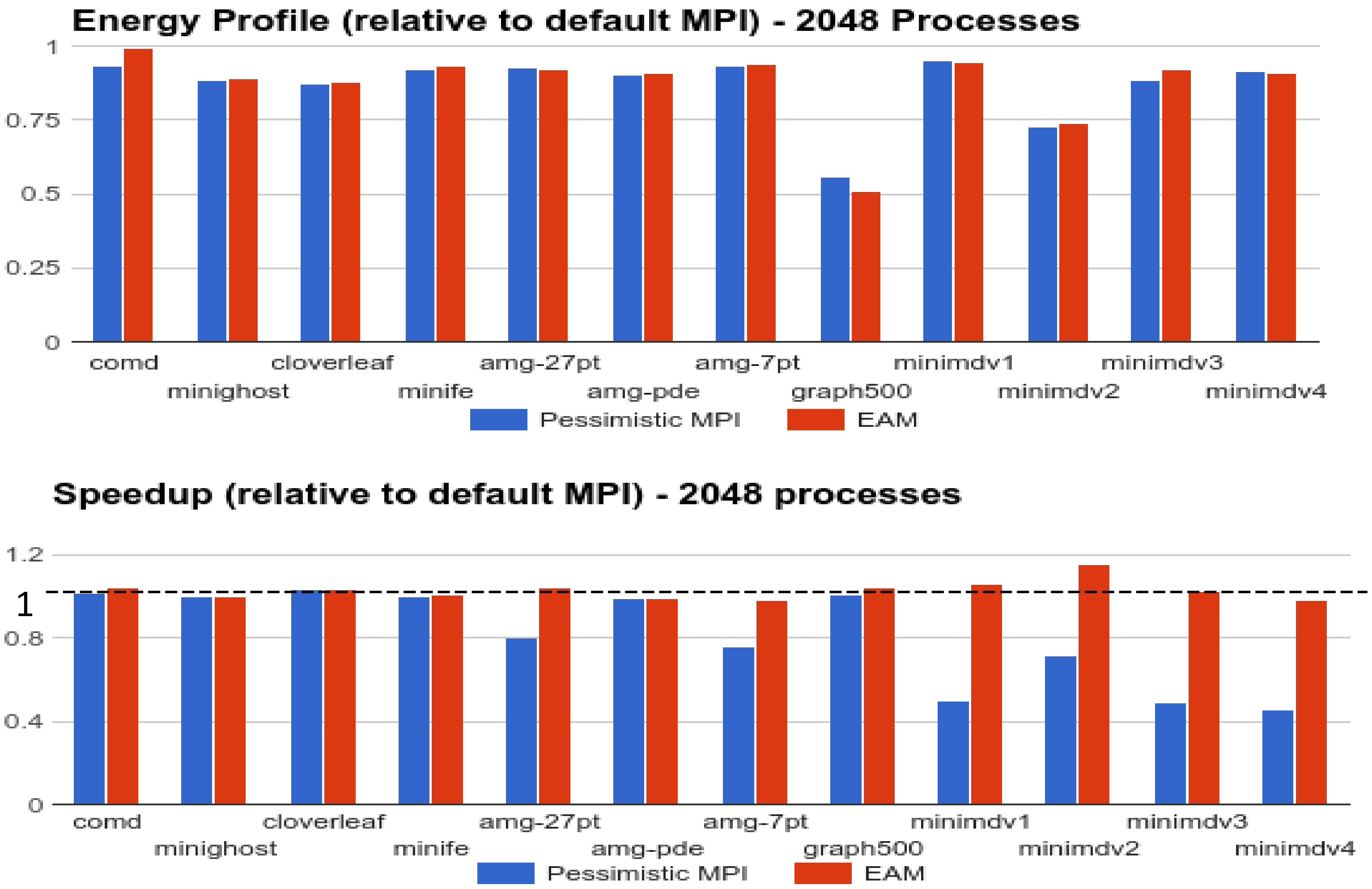
Energy-Aware MVAPICH2 & OSU Energy Management Tool (OEMT)

- MVAPICH2-EA 2.1 (Energy-Aware)
  - A white-box approach
  - New Energy-Efficient communication protocols for pt-pt and collective operations
  - Intelligently apply the appropriate Energy saving techniques
  - Application oblivious energy saving

- OEMT
  - A library utility to measure energy consumption for MPI applications
  - Works with all MPI runtimes
  - PRELOAD option for precompiled applications
  - Does not require ROOT permission:
    - A safe kernel module to read only a subset of MSRs

MVAPICH2-EA: Application Oblivious Energy-Aware-MPI (EAM)

- An energy efficient runtime that provides energy savings without application knowledge
- Uses automatically and transparently the best energy lever
- Provides guarantees on maximum degradation with 5-41% savings at <= 5% degradation
- Pessimistic MPI applies energy reduction lever to each MPI call

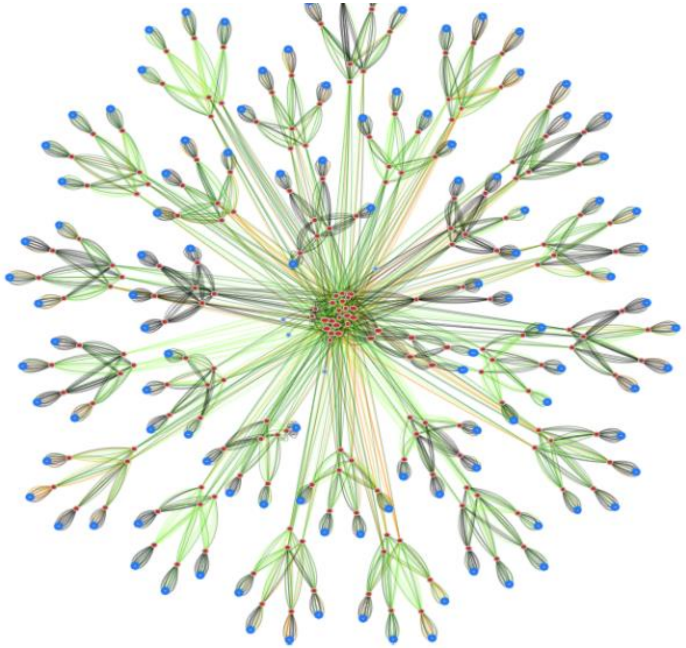


A Case for Application-Oblivious Energy-Efficient MPI Runtime A. Venkatesh, A. Vishnu, K. Hamidouche, N. Tallent, D. K. Panda, D. Kerbyson, and A. Hoise, Supercomputing '15, Nov 2015 *[Best Student Paper Finalist]*

Overview of OSU INAM

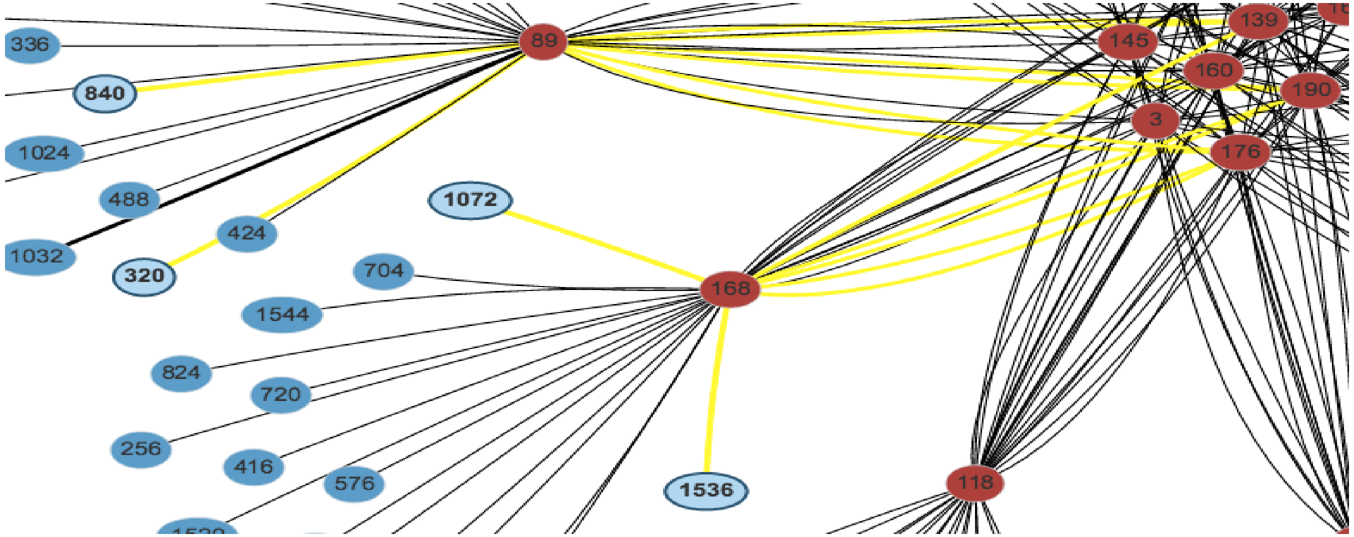
- A network monitoring and analysis tool that is capable of analyzing traffic on the InfiniBand network with inputs from the MPI runtime
  - http://mvapich.cse.ohio-state.edu/tools/osu-inam/
- Monitors IB clusters in real time by querying various subnet management entities and gathering input from the MPI runtimes
- OSU INAM v0.9.1 released on 05/13/16
- Significant enhancements to user interface to enable scaling to clusters with thousands of nodes
- Improve database insert times by using 'bulk inserts‘
- Capability to look up list of nodes communicating through a network link
- Capability to classify data flowing over a network link at job level and process level granularity in conjunction with MVAPICH2-X 2.2rc1
- “Best practices “ guidelines for deploying OSU INAM on different clusters
- Capability to analyze and profile node-level, job-level and process-level activities for MPI communication
  - Point-to-Point, Collectives and RMA
- Ability to filter data based on type of counters using “drop down” list
- Remotely monitor various metrics of MPI processes at user specified granularity
- "Job Page" to display jobs in ascending/descending order of various performance metrics in conjunction with MVAPICH2-X
- Visualize the data transfer happening in a “live” or “historical” fashion for entire network, job or set of nodes

