An Assessment of MPI 3.x (Part I)

High Performance MPI Support for Clouds with IB and SR-IOV (Part II)

Talk at HP-CAST 25 (Nov 2015)

by

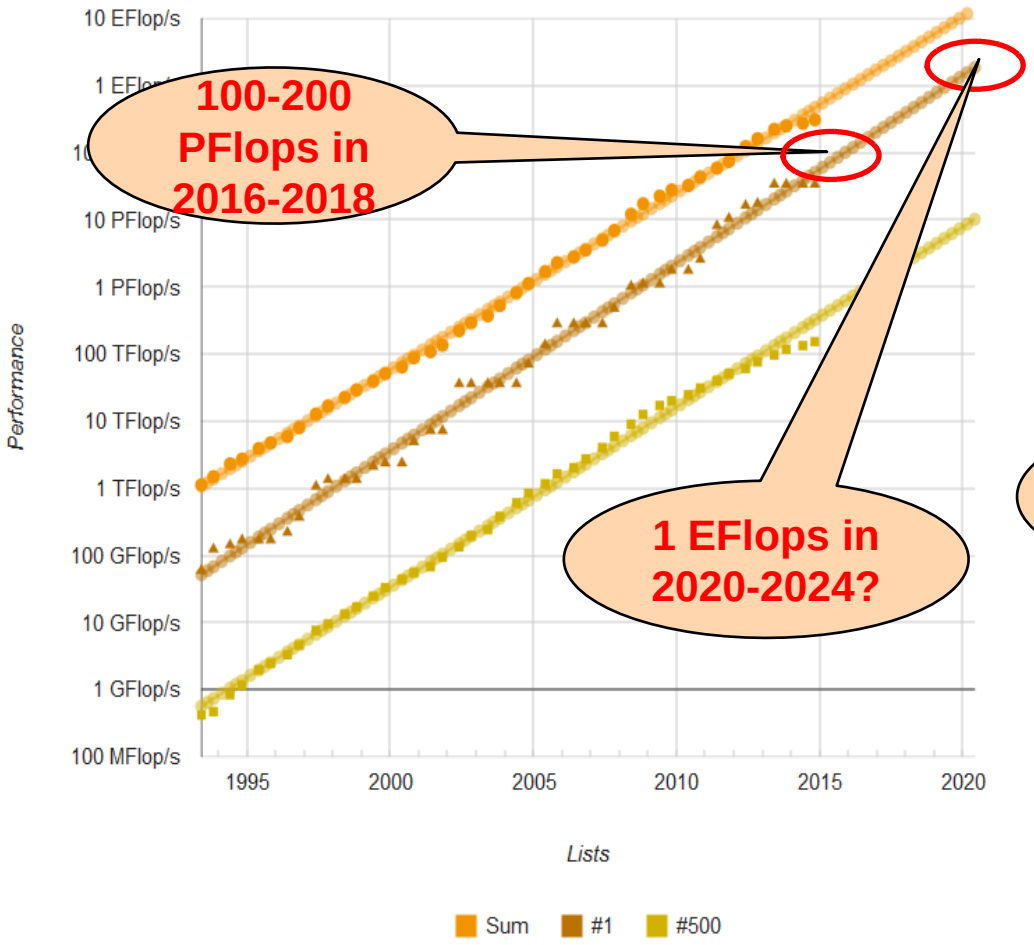
Dhabaleswar K. (DK) Panda

The Ohio State University

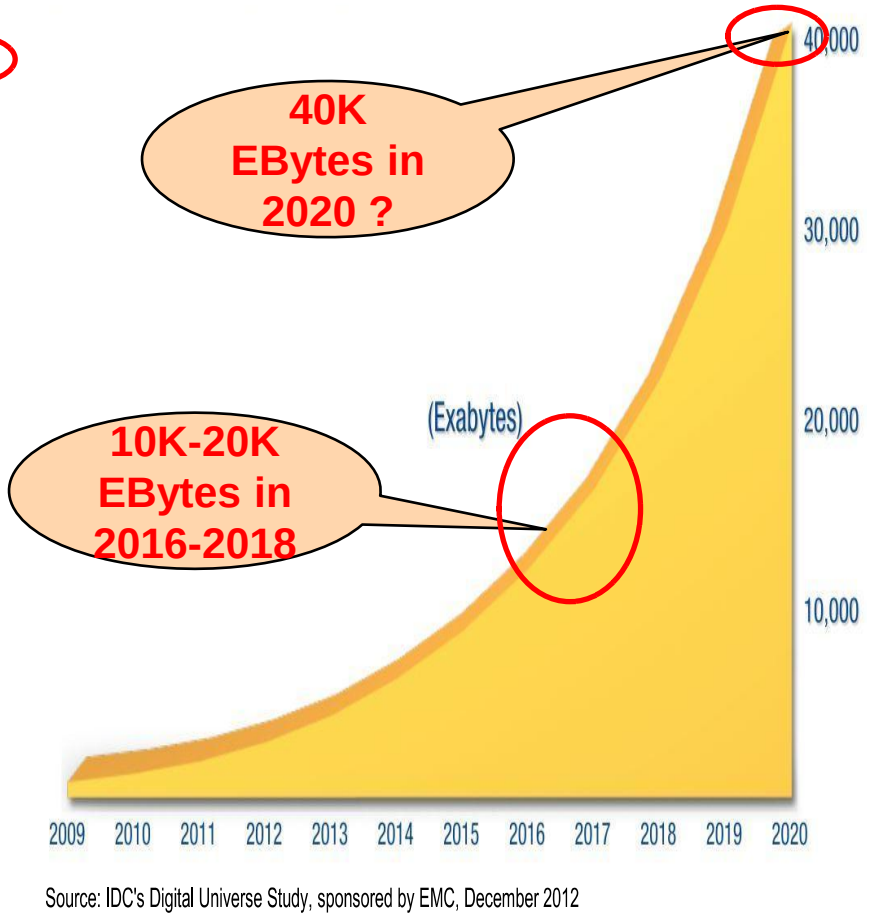
E-mail: panda@cse.ohio-state.edu

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

High-End Computing (HEC): ExaFlop & ExaByte



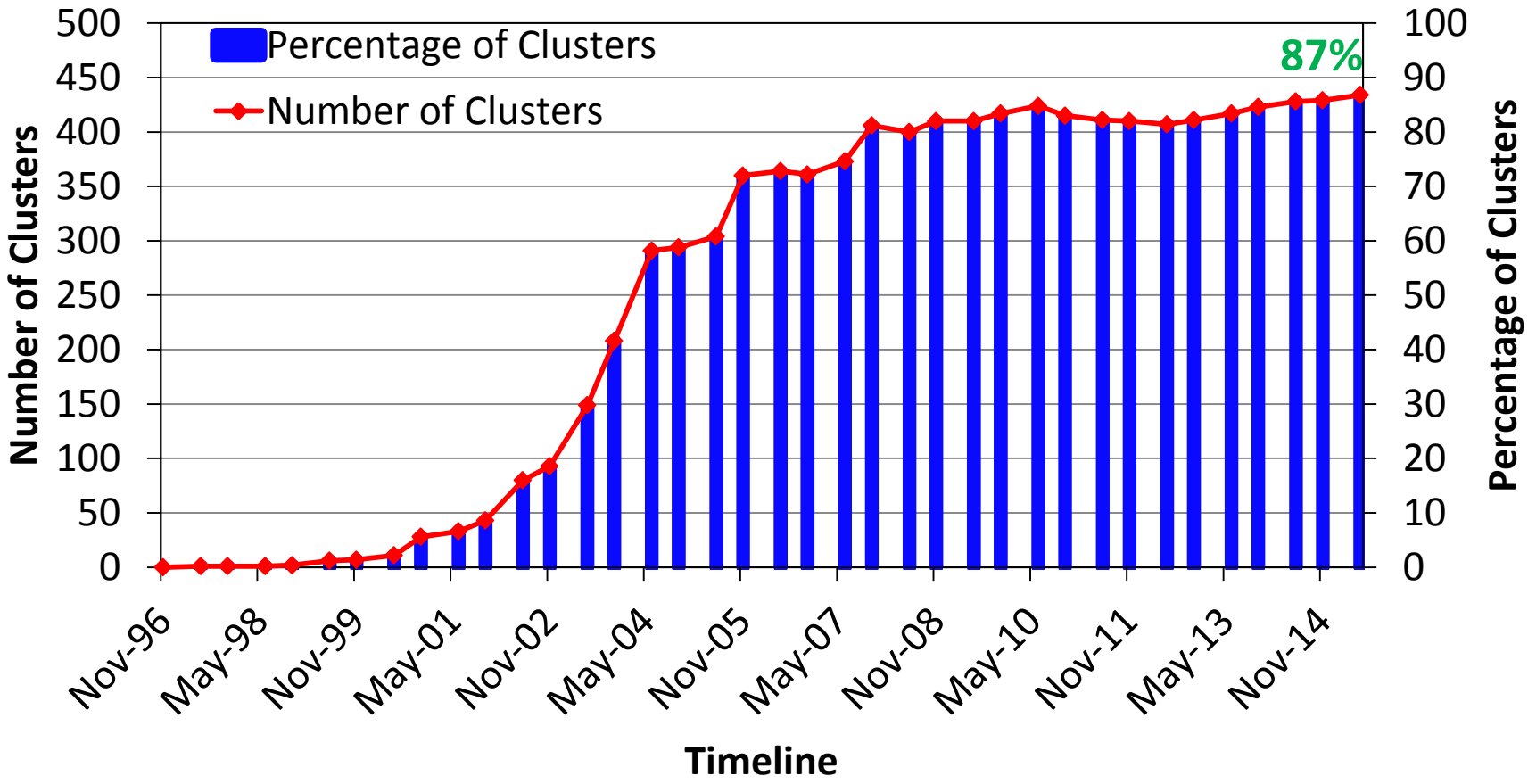
ExaFlop & HPC



Source: IDC's Digital Universe Study, sponsored by EMC, December 2012

ExaByte & BigData

Trends for Commodity Computing Clusters in the Top 500 List (http://www.top500.org)



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

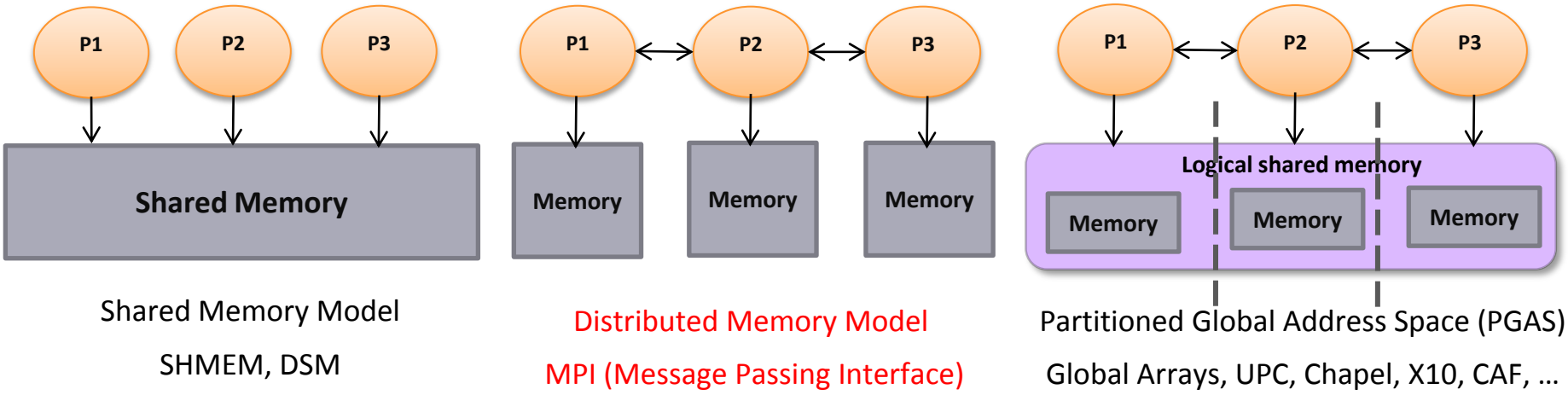
Tianhe – 1A

Large-scale InfiniBand Installations

- 259 IB Clusters (51%) in the June 2015 Top500 list (http://www.top500.org)
- Installations in the Top 50 (24 systems):

519,640 cores (Stampede) at TACC (8th)	76,032 cores (Tsubame 2.5) at Japan/GSIC (22nd)
185,344 cores (Pleiades) at NASA/Ames (11th)	194,616 cores (Cascade) at PNNL (25th)
72,800 cores Cray CS-Storm in US (13th)	76,032 cores (Makman-2) at Saudi Aramco (28th)
72,800 cores Cray CS-Storm in US (14th)	110,400 cores (Pangea) in France (29th)
265,440 cores SGI ICE at Tulip Trading Australia (15th)	37,120 cores (Lomonosov-2) at Russia/MSU (31st)
124,200 cores (Topaz) SGI ICE at ERDC DSRC in US (16th)	57,600 cores (SwiftLucy) in US (33rd)
72,000 cores (HPC2) in Italy (17th)	50,544 cores (Occigen) at France/GENCI-CINES (36th)
115,668 cores (Thunder) at AFRL/USA (19th)	76,896 cores (Salomon) SGI ICE in Czech Republic (40th)
147,456 cores (SuperMUC) in Germany (20th)	73,584 cores (Spirit) at AFRL/USA (42nd)
86,016 cores (SuperMUC Phase 2) in Germany (21st)	**and many more!**

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.

MPI Overview and History

- Message Passing Library standardized by MPI Forum
  - C and Fortran
- Goal: portable, efficient and flexible standard for writing parallel applications
- Not IEEE or ISO standard, but widely considered “industry standard” for HPC application
- Evolution of MPI
  - MPI-1: 1994
  - MPI-2: 1996
  - MPI-3.0: 2008 – 2012, standardized before SC ’12
  - MPI-3.1, standardized on June 4, 2015
  - Next plans for MPI 4.0

Major MPI Features

- Point-to-point Two-sided Communication
- Collective Communication
- One-sided Communication
- Job Startup
- Parallel I/O

How does MPI Plan to Meet Exascale Challenges?

- Power required for data movement operations is one of the main challenges
- Non-blocking collectives
  - Overlap computation and communication
- Much improved One-sided interface
  - Reduce synchronization of sender/receiver
- Manage concurrency
  - Improved interoperability with PGAS (e.g. UPC, Global Arrays, OpenSHMEM)
- Resiliency
  - New interface for detecting failures

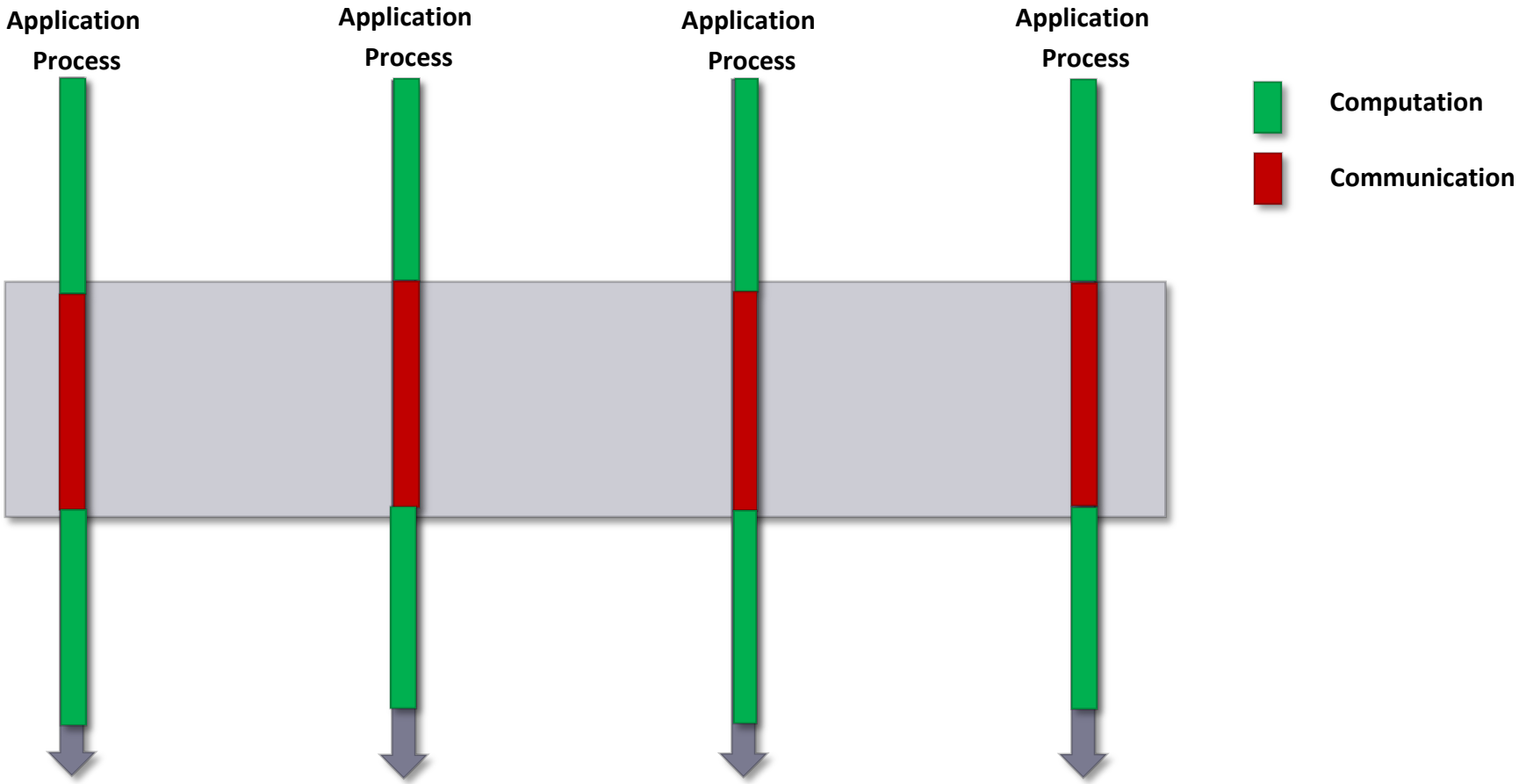
Presentation Outline

- MPI 3.0 Features
- MPI 3.1 Features
- MPI 4.0 Features
- Upcoming Trends for PGAS and Hybrid MPI+PGAS
- Challenges in Supporting MPI, PGAS and Hybrid MPI+PGAS Features and Solutions

Major New Features in MPI-3

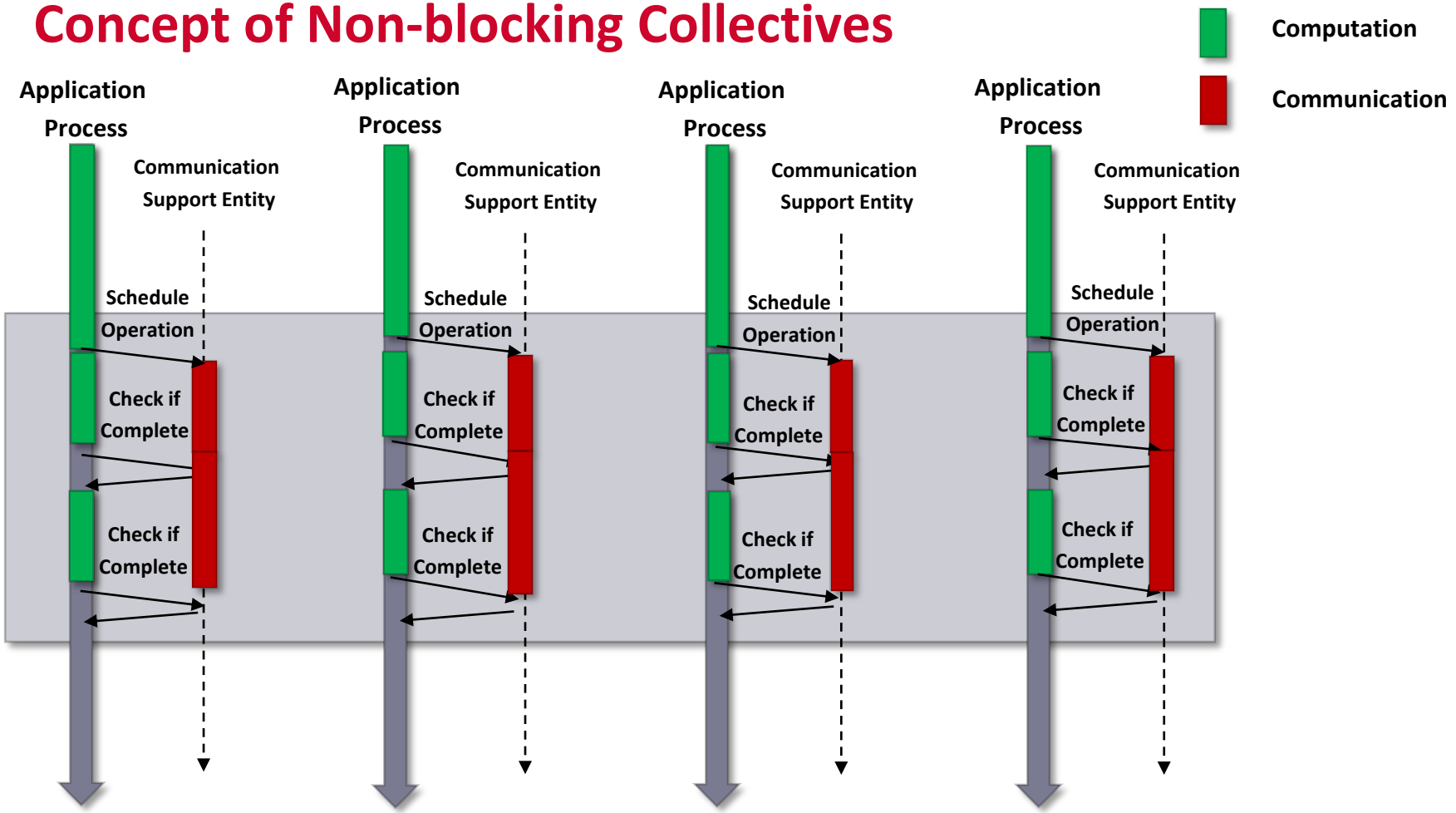
- Major Features
  - Non-blocking Collectives
  - Improved One-Sided (RMA) Model
  - MPI Tools Interface
- Specification is available from: http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf

Problems with Blocking Collective Operations



- Communication time cannot be used for compute
  - No overlap of computation and communication
  - Inefficient

Concept of Non-blocking Collectives



- Application processes schedule collective operation
- Check periodically if operation is complete
- **Overlap of computation and communication => Better Performance**
- *Catch: Who will progress communication*

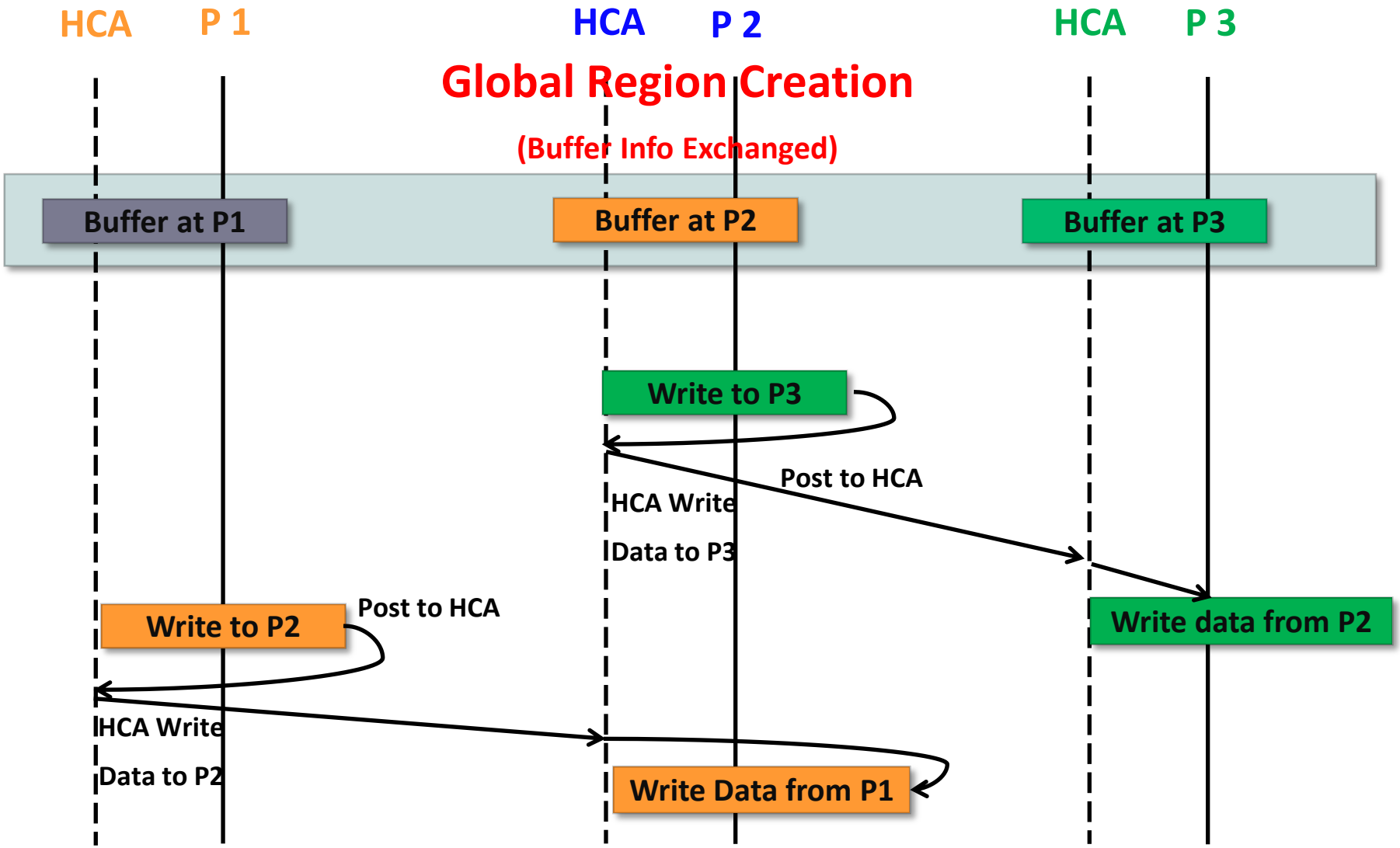
How do I write applications with NBC?

```
void main()
{
    MPI_Init()
    …..
    MPI_Ialltoall(…)
    Computation that does not depend on result of Alltoall
    MPI_Test(for Ialltoall) /* Check if complete (non-blocking) */
    Computation that does not depend on result of Alltoall
    MPI_Wait(for Ialltoall) /* Wait till complete (Blocking) */
    …
    MPI_Finalize()
}
```

MPI-3 Features

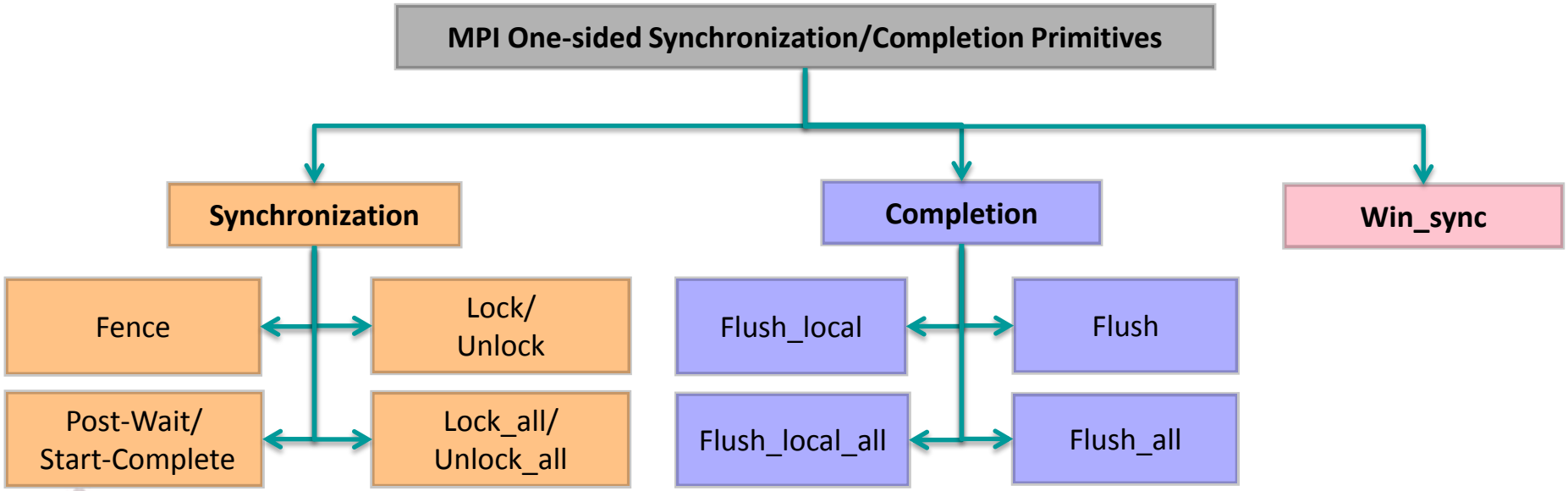
- Non-blocking Collectives
- Improved One-Sided (RMA) Model
- MPI Tools Interface

One-sided Communication Model



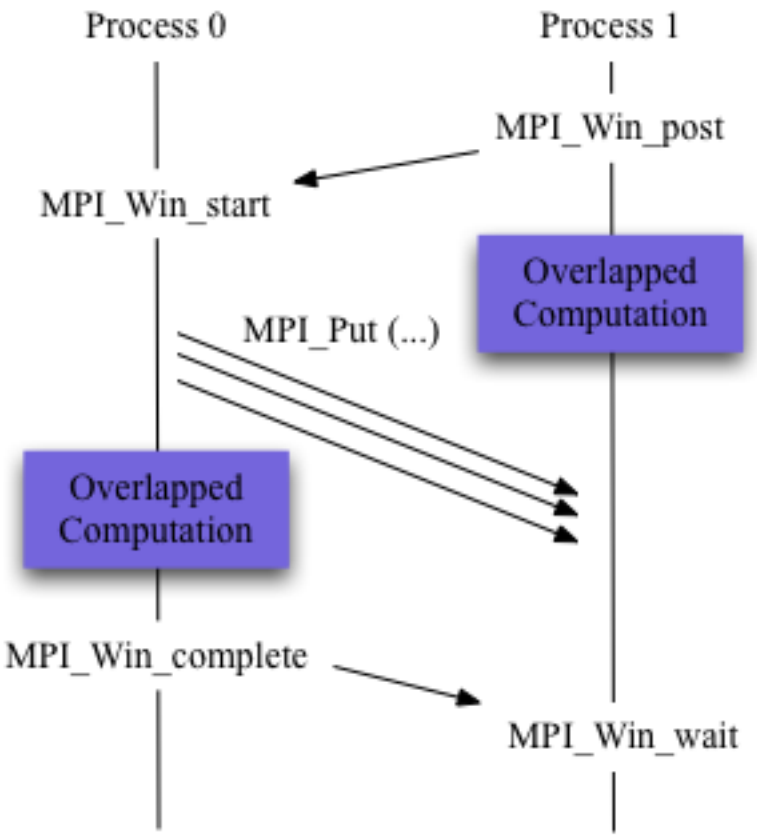
MPI-3 One-Sided Primitives

- Non-blocking one-sided communication routines
  - Put, Get
  - Accumulate, Get_accumulate
  - Atomics
- Flexible synchronization operations to control initiation and completion