OSU INAM Features



Comet@SDSC --- Clustered View

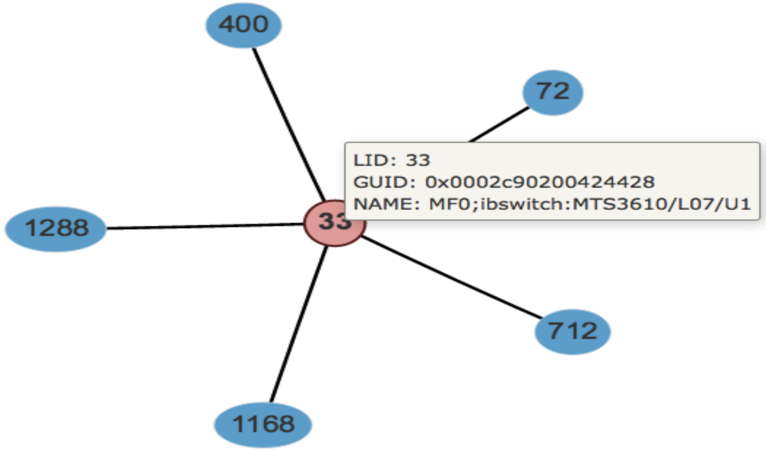
(1,879 nodes, 212 switches, 4,377 network links)



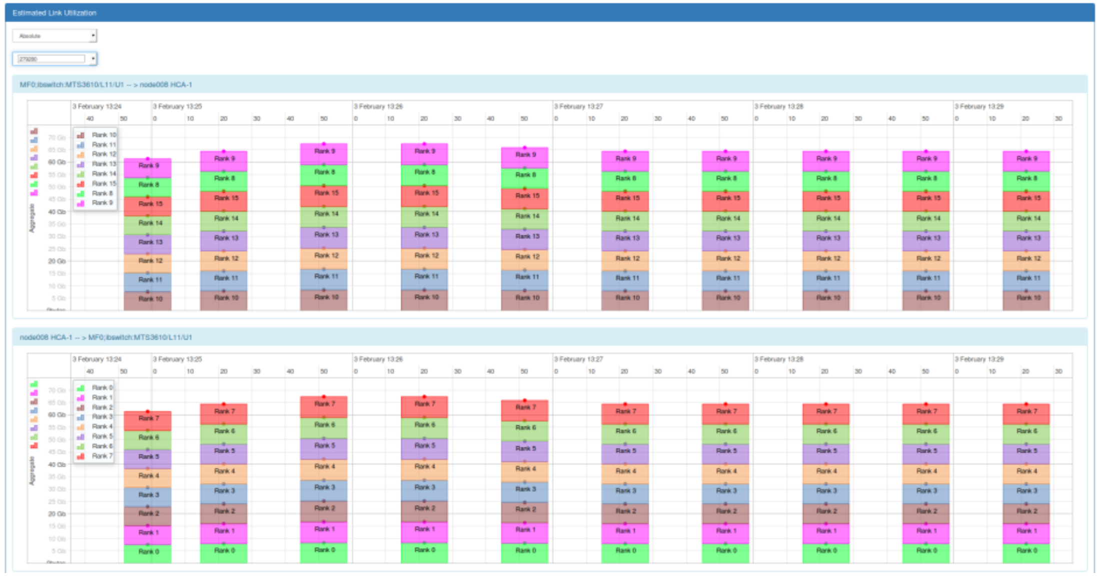
Finding Routes Between Nodes

- Show network topology of large clusters
- Visualize traffic pattern on different links
- Quickly identify congested links/links in error state
- See the history unfold – play back historical state of the network

OSU INAM Features (Cont.)



Visualizing a Job (5 Nodes)



Estimated Process Level Link Utilization

- Job level view
  - Show different network metrics (load, error, etc.) for any live job
  - Play back historical data for completed jobs to identify bottlenecks
- Node level view - details per process or per node
  - CPU utilization for each rank/node
  - Bytes sent/received for MPI operations (pt-to-pt, collective, RMA)
  - Network metrics (e.g. XmitDiscard, RcvError) per rank/node

- Estimated Link Utilization view
  - Classify data flowing over a network link at different granularity in conjunction with MVAPICH2-X 2.2rc1
    - Job level and
    - Process level

Demo Presentation Session Tomorrow

OSU Microbenchmarks

- Available since 2004
- Suite of microbenchmarks to study communication performance of various programming models
- Benchmarks available for the following programming models
  - Message Passing Interface (MPI)
  - Partitioned Global Address Space (PGAS)
    - Unified Parallel C (UPC)
    - Unified Parallel C++ (UPC++)
    - OpenSHMEM
- Benchmarks available for multiple accelerator based architectures
  - Compute Unified Device Architecture (CUDA)
  - OpenACC Application Program Interface
- Part of various national resource procurement suites like NERSC-8 / Trinity Benchmarks
- Please visit the following link for more information
  - http://mvapich.cse.ohio-state.edu/benchmarks/

Looking into the Future ….

- Application Best Practices
- One-build To Rule Them All ☺
- Dynamic and Adaptive Tag Matching
- High-performance fault recovery using Reinit
- Performance Engineering Applications using MVAPICH2 and TAU
- High-Performance Support for GPU-based Streaming Applications in MVAPICH2-GDR
- Control Flow Decoupling through GPUDirect Async
- MVAPICH2-GDR Powered High-performance Deep Learning
- Energy-awareness for MPI3-RMA model
- Extending OpenSHMEM for Accelerators
- Challenges and Opportunities in Enabling SR-IOV and IVSHMEM with SLURM

Applications-Level Tuning: Compilation of Best Practices

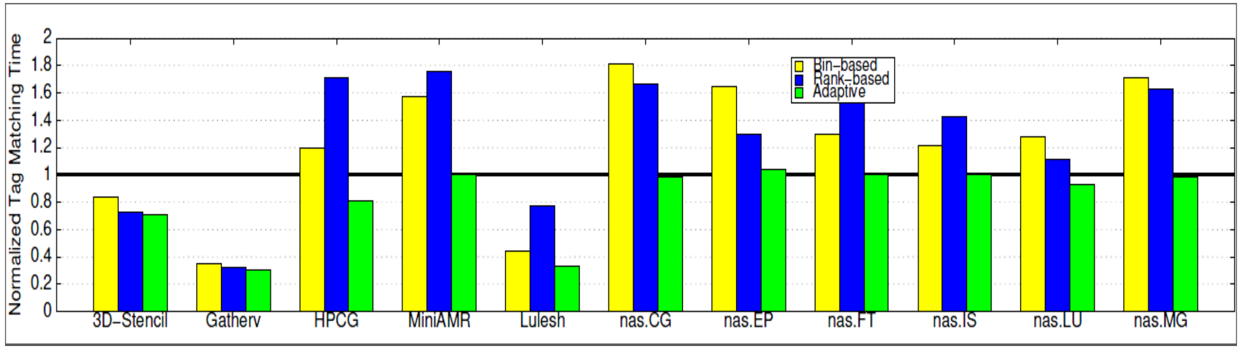
- MPI runtime has many parameters
- Tuning a set of parameters can help you to extract higher performance
- Compiled a list of such contributions through the MVAPICH Website
  - http://mvapich.cse.ohio-state.edu/best_practices/
- Initial list of applications
  - Amber
  - HoomDBlue
  - HPCG
  - Lulesh
  - MILC
  - Neuron
  - SMG2000
- Soliciting additional contributions, send your results to mvapich-help at cse.ohio-state.edu.
- We will link these results with credits to you.