Overlapping Communication with MPI-3-RMA

- Network adapters can provide RDMA feature that doesn't require software involvement at remote side
- As long as puts/gets are executed as soon as they are issued, overlap can be achieved
- RDMA-based implementations do just that



MPI-3 Features

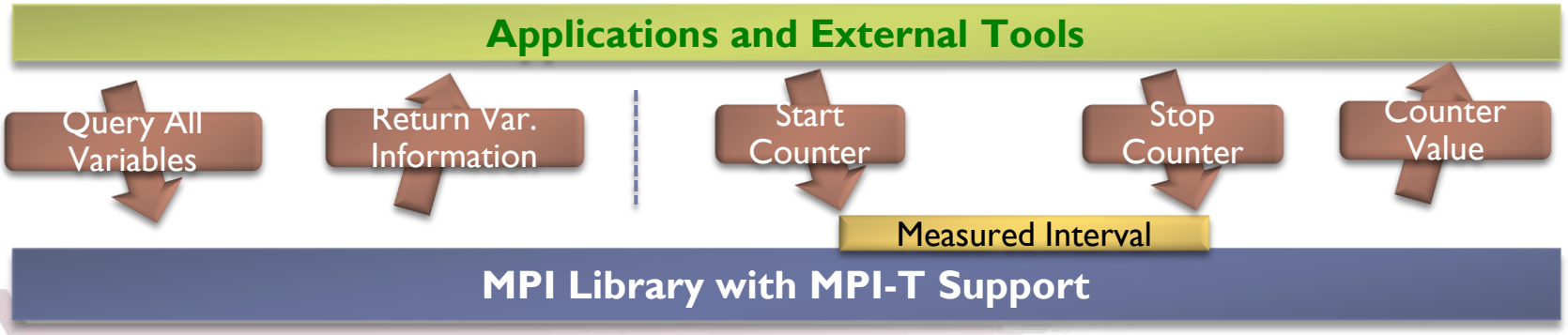
- Non-blocking Collectives
- Improved One-Sided (RMA) Model
- MPI Tools Interface

MPI Tools Interface

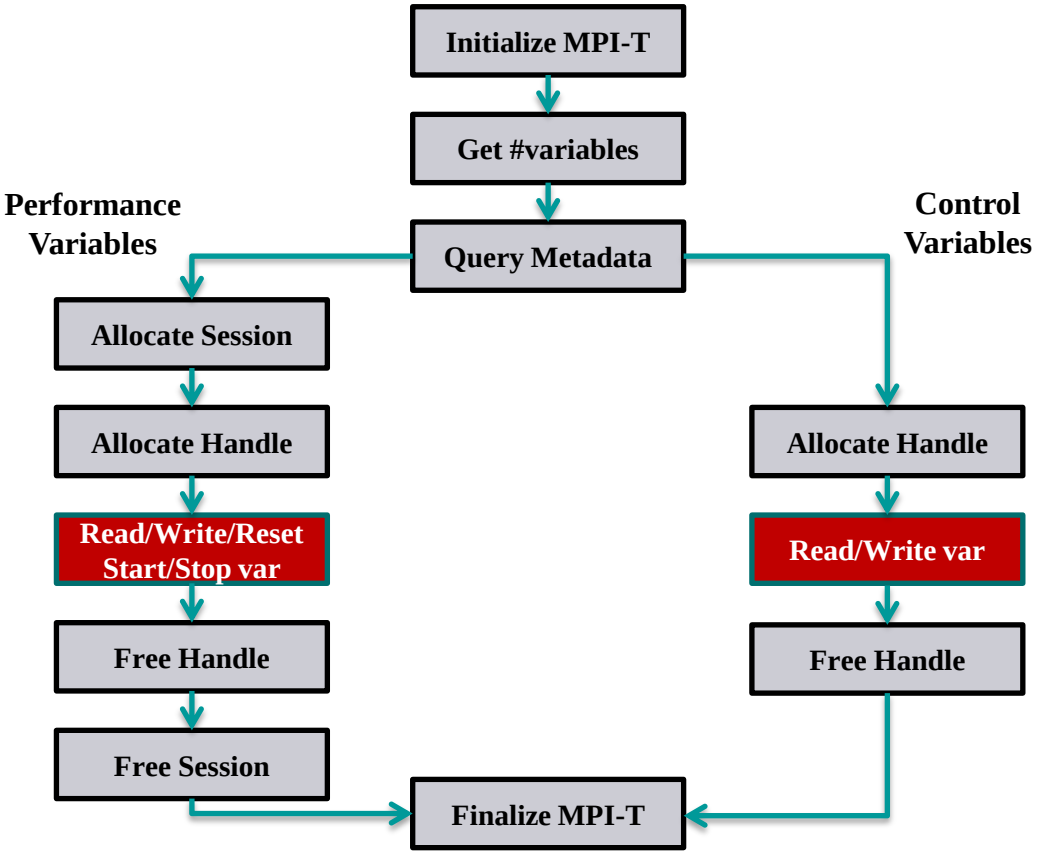
- Introduced to expose internals of MPI tools and applications
- Generalized interface – no defined variables in the standard
- Variables can differ between
  - MPI implementations
  - Compilations of same MPI library (production vs debug)
  - Executions of the same application/MPI library
  - There could be no variables provided
- Two types of variables supported
  - **Control Variables (CVARS)**
    - Typically used to configure and tune MPI internals
    - Environment variables, configuration parameters and toggles
  - **Performance Variables (PVARS)**
    - Insights into performance of an MPI library
    - Highly-implementation specific
    - Memory consumption, timing information, resource-usage, data transmission info.
    - Per-call basis or an entire MPI job

Who should use MPI-T and How?

- Who???
  - Interface intended for tool developers
    - Generally will do *anything* to get the data
    - Are willing to support the many possible variations
- How???
  - Can be called from user code
  - Useful for setting control variables for performance
  - Documenting settings for understanding performance
  - Care must be taken to avoid code that is not portable
  - Several workflows based on role: End Users / Performance Tuners / MPI Implementers
    - Two main workflows
      - Transparently using MPIT-Aware external tools
      - Co-designing applications and MPI-libraries using MPI-T



Co-designing Applications to use MPI-T



Example: Optimizing the eager limit dynamically ->

```
MPI_T_init_thread()
MPI_T_cvar_get_info(MV2_EAGER_THRESHOLD)
if (msg_size < MV2_EAGER_THRESHOLD + 1KB)
        MPI_T_cvar_write(MV2_EAGER_THRESHOLD, +1024)
MPI_Send(..)
MPI_T_finalize()
```

Presentation Outline

- MPI 3.0 Features
- MPI 3.1 Features
- MPI 4.0 Features
- Upcoming Trends for PGAS and Hybrid MPI+PGAS
- Challenges in Supporting MPI, PGAS and Hybrid MPI+PGAS Features and Solutions

MPI-3.1 – Features

- Standardized in June 2015 meeting. Available from http://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf
- Primarily Errata Items
  - Datatype Allowed in Overlapping Accumulates
  - Deprecate MPI_Cancel for send requests
  - Correcting Shared Memory Access with RMA
  - Add Immediate for Non-blocking Collective I/O routines
    - Similar to Non-blocking Collective communication operations
    - E.g. MPI_File_iread_all (MPI_File file, void *buf, int count, MPI_Datatype type, MPI_Request *req);

Presentation Outline

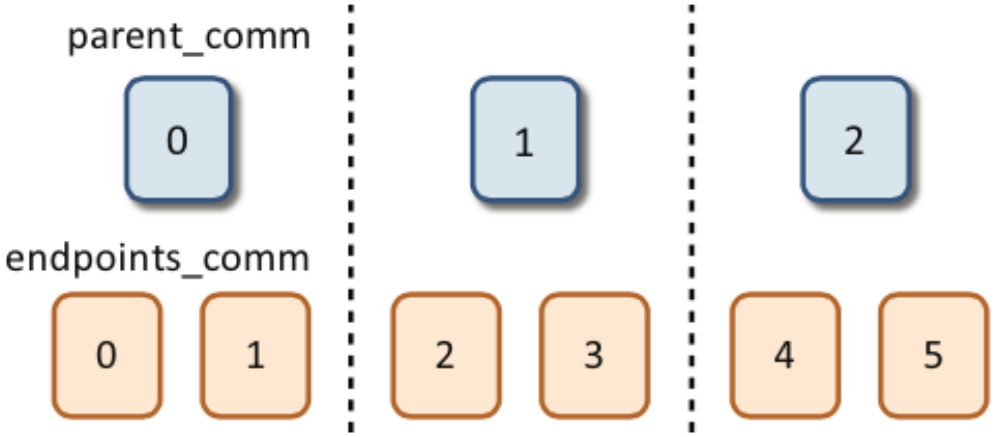
- MPI 3.0 Features
- MPI 3.1 Features
- MPI 4.0 Features
- Upcoming Trends for PGAS and Hybrid MPI+PGAS
- Challenges in Supporting MPI, PGAS and Hybrid MPI+PGAS Features and Solutions

MPI-4.0 - Proposed Features

- Dynamic Endpoints
- Persistent collective operations
  - Explore “streams” and “channels” to cut overhead
  - Explore I/O persistence
- Better support hybrid programming models
- Support for fault tolerance
  - User Level Failure Migration (ULFM)
  - Fault Tolerance for MPI – (Re-init)
- Performance Assertions and Hints
- Enhancements to RMA/One-sided communication
- Large Datatype Support

Dynamic Endpoints

- Introduces a new communicator creation function
- Can be used to create additional ranks, or endpoints, at an existing MPI process
- Endpoints behave the same as processes and can be associated with threads, allowing threads to fully participate in MPI operation



Persistent Collectives

- Use concept of “streams” and “channels” to cut the overhead
- Explore I/O persistence

User Level Failure Mitigation (ULFM)

- Allows the application react to failures but maintain a minimal code path for failure-free execution
- Processes can invalidate and shrink communication objects and prevent waiting indefinitely
- Failures do not alter state of MPI communicators
- Allows point-to-point communication to continue between non-faulty processes

RMA Enhancements

- Enhance notification in passive synchronization
  - Add notification counter associated with the window is incremented at the target after each epoch
  - Processes can query number of notifications received

Immediate Non-blocking I/O

- Proposed non-blocking I/O semantics use MPI_Request
- MPI_Test is used to check for completion