One Build to Rule Them All ☺

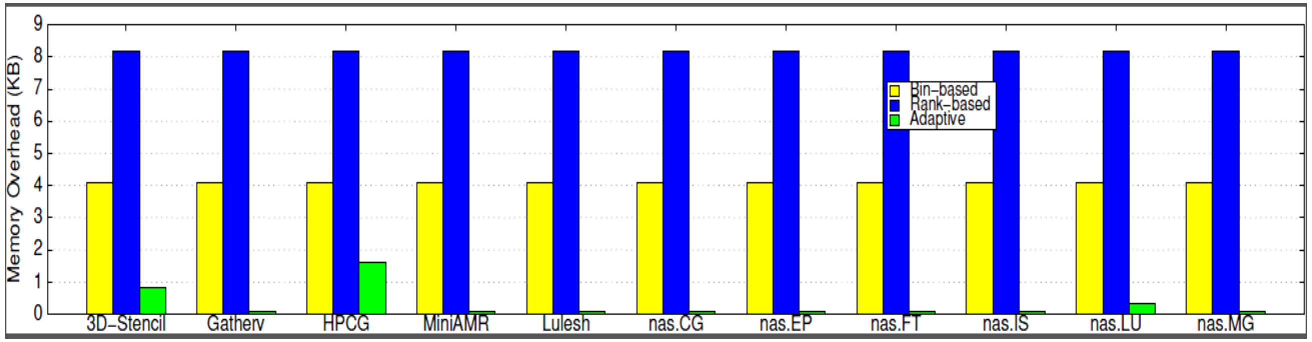
- Goal
  - To avoid the hassle of having to build MVAPICH2 for each supported network interface
    - InfiniBand, PSM and PSM2
- Challenges
  - Unifying structure of data types controlled by build time flags suitable for both interfaces
    - Restructuring code paths effected by build time flags to respond correctly to runtime switches
    - Separating interface business logic from logic common to both interfaces
- What MVAPICH2 users see with one-build
  - Build for both gen2 and psm interfaces in one installation
  - Dynamically select between interfaces at runtime
    - Environment variable set for job
  - Will be available in future MVAPICH2 releases

Dynamic and Adaptive Tag Matching

- Challenge:
  - Tag matching presents a significant overhead for receiver
  - Existing solutions
    - Are static and do not adapt dynamically to the communication load
    - Do not consider memory overhead of advanced solutions
- Solution:
  - A tag matching design which dynamically adapts to communication load at each individual process at runtime
- Results
  - Delivers the best performance when compared with multiple state-of-the-art tag matching schemes
  - Limits the memory consumed to the absolute minimum necessary
  - Will be available in future MVAPICH2 releases



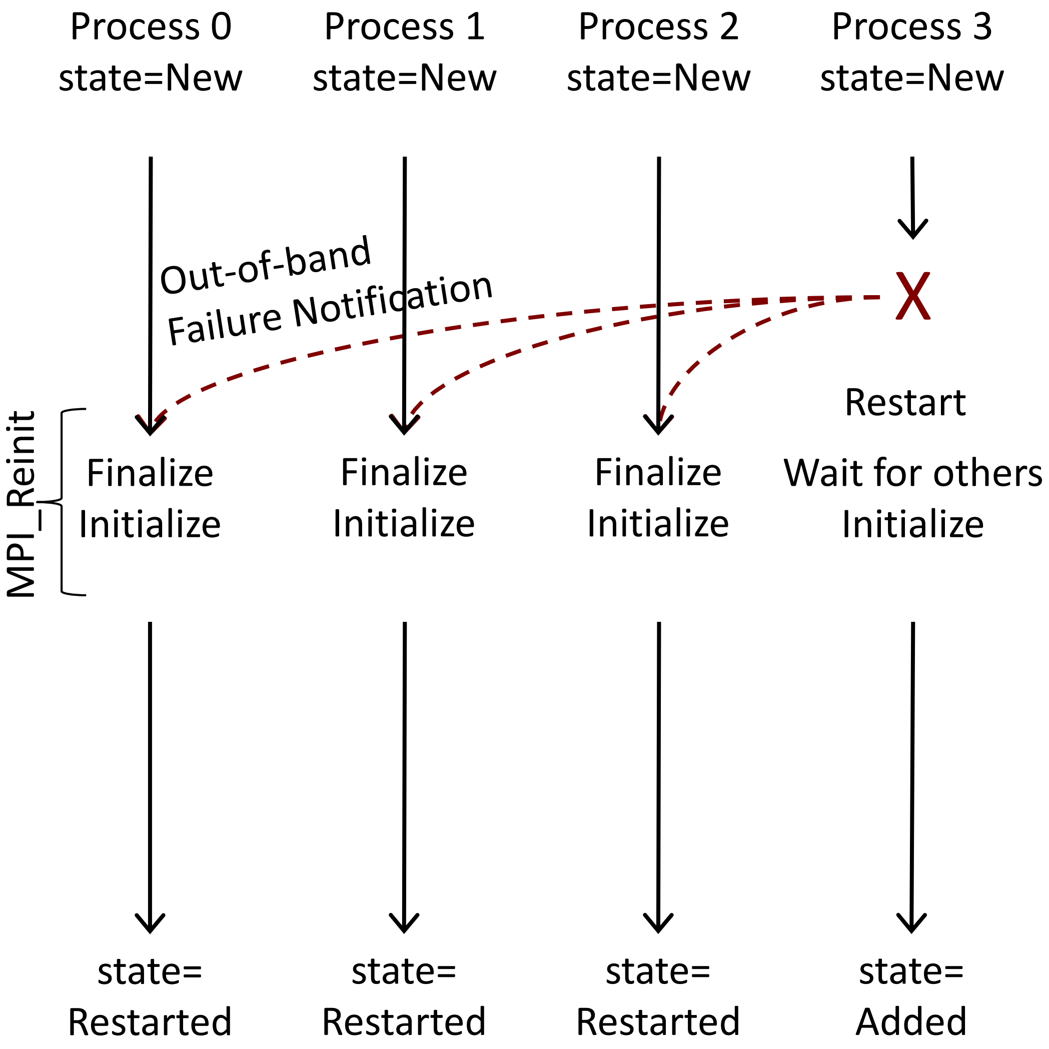
Normalized Total Tag Matching Time at 512 Processes
Normalized to Default (Lower is Better)

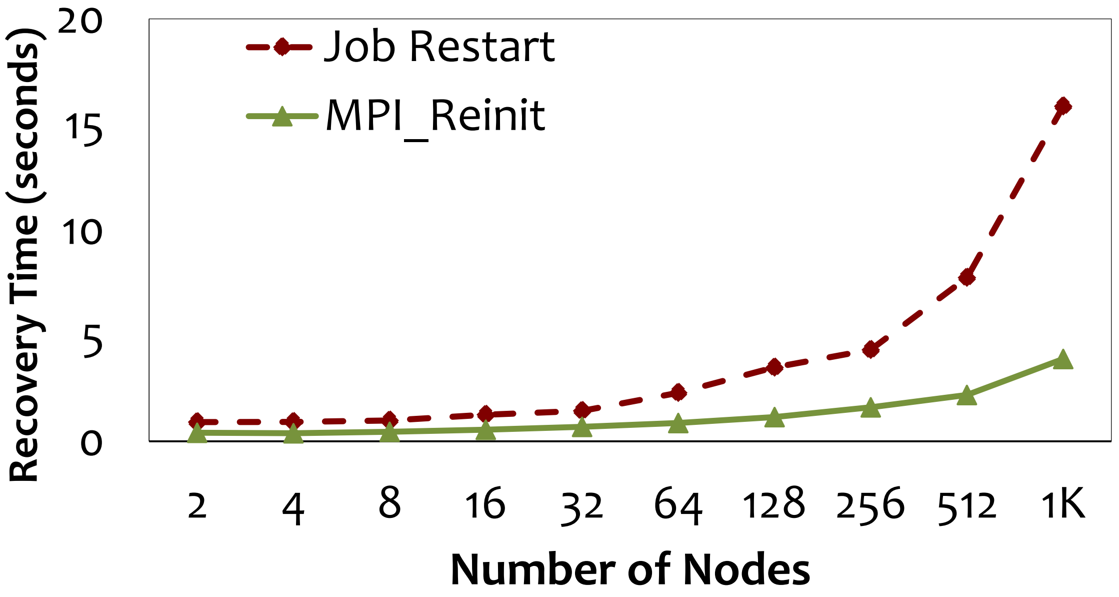


Normalized Memory Overhead per Process at 512 Processes
Compared to Default (Lower is Better)

Adaptive and Dynamic Design for MPI Tag Matching; M. Bayatpour, H. Subramoni, S. Chakraborty, and D. K. Panda; IEEE Cluster 2016. [Best Paper Nominee]

MPI_Reinit – A New Approach to Fault Tolerance





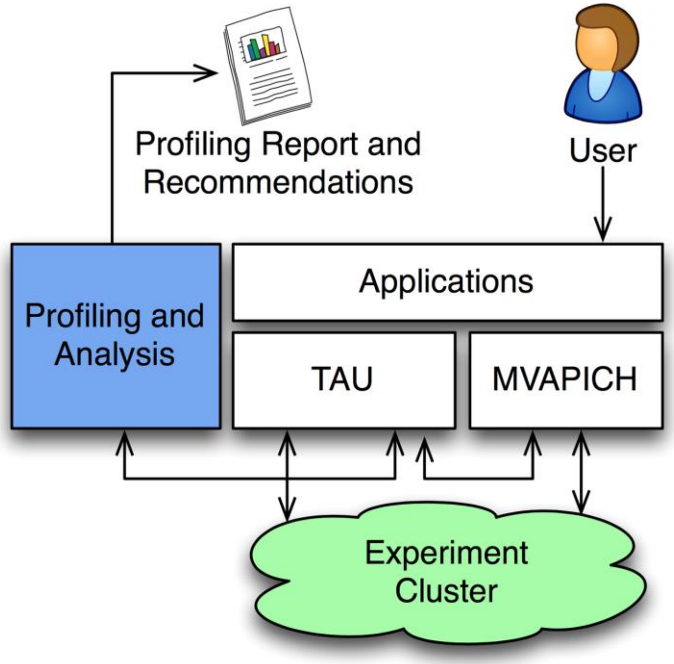
- Restart only the failed process
- Less than 4 seconds to recover from process failure with 1K nodes, 12K processes
- Recovery with Reinit is up to 4 times faster than Job restart

More details in Ignacio Laguna’s talk

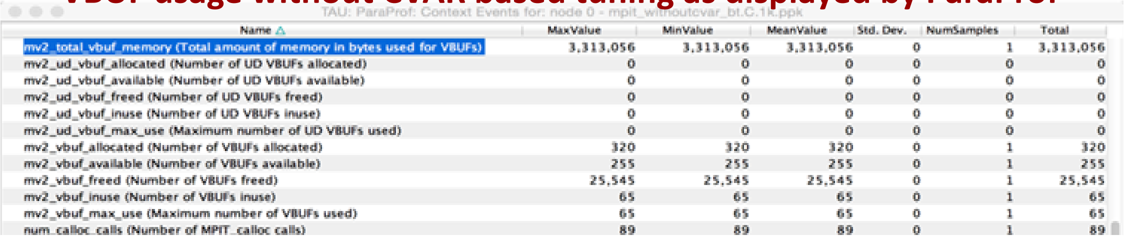
Performance Engineering Applications using MVAPICH2 and TAU

- Enhance existing support for MPI_T in MVAPICH2 to expose a richer set of performance and control variables
- Get and display MPI Performance Variables (PVARs) made available by the runtime in TAU
- Control the runtime's behavior via MPI Control Variables (CVARs)

- Introduced support for new MPI_T based CVARs to MVAPICH2
  - MPIR_CVAR_MAX_INLINE_MSG_SZ, MPIR_CVAR_VBUF_POOL_SIZE, MPIR_CVAR_VBUF_SECONDARY_POOL_SIZE
- TAU enhanced with support for setting MPI_T CVARs in a non-interactive mode for uninstrumented applications
- More details in Sameer Shende's talk tomorrow and student poster presentation

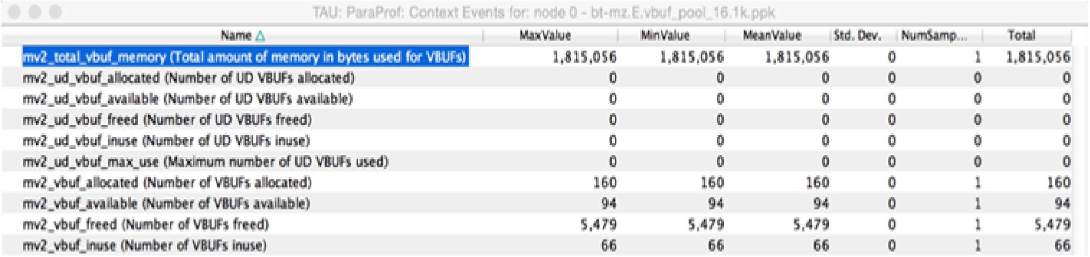


VBUF usage without CVAR based tuning as displayed by ParaProf


TAU: ParaProf: Context Events for: node 0 - mpit_withoutcvar_bt.C.1k.ppk

Name △	MaxValue	MinValue	MeanValue	Std. Dev.	NumSamples	Total
mv2_total_vbuf_memory (Total amount of memory in bytes used for VBUFs)	3,313,056	3,313,056	3,313,056	0	1	3,313,056
mv2_ud_vbuf_allocated (Number of UD VBUFs allocated)	0	0	0	0	0	0
mv2_ud_vbuf_available (Number of UD VBUFs available)	0	0	0	0	0	0
mv2_ud_vbuf_freed (Number of UD VBUFs freed)	0	0	0	0	0	0
mv2_ud_vbuf_inuse (Number of UD VBUFs inuse)	0	0	0	0	0	0
mv2_ud_vbuf_max_use (Maximum number of UD VBUFs used)	0	0	0	0	0	0
mv2_vbuf_allocated (Number of VBUFs allocated)	320	320	320	0	1	320
mv2_vbuf_available (Number of VBUFs available)	255	255	255	0	1	255
mv2_vbuf_freed (Number of VBUFs freed)	25,545	25,545	25,545	0	1	25,545
mv2_vbuf_inuse (Number of VBUFs inuse)	65	65	65	0	1	65
mv2_vbuf_max_use (Maximum number of VBUFs used)	65	65	65	0	1	65
num_calloc_calls (Number of MPIT_calloc calls)	89	89	89	0	1	89

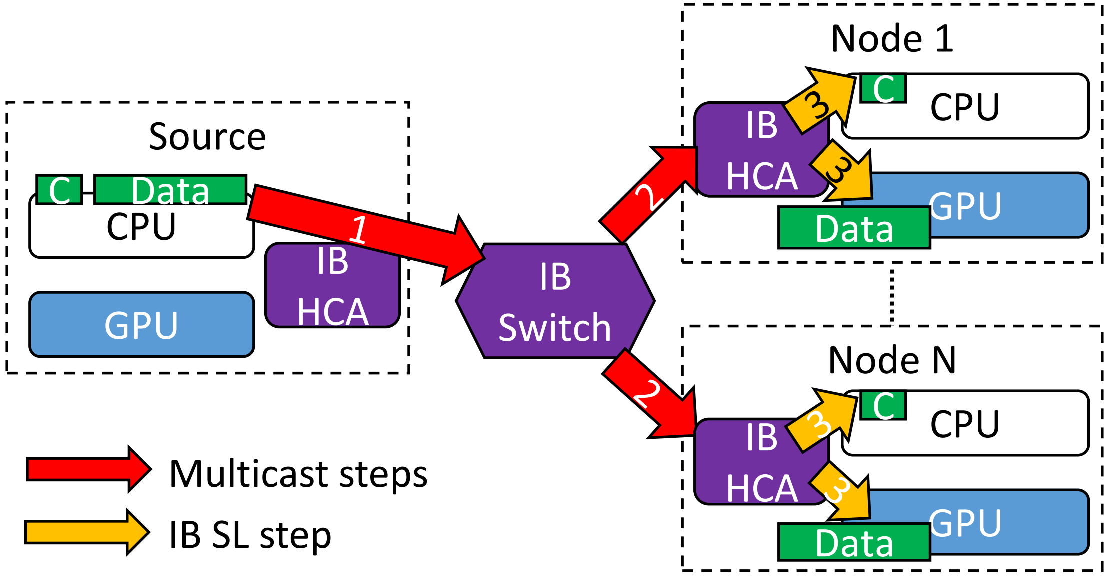
VBUF usage with CVAR based tuning as displayed by ParaProf