Very Large Datatype Support

- Current datatype interface supports count up to INT_MAX
- Proposed function MPI_Type_contiguous_x allows the creation of very large contiguous datatypes

MPI-4.0 - Additional Features

- Active Messages
- Tools: Extensions for MPI_T
- Generalized Requests
- MPI+X (special focus on MPI+CAF)
- New assertions for passive-target epochs
- MPI Plans (An alternate strategy for collective communication)

Presentation Outline

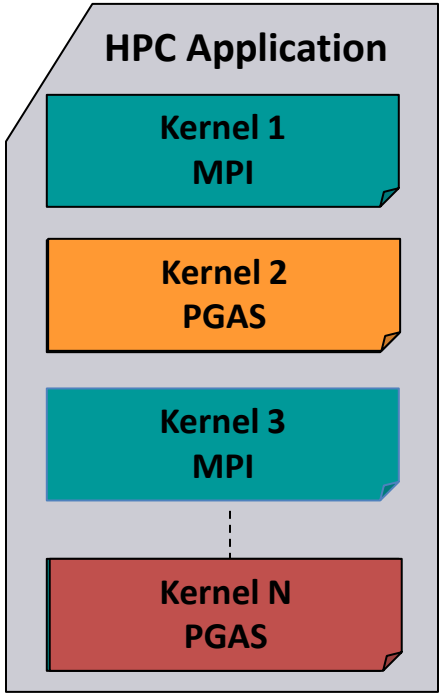
- MPI 3.0 Features
- MPI 3.1 Features
- MPI 4.0 Features
- Upcoming Trends for PGAS and Hybrid MPI+PGAS
- Challenges in Supporting MPI, PGAS and Hybrid MPI+PGAS Features and Solutions

Partitioned Global Address Space (PGAS) Models

- Key features
  - Simple shared memory abstractions
  - Light weight one-sided communication
  - Easier to express irregular communication
- Different approaches to PGAS
  - Languages
    - Unified Parallel C (UPC)
    - Co-Array Fortran (CAF)
    - X10
    - Chapel
  - Libraries
    - OpenSHMEM
    - Global Arrays

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"

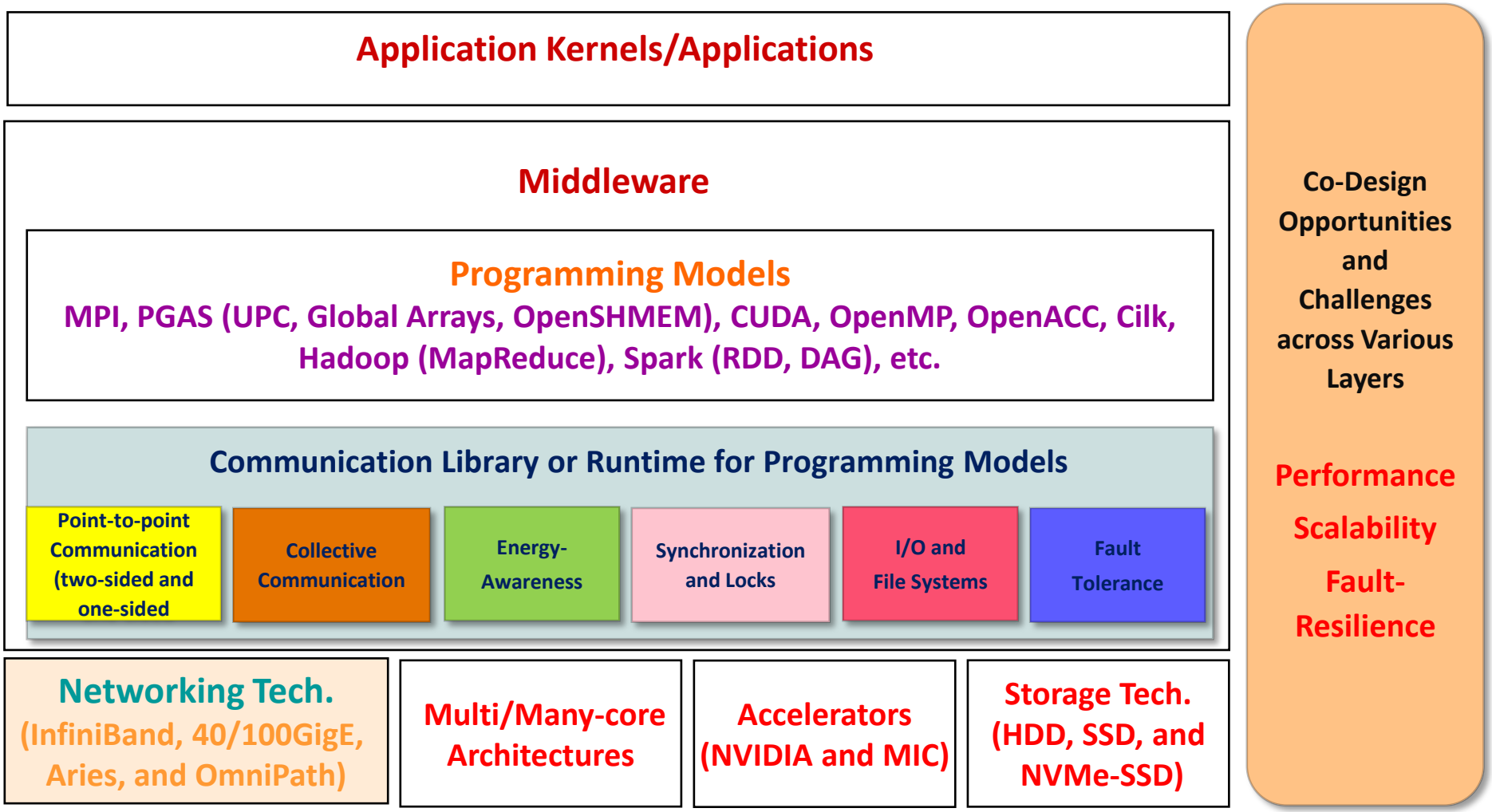


** The International Exascale Software Roadmap, Dongarra, J., Beckman, P. et al., Volume 25, Number 1, 2011, International Journal of High Performance Computer Applications, ISSN 1094-3420*

Presentation Outline

- MPI 3.0 Features
- MPI 3.1 Features
- MPI 4.0 Features
- Upcoming Trends for PGAS and Hybrid MPI+PGAS
- Challenges in Supporting MPI, PGAS and Hybrid MPI+PGAS Features and Solutions

Designing High-Performance Middleware for HPC and Big Data Applications: Challenges



Broad Challenges in Designing Communication Libraries for (MPI+X) at 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
  - Offload
  - Non-blocking
  - Topology-aware
- Balancing intra-node and inter-node communication for next generation multi-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, …)
- Virtualization
- Energy-Awareness

MVAPICH2 Software

- High Performance open-source MPI Library for InfiniBand, 10-40Gig/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 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
  - **Used by more than 2,475 organizations in 76 countries**
  - **More than 300,000 downloads from the OSU site directly**
  - Empowering many TOP500 clusters (June '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 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 (8th in Jun'15, 519,640 cores, 5.168 Plops)

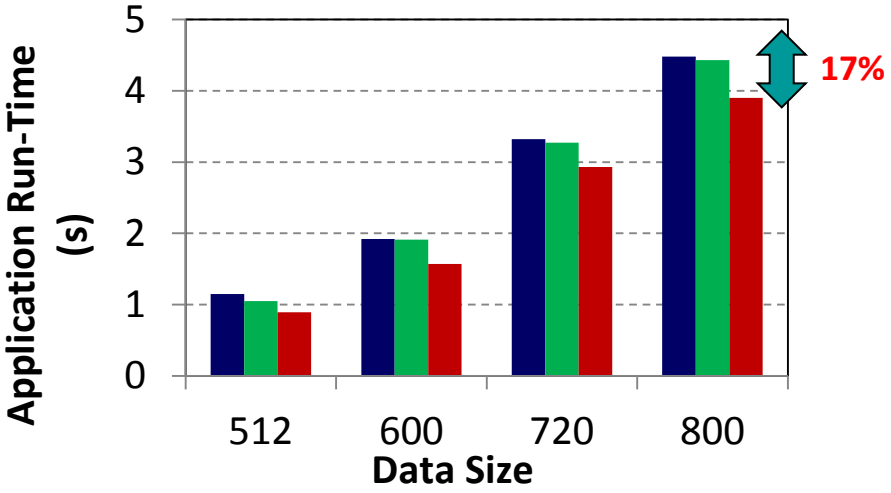
MVAPICH2 Software Family

Requirements	MVAPICH2 Library to use
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

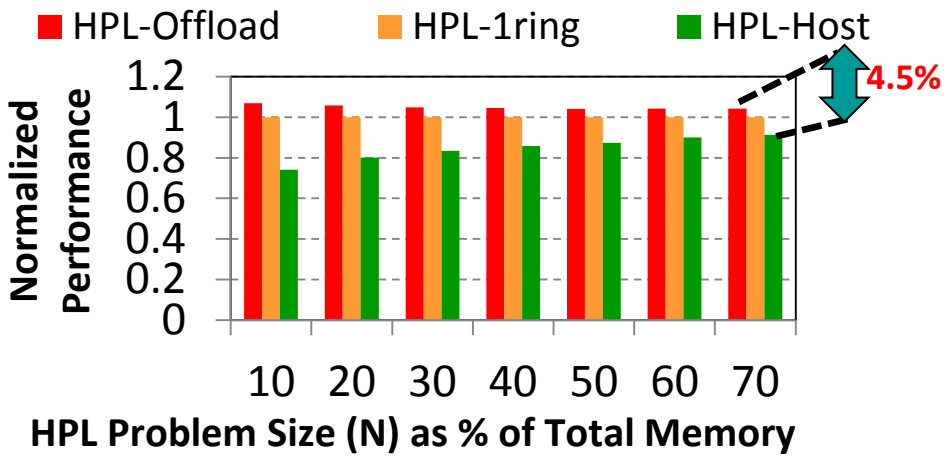
Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-X for Exascale

- Non-blocking Collectives
- Support for MPI-T Interface
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, …) with Unified Runtime
- Support for Virtualization with SR-IOV