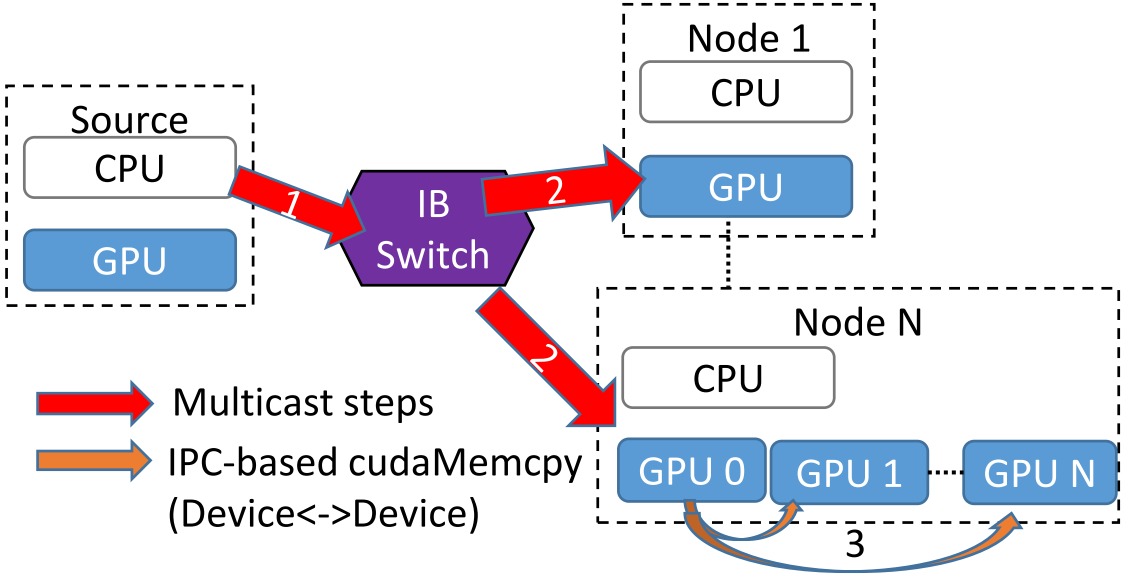

TAU: ParaProf: Context Events for: node 0 - bt-mz.E.vbuf_pool_16.1k.ppk

Name △	MaxValue	MinValue	MeanValue	Std. Dev.	NumSamp...	Total
mv2_total_vbuf_memory (Total amount of memory in bytes used for VBUFs)	1,815,056	1,815,056	1,815,056	0	1	1,815,056
mv2_ud_vbuf_allocated (Number of UD VBUFs allocated)	0	0	0	0	0	0
mv2_ud_vbuf_available (Number of UD VBUFs available)	0	0	0	0	0	0
mv2_ud_vbuf_freed (Number of UD VBUFs freed)	0	0	0	0	0	0
mv2_ud_vbuf_inuse (Number of UD VBUFs inuse)	0	0	0	0	0	0
mv2_ud_vbuf_max_use (Maximum number of UD VBUFs used)	0	0	0	0	0	0
mv2_vbuf_allocated (Number of VBUFs allocated)	160	160	160	0	1	160
mv2_vbuf_available (Number of VBUFs available)	94	94	94	0	1	94
mv2_vbuf_freed (Number of VBUFs freed)	5,479	5,479	5,479	0	1	5,479
mv2_vbuf_inuse (Number of VBUFs inuse)	66	66	66	0	1	66

High-Performance Heterogeneous Broadcast for Streaming Applications

- Streaming applications on GPU clusters
  - Using a pipeline of **broadcast** operations to move host-resident data from a single source—typically live— to multiple GPU-based computing sites
  - Existing schemes require explicitly data movements between Host and GPU memories
    - ➔Poor performance and breaking the pipeline
- IB hardware multicast + Scatter-List
  - Efficient heterogeneous-buffer broadcast operation
- CUDA Inter-Process Communication (IPC)
  - Efficient intra-node topology-aware broadcast operations for multi-GPU systems
- Will be available in future MVAPICH2-GDR release

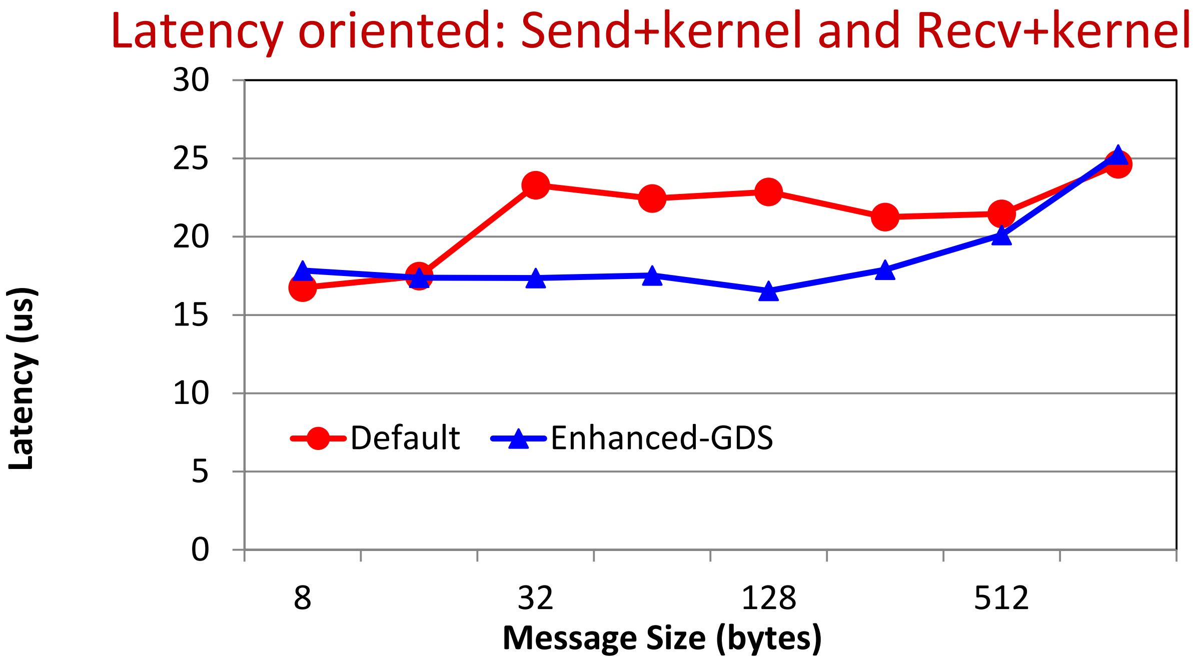


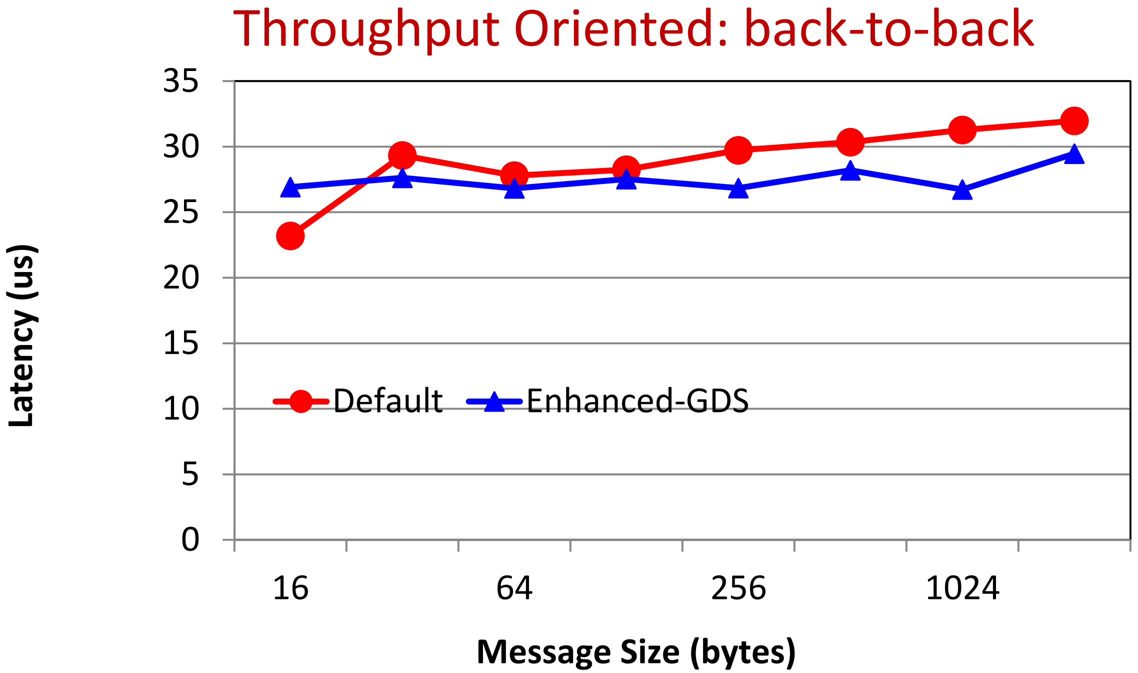


Designing High Performance Heterogeneous Broadcast for Streaming Applications on GPU Clusters. C.-H. Chu, K. Hamidouche, H. Subramoni, A. Venkatesh , B. Elton, and D. K. Panda, SBAC-PAD'16, Oct 2016. *(Accepted to be presented)*

More Details in Student Poster Presentation

MVAPICH2-GDS: Preliminary Results





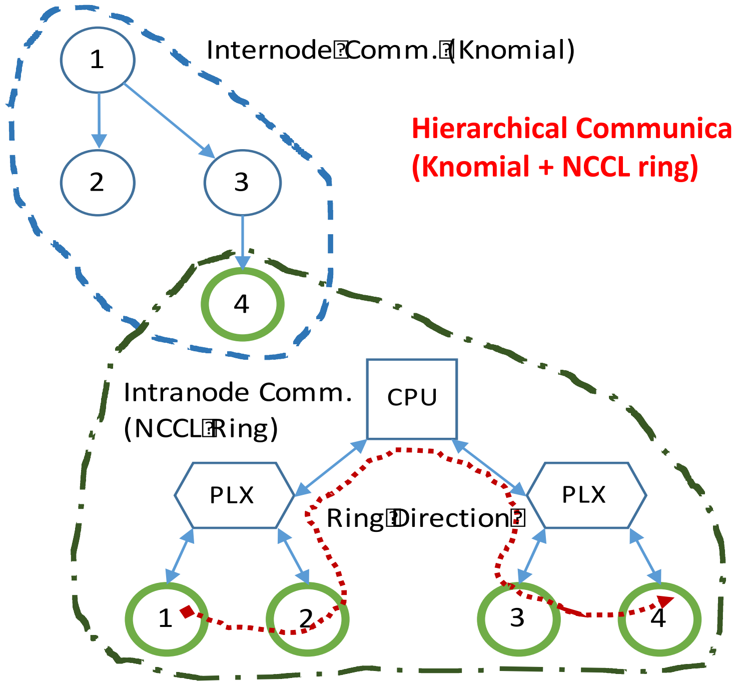
- Latency Oriented: Able to hide the kernel launch overhead
  - 25% improvement at 256 Bytes compared to default behavior
- Throughput Oriented: Asynchronously to offload queue the Communication and computation tasks
  - 14% improvement at 1KB message size

Intel Sandy Bridge, NVIDIA K20 and Mellanox FDR HCA

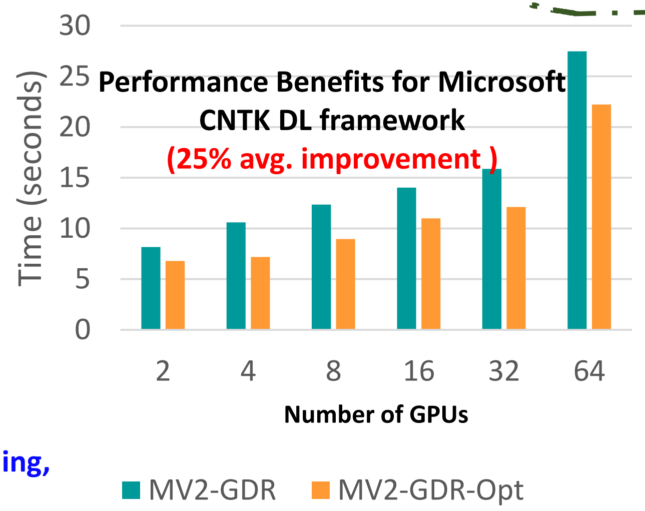
Will be available in a public release soon

Efficient Large Message Broadcast using MVAPICH2-GDR and NCCL for Deep Learning

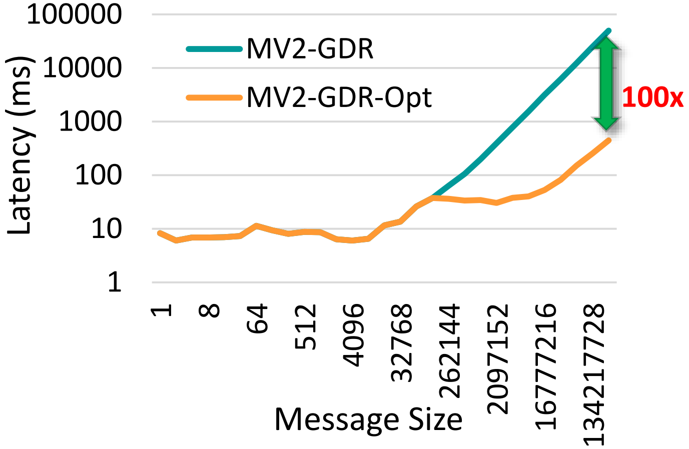
- NCCL has some limitations
  - Only works for a single node, thus, no **scale-out** on multiple nodes
  - Degradation across IOH (socket) for **scale-up** (within a node)
- Propose optimized MPI_Bcast for MVAPICH2-GDR
  - Communication of very large GPU buffers
    - Order of megabytes
  - Scalability on dense multi-GPU nodes
- Efficiently exploits:
  - CUDA-Aware **MPI_Bcast** in MV2-GDR
  - **NCCL Broadcast** primitive
    - Ring exchange
  - Hierarchical Communication (Knomial + NCCL ring)
- Will be available in future MV2-GDR releases



Hierarchical Communication (Knomial + NCCL ring)


Performance Benefits for Microsoft CNTK DL framework

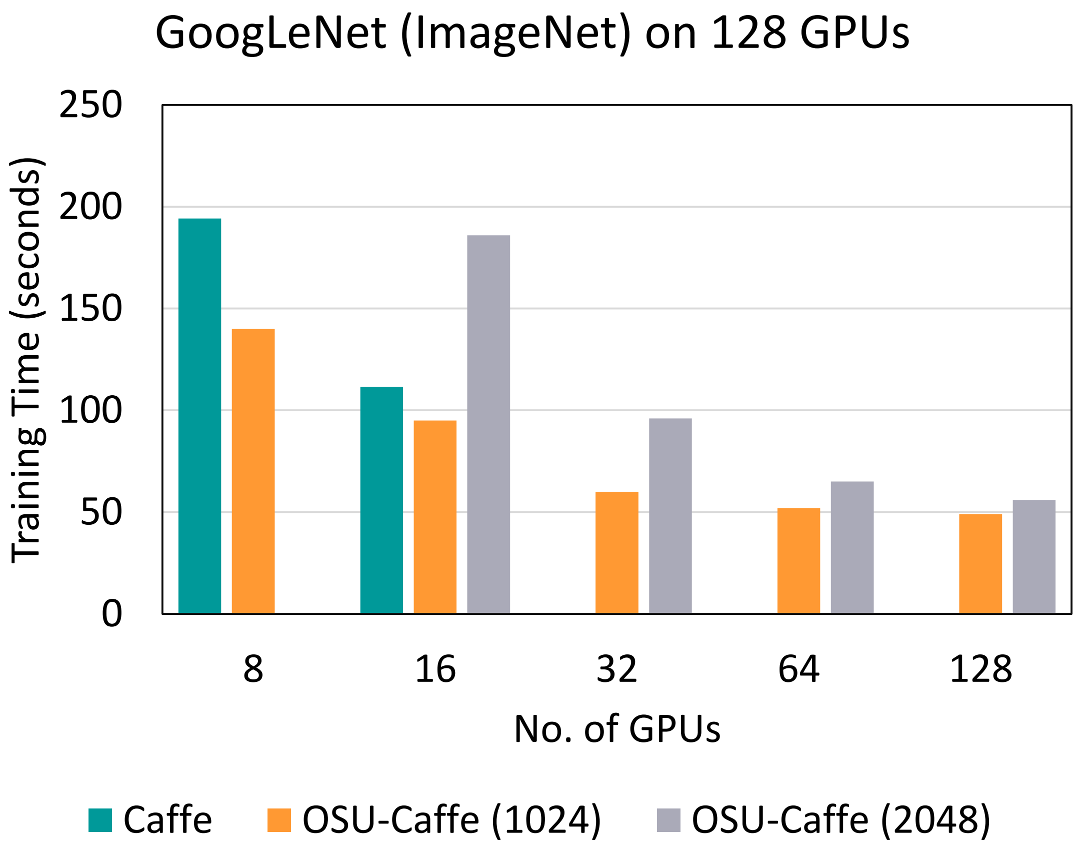
(25% avg. improvement)



Efficient Large Message Broadcast using NCCL and CUDA-Aware MPI for Deep Learning, A. Awan , K. Hamidouche , A. Venkatesh , and D. K. Panda, The 23rd European MPI Users' Group Meeting (EuroMPI 16), Sep 2016 [Best Paper Runner-Up]