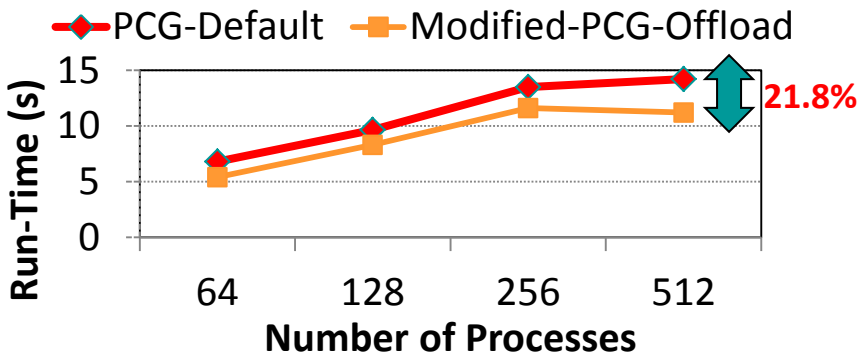
Co-Design with MPI-3 Non-Blocking Collectives and Collective Offload Co-Direct Hardware (Available since MVAPICH2-X 2.2a)



Modified P3DFFT with Offload-Alltoall does up to 17% better than default version (128 Processes)



Modified HPL with Offload-Bcast does up to 4.5% better than default version (512 Processes)



Modified Pre-Conjugate Gradient Solver with Offload-Allreduce does up to 21.8% better than default version

K. Kandalla, et. al.. High-Performance and Scalable Non-Blocking All-to-All with Collective Offload on InfiniBand Clusters: A Study with Parallel 3D FFT. ISC 2011

K. Kandalla, et. al, Designing Non-blocking Broadcast with Collective Offload on InfiniBand Clusters: A Case Study with HPL, HotI 2011

K. Kandalla, et. al., Designing Non-blocking Allreduce with Collective Offload on InfiniBand Clusters: A Case Study with Conjugate Gradient Solvers, IPDPS '12

Can Network-Offload based Non-Blocking Neighborhood MPI Collectives Improve Communication Overheads of Irregular Graph Algorithms? K. Kandalla, A. Buluc, H. Subramoni, K. Tomko, J. Vienne, L. Oliker, and D. K. Panda, IWPAPS' 12

MPI-T Support in MVAPICH2

Memory Usage:
- current level
- maximum watermark

InfiniBand N/W:
- #control packets
- #out-of-order packets

Pt-to-pt messages:
- unexpected queue length
- unexp. match attempts
- recvq. length

Registration cache:
- hits
- misses

Shared-memory:
- LiMIC2/ CMA
- buffer pool size & usage

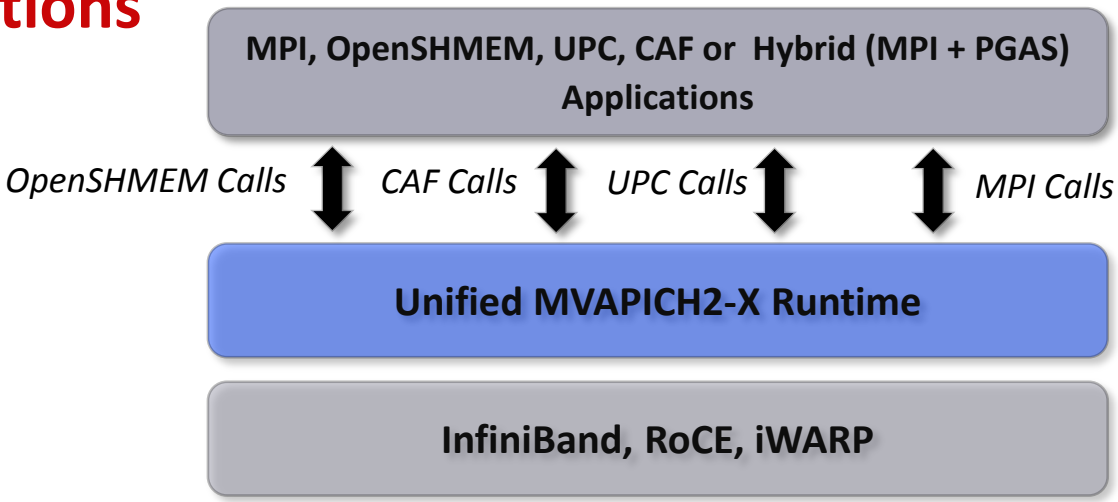
Collective ops:
- comm. creation
- #algorithm invocations

[Bcast – 8; Gather – 10]

…

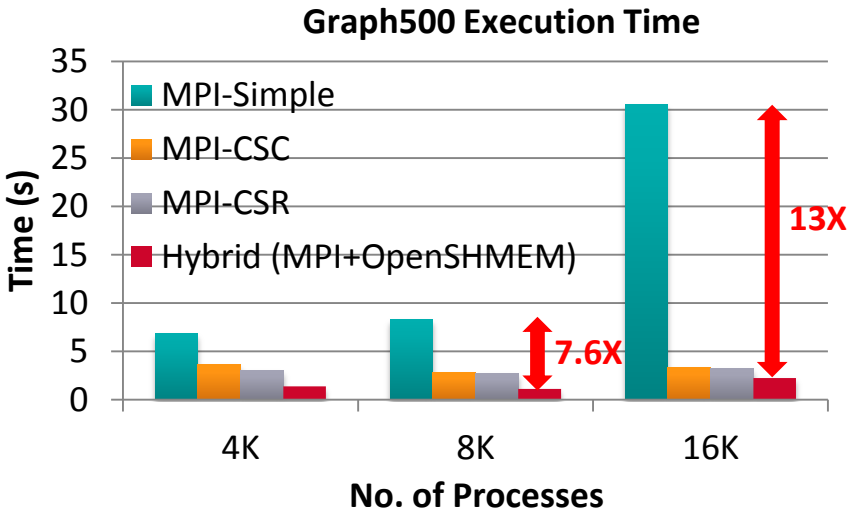
- Initial focus on performance variables
- Variables to track different components
  - MPI library's internal memory usage
  - Unexpected receive queue
  - Registration cache
  - VBUF allocation
  - Shared-memory communication
  - Collective operation algorithms
  - IB channel packet transmission
  - Many more in progress..

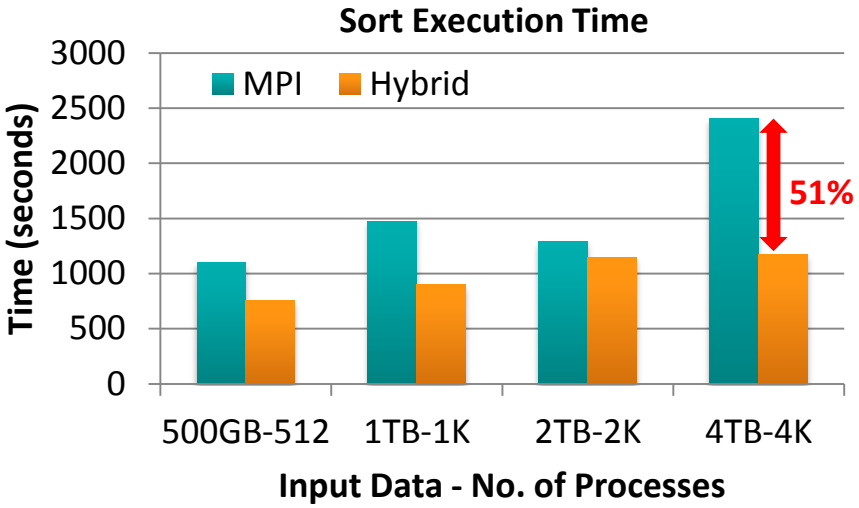
MVAPICH2-X for Advanced MPI and Hybrid MPI + PGAS Applications



- Unified communication runtime for MPI, UPC, OpenSHMEM, CAF available with MVAPICH2-X 1.9 (2012) onwards!
  - http://mvapich.cse.ohio-state.edu
- Feature Highlights
  - Supports MPI(+OpenMP), OpenSHMEM, UPC, CAF, MPI(+OpenMP) + OpenSHMEM, MPI(+OpenMP) + UPC
  - MPI-3 compliant, OpenSHMEM v1.0 standard compliant, UPC v1.2 standard compliant (with initial support for UPC 1.3), CAF 2008 standard (OpenUH)
  - Scalable Inter-node and intra-node communication – point-to-point and collectives

Application Level Performance with Graph500 and Sort





- Performance of Hybrid (MPI+ OpenSHMEM) Graph500 Design
  - 8,192 processes
    - **2.4X** improvement over MPI-CSR
    - **7.6X** improvement over MPI-Simple
  - 16,384 processes
    - **1.5X** improvement over MPI-CSR
    - **13X** improvement over MPI-Simple

- Performance of Hybrid (MPI+OpenSHMEM) Sort Application
  - 4,096 processes, 4 TB Input Size
    - MPI – 2408 sec; 0.16 TB/min
    - Hybrid – 1172 sec; 0.36 TB/min
    - **51%** improvement over MPI-design

J. Jose, S. Potluri, H. Subramoni, X. Lu, K. Hamidouche, K. W. Schulz, H. Sundar, D. K. Panda, Designing Scalable Out-of-core Sorting with Hybrid MPI+PGAS Programming Models, PGAS '14, Oct. 2014.

J. Jose, S. Potluri, K. Tomko and D. K. Panda, Designing Scalable Graph500 Benchmark with Hybrid MPI+OpenSHMEM Programming Models, International Supercomputing Conference (ISC'13), June 2013

Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-X for Exascale

- Non-blocking Collectives
- Support for MPI-T Interface
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, …) with Unified Runtime
- Support for Virtualization with SR-IOV

HPC Cloud - Combining HPC with Cloud

- IDC expects that by 2017, HPC ecosystem revenue will jump to a record $30.2 billion. IDC foresees public clouds, and especially custom public clouds, supporting an increasing proportion of the aggregate HPC workload as these cloud facilities grow more capable and mature
  - Courtesy: http://www.idc.com/getdoc.jsp?containerId=247846
- Combining HPC with Cloud is still facing challenges because of the performance overhead associated virtualization support
  - Lower performance of virtualized I/O devices
- HPC Cloud Examples
  - **Amazon EC2 with Enhanced Networking**
    - Using Single Root I/O Virtualization (SR-IOV)
    - Higher performance (packets per second), lower latency, and lower jitter.
    - 10 GigE
  - **NSF Chameleon Cloud**

NSF Chameleon Cloud: A Powerful and Flexible Experimental Instrument

- Large-scale instrument
  - Targeting Big Data, Big Compute, Big Instrument research
  - ~650 nodes (~14,500 cores), 5 PB disk over two sites, 2 sites connected with 100G network
  - Virtualization technology (e.g., SR-IOV, accelerators), systems, networking (InfiniBand), infrastructure-level resource management, etc.
- Reconfigurable instrument
  - Bare metal reconfiguration, operated as single instrument, graduated approach for ease-of-use
- Connected instrument
  - Workload and Trace Archive
  - Partnerships with production clouds: CERN, OSDC, Rackspace, Google, and others
  - Partnerships with users
- Complementary instrument
  - Complementing GENI, Grid'5000, and other testbeds
- Sustainable instrument
  - Industry connections