Efficient Deep Learning with MVAPICH2-GDR

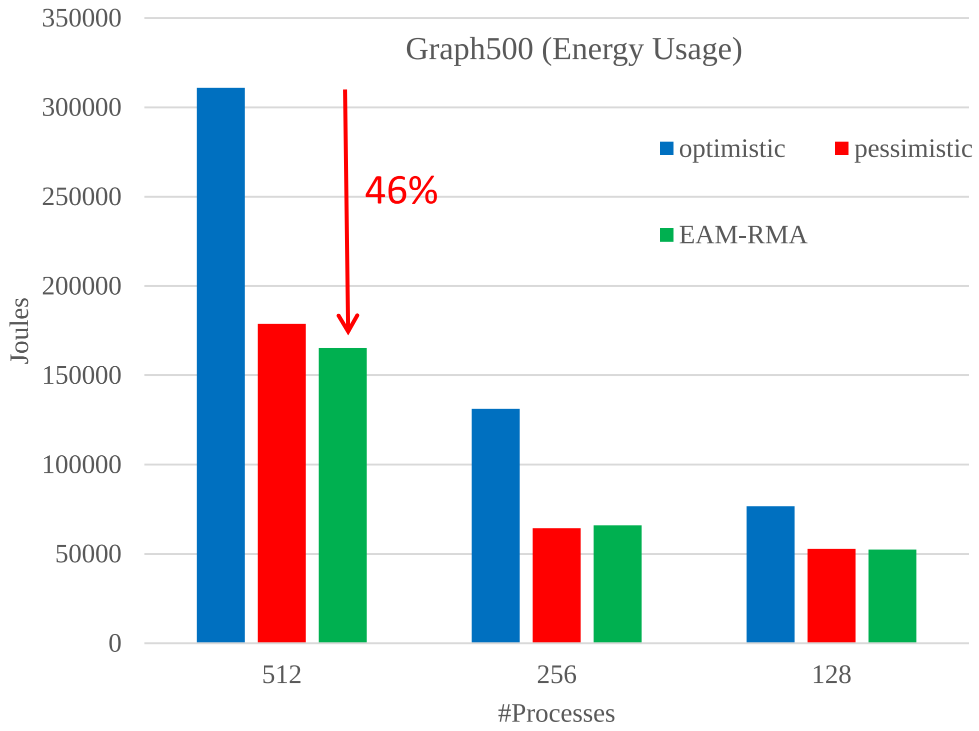
- Caffe : A flexible and layered Deep Learning framework.
- Benefits and Weaknesses
  – Multi-GPU Training within a single node
  – Performance degradation for GPUs across different sockets
  – No Scale-out available
- OSU-Caffe: MPI-based Parallel Training
  – Enable Scale-up (within a node) and Scale-out (across multi-GPU nodes)
  – Scale-out on 64 GPUs for training CIFAR-10 network on CIFAR-10 dataset
  – Scale-out on 128 GPUs for training GoogLeNet network on ImageNet dataset

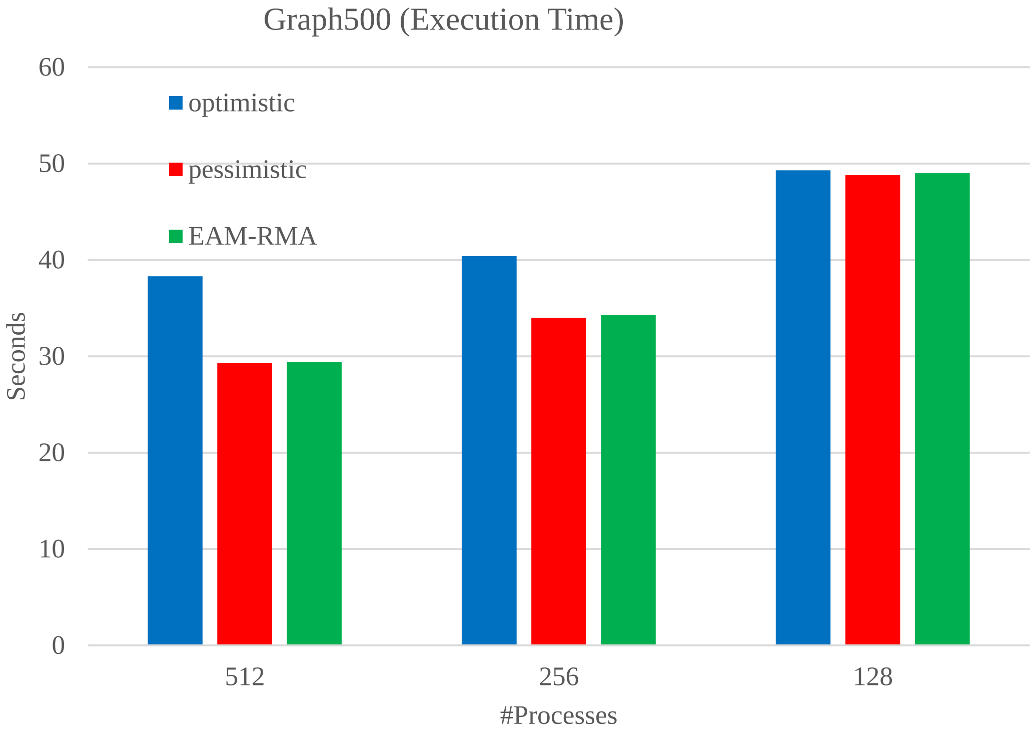


More in a Student Poster Presentation

OSU-Caffe will be publicly available soon

MPI-3 RMA Energy Savings with Proxy-Applications





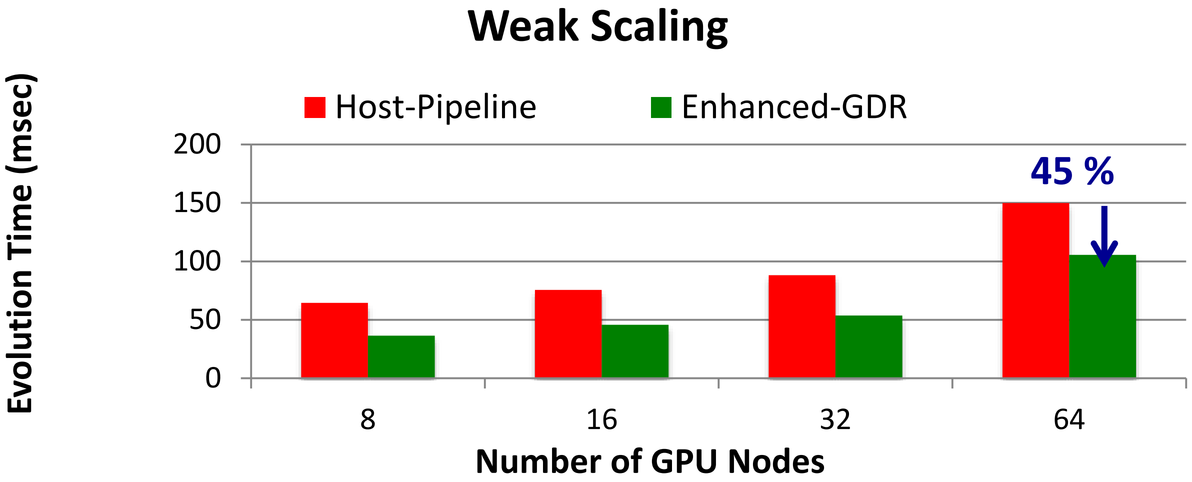
- MPI_Win_fence dominates application execution time in graph500
- Between 128 and 512 processes, EAM-RMA yields between 31% and 46% savings with no degradation in execution time in comparison with the default optimistic MPI runtime
- MPI-3 RMA Energy-efficient support will be available in upcoming MVAPICH2-EA release

CUDA-aware OpenSHMEM Runtime

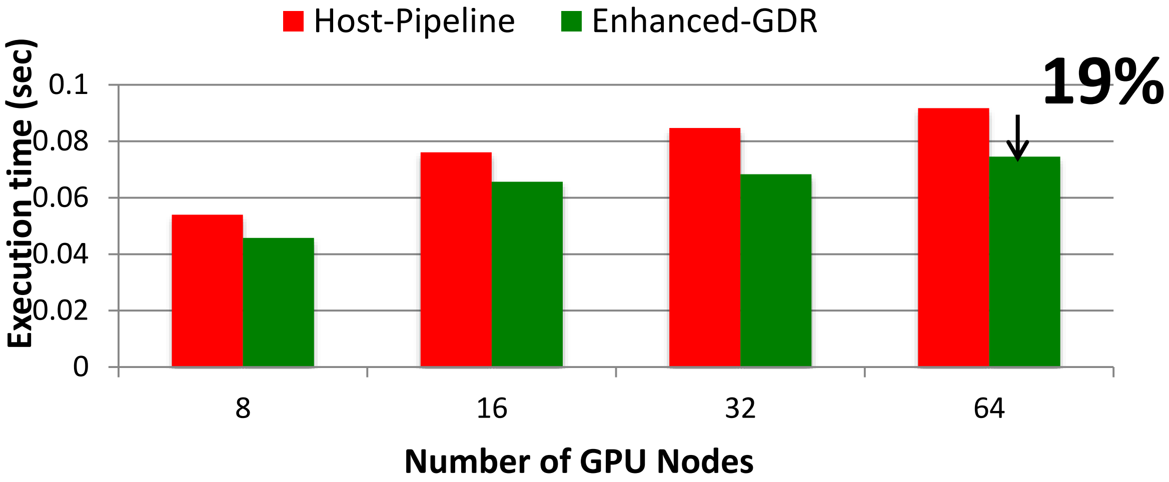
- After device memory becomes part of the global shared space:
  - Accessible through standard OpenSHMEM communication APIs
  - Data movement transparently handled by the runtime
  - Preserves one-sided semantics at the application level
- Efficient designs to handle communication
  - Inter-node transfers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