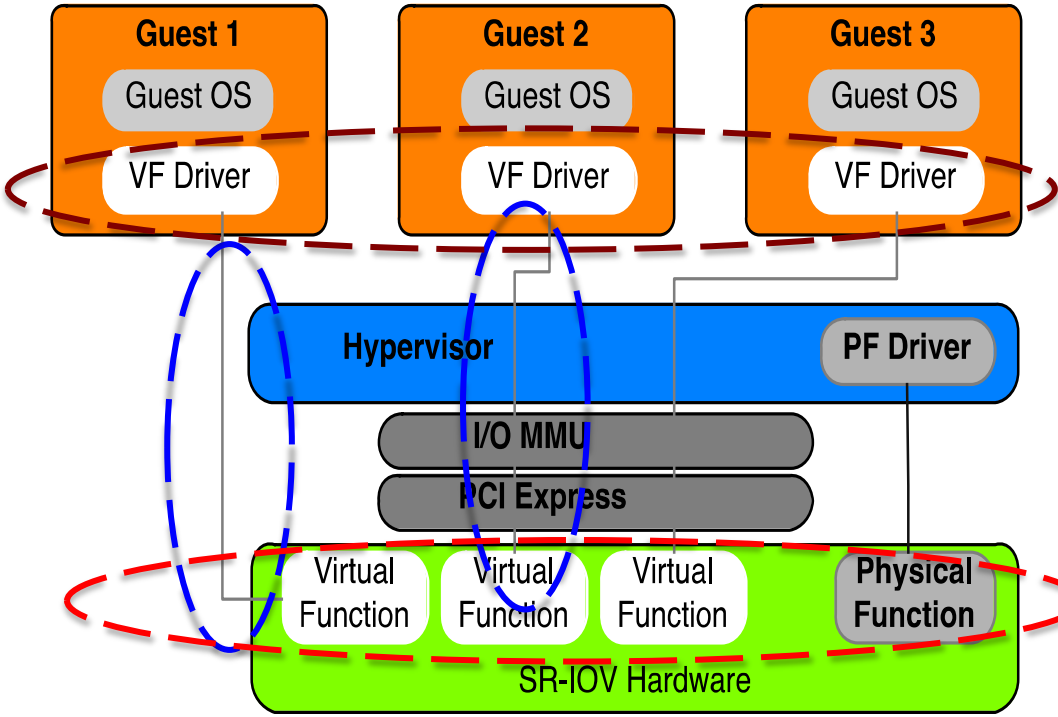
http://www.chameleoncloud.org/

TACC iCAIR

UTSA

Single Root I/O Virtualization (SR-IOV)

- Single Root I/O Virtualization (SR-IOV) is providing new opportunities to design HPC cloud with very little low overhead
  - Allows a single physical device, or a Physical Function (PF), to present itself as multiple virtual devices, or Virtual Functions (VFs)
  - Each VF can be dedicated to a single VM through PCI pass-through
  - VFs are designed based on the existing non-virtualized PFs, no need for driver change
  - Work with 10/40 GigE and InfiniBand

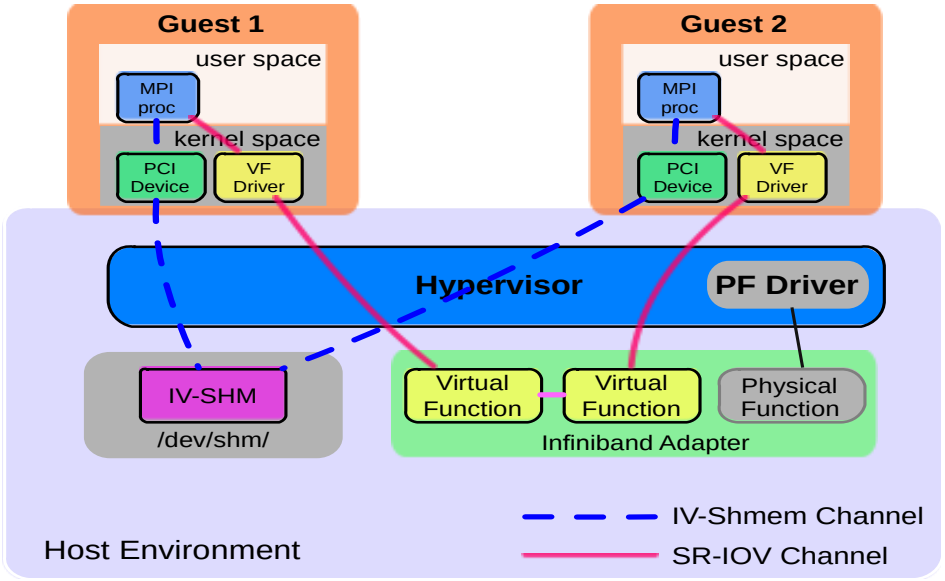


MVAPICH2-Virt: High-Performance MPI Library over SR-IOV capable InfiniBand Clusters

- Support for SR-IOV
  - Inter-node Inter-VM communication
- Locality-aware communication through IVSHMEM
  - Inter-VM Shared Memory (IVSHMEM) is a novel feature proposed for inter-VM communication, and offers shared memory backed communication for VMs within a given host
  - Intra-node Inter-VM communication
- Building efficient HPC Cloud
- Available publicly as MVAPICH2-Virt 2.1 Library

Overview of MVAPICH2-Virt with SR-IOV and IVSHMEM

- Redesign MVAPICH2 to make it virtual machine aware
  - SR-IOV shows near to native performance for inter-node point to point communication
  - IVSHMEM offers zero-copy access to data on shared memory of co-resident VMs
  - Locality Detector: maintains the locality information of co-resident virtual machines
  - Communication Coordinator: selects the communication channel (SR-IOV, IVSHMEM) adaptively

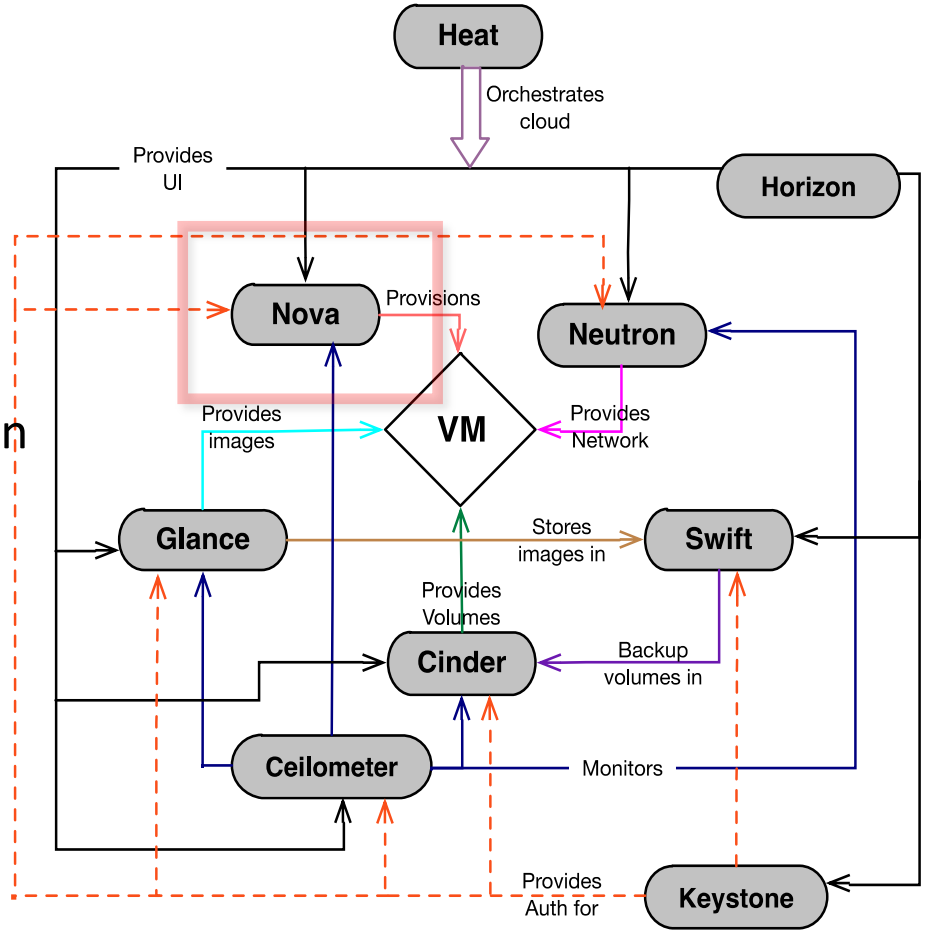


J. Zhang, X. Lu, J. Jose, R. Shi, D. K. Panda. Can Inter-VM Shmem Benefit MPI Applications on SR-IOV based Virtualized InfiniBand Clusters? **Euro-Par**, 2014.

J. Zhang, X. Lu, J. Jose, R. Shi, M. Li, D. K. Panda. High Performance MPI Library over SR-IOV Enabled InfiniBand Clusters. **HiPC**, 2014.

MVAPICH2-Virt with SR-IOV and IVSHMEM over OpenStack

- OpenStack is one of the most popular open-source solutions to build clouds and manage virtual machines
- Deployment with OpenStack
  - Supporting SR-IOV configuration
  - Supporting IVSHMEM configuration
  - Virtual Machine aware design of MVAPICH2 with SR-IOV
- An efficient approach to build HPC Clouds with MVAPICH2-Virt and OpenStack



J. Zhang, X. Lu, M. Arnold, D. K. Panda. MVAPICH2 over OpenStack with SR-IOV: An Efficient Approach to Build HPC Clouds. **CCGrid**, 2015.