Application Evaluation: GPULBM and 2DStencil



GPULBM: 64x64x64



2DStencil 2Kx2K

- Redesign the application
  - CUDA-Aware MPI : Send/Recv => hybrid CUDA-Aware MPI+OpenSHMEM
  - cudaMalloc =>shmalloc(size,1);
  - MPI_Send/recv => shmem_put + fence
  - 53% and 45%
  - Degradation is due to small input size
  - Will be available in future MVAPICH2-GDR releases

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

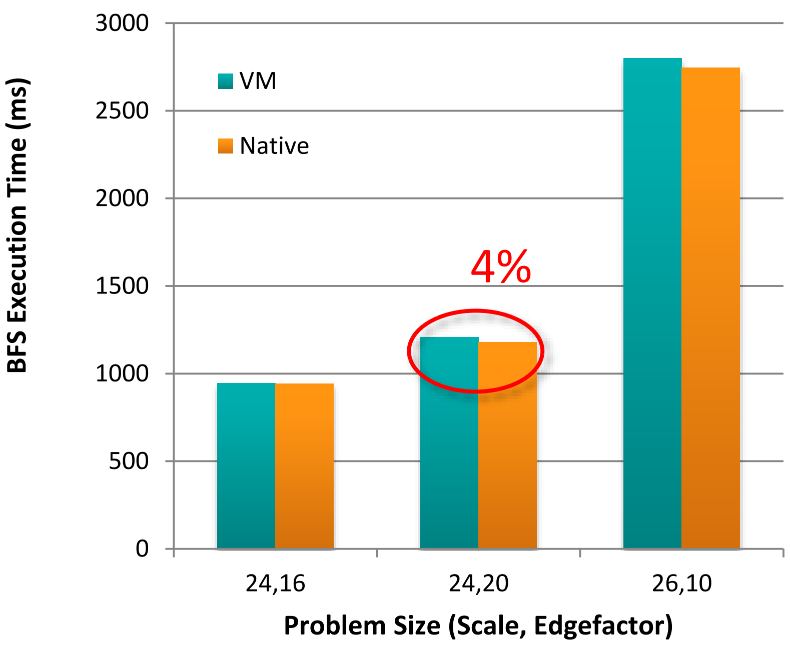
1. K. Hamidouche, A. Venkatesh, A. Awan, H. Subramoni, C. Ching and D. K. Panda, Exploiting GPUDirect RDMA in Designing High Performance OpenSHMEM for GPU Clusters. IEEE Cluster 2015.
2. K. Hamidouche, A. Venkatesh, A. Awan, H. Subramoni, C. Ching and D. K. Panda, CUDA-Aware OpenSHMEM: Extensions and Designs for High Performance OpenSHMEM on GPU Clusters. To appear in PARCO.

SR-IOV and IVSHMEM in SLURM

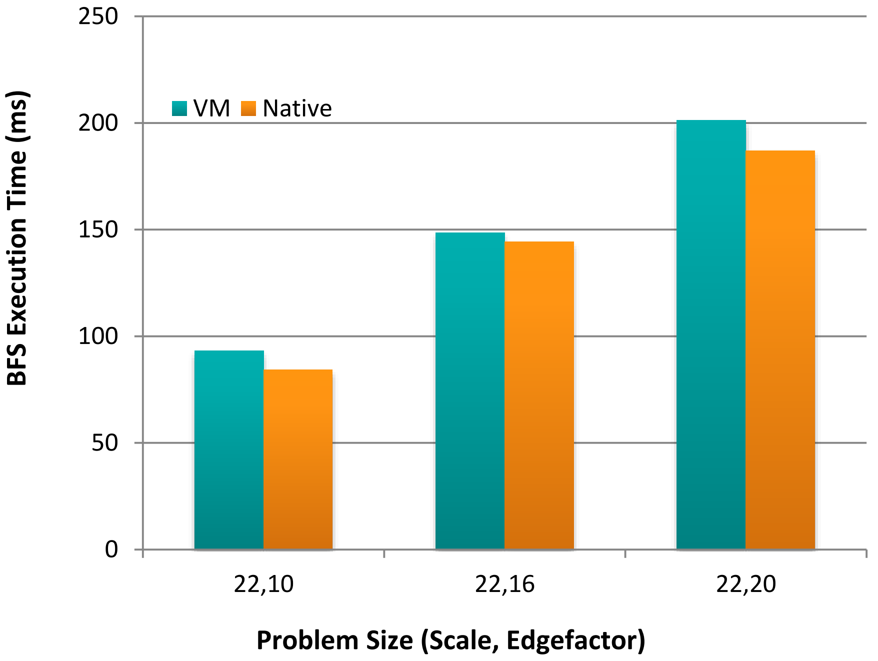
- Requirement of managing and isolating virtualized resources of SR-IOV and IVSHMEM
- Such kind of management and isolation is hard to be achieved by MPI library alone, but much easier with SLURM
- Efficient running MPI applications on HPC Clouds needs SLURM to support managing SR-IOV and IVSHMEM
  - Can critical HPC resources be efficiently shared among users by extending SLURM with support for SR-IOV and IVSHMEM based virtualization?
  - Can SR-IOV and IVSHMEM enabled SLURM and MPI library provide bare-metal performance for end applications on HPC Clouds?

J. Zhang, X. Lu, S. Chakraborty, and D. K. Panda. SLURM-V: Extending SLURM for Building Efficient HPC Cloud with SR-IOV and IVShmem. The 22nd International European Conference on Parallel Processing (Euro-Par), 2016.

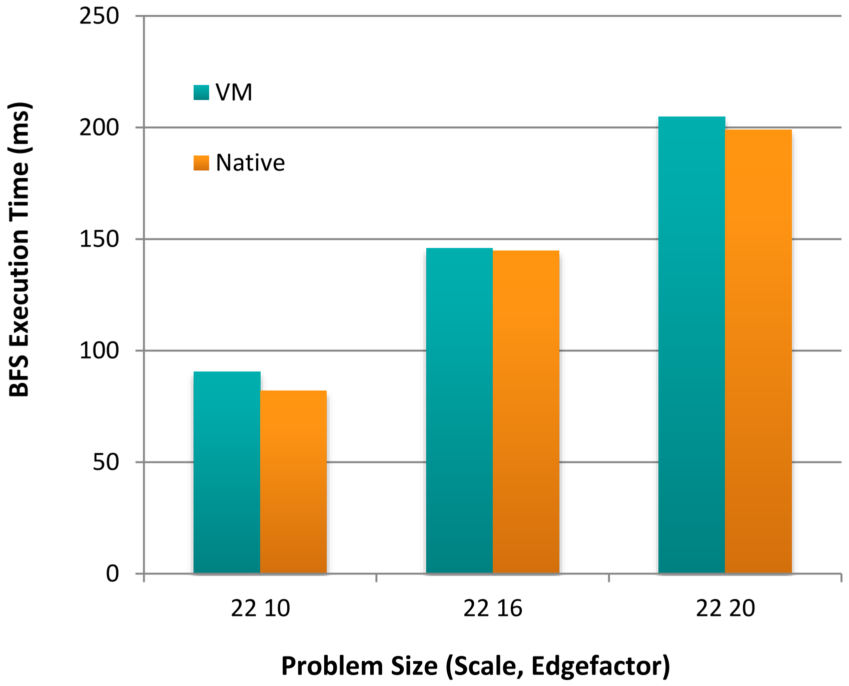
Application-Level Performance on Chameleon (Graph500)



Exclusive Allocations
Sequential Jobs



Shared-host Allocations
Concurrent Jobs



Exclusive Allocations
Concurrent Jobs

- 32 VMs across 8 nodes, 6 Core/VM
- EASJ - Compared to Native, less than 4% overhead with 128 Procs
- SACJ, EACJ – Also minor overhead, when running NAS as concurrent job with 64 Procs

Will be available in future MVAPICH2-Virt release together with support for Shifter and Singularity

MVAPICH2 – Plans for Exascale

- Performance and Memory scalability toward 1M cores
- Hybrid programming (MPI + OpenSHMEM, MPI + UPC, MPI + CAF …)
  - MPI + Task*
- Enhanced Optimization for GPU Support and Accelerators
- Taking advantage of advanced features of Mellanox InfiniBand
  - Switch-IB2 SHArP*
  - GID-based support*
- Enhanced communication schemes for upcoming architectures
  - Knights Landing with MCDRAM*
  - NVLINK*
  - CAPI*
- Extended topology-aware collectives
- Extended Energy-aware designs and Virtualization Support
- Extended Support for MPI Tools Interface (as in MPI 3.0)
- Extended Checkpoint-Restart and migration support with SCR
- Support for * features will be available in future MVAPICH2 Releases

Thank You!

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

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