Experimental Setup

Cluster	Nowlab Cloud		Amazon EC2	
Instance	4 Core/VM	8 Core/VM	4 Core/VM	8 Core/VM
Platform	RHEL 6.5 Qemu+KVM HVM Slurm 14.11.8		Amazon Linux (EL6) Xen HVM C3.xlarge [1] Instance	Amazon Linux (EL6) Xen HVM C3.2xlarge [1] Instance
CPU	SandyBridge Intel(R) Xeon E5-2670 (2.6GHz)		IvyBridge Intel(R) Xeon E5-2680v2 (2.8GHz)	
RAM	6 GB	12 GB	7.5 GB	15 GB
Interconnect	FDR (56Gbps) InfiniBand Mellanox ConnectX-3 with SR-IOV [2]		10 GigE with Intel ixgbevf SR-IOV driver [2]	

[1] Amazon EC2 C3 instances: compute-optimized instances, providing customers with the highest performing processors, good for HPC workloads

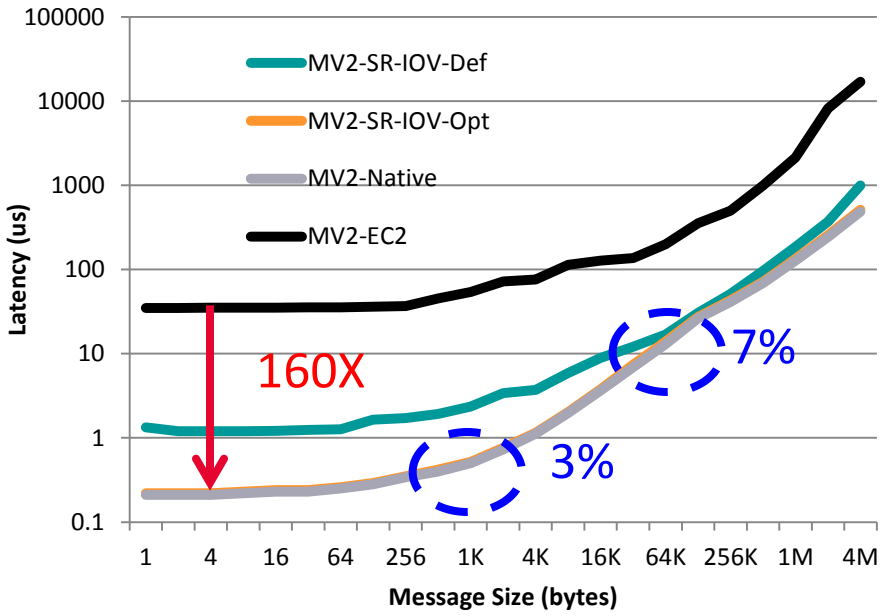
[2] Nowlab Cloud is using InfiniBand FDR (56Gbps), while Amazon EC2 C3 instances are using 10 GigE. Both have SR-IOV support.

Experiments Carried Out

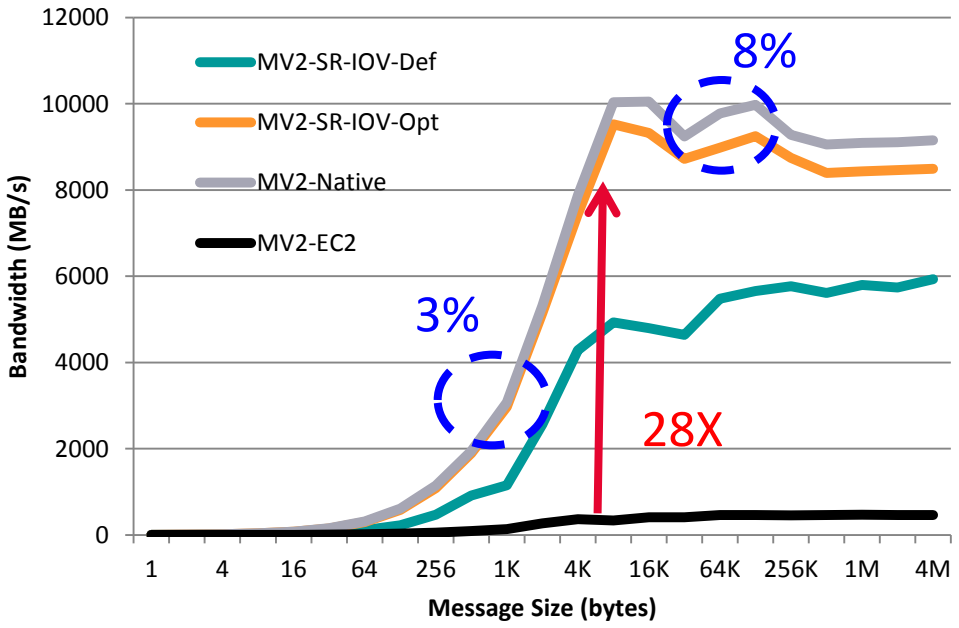
- Point-to-point
  - Two-sided and One-sided
  - Latency and Bandwidth
  - Intra-node and Inter-node [1]
- Applications
  - NAS and Graph500

[1] Amazon EC2 does not support users to explicitly allocate VMs in one physical node so far. We allocate multiple VMs in one logical group and compare the point-to-point performance for each pair of VMs. We see the VMs who have the lowest latency as located within one physical node (Intra-node), otherwise Inter-node.

Point-to-Point Performance – Latency & Bandwidth (Intra-node)



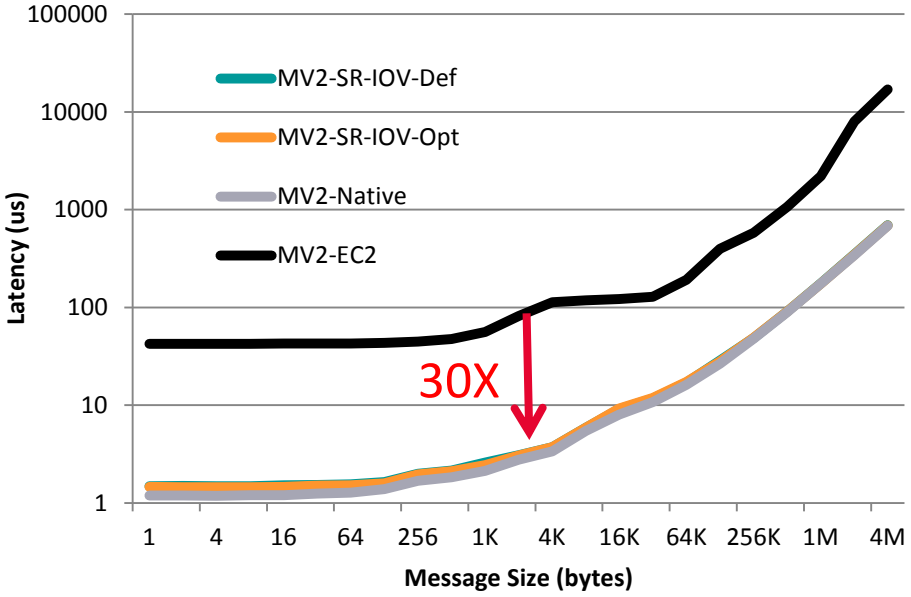
Intra-node Inter-VM pt2pt Latency



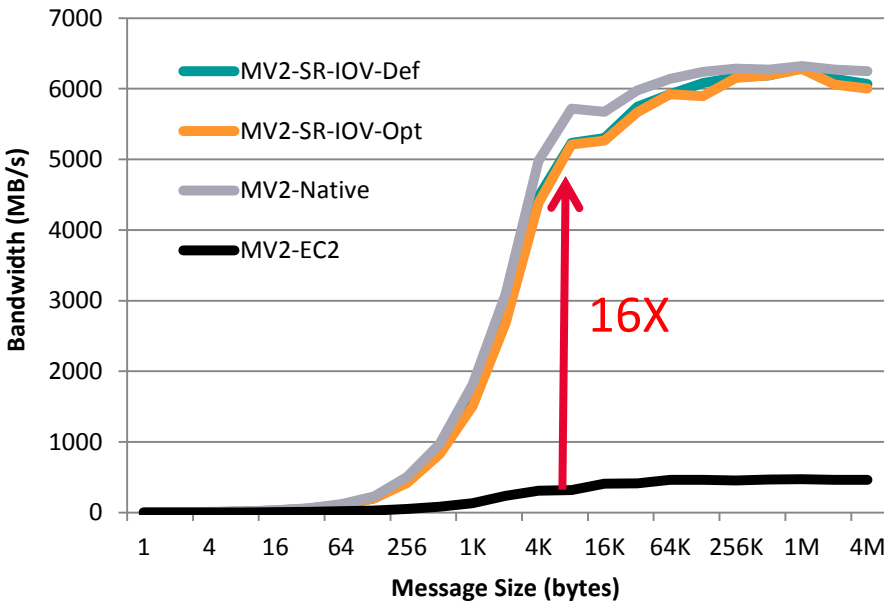
Intra-node Inter-VM pt2pt Bandwidth

- EC2 C3.2xlarge instances
- Compared to SR-IOV-Def, up to 84% and 158% performance improvement on Lat & BW
- Compared to Native, 3%-7% overhead for Lat, 3%-8% overhead for BW
- Compared to EC2, up to 160X and 28X performance speedup on Lat & BW

Point-to-Point Performance – Latency & Bandwidth (Inter-node)



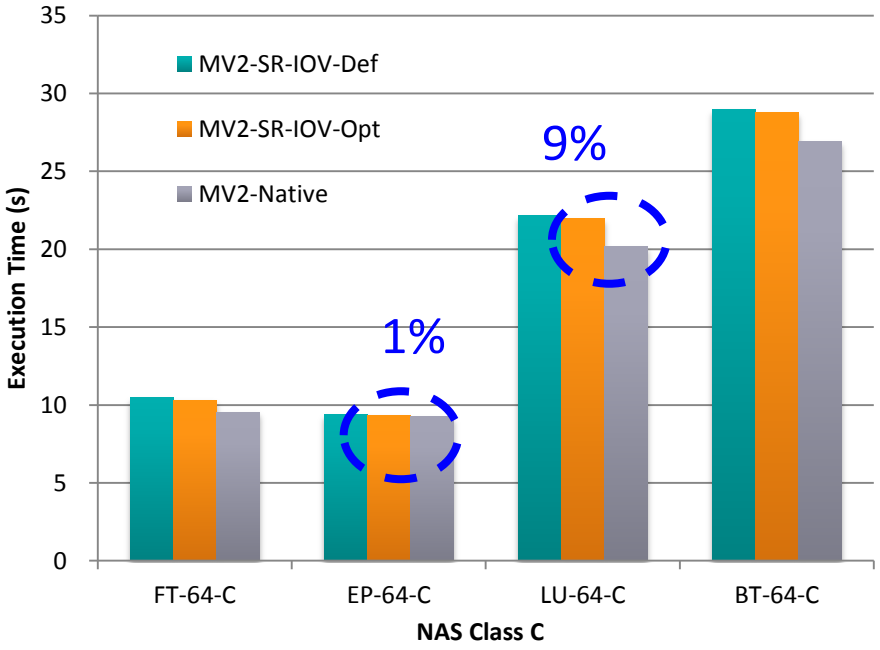
Inter-node Inter-VM pt2pt Latency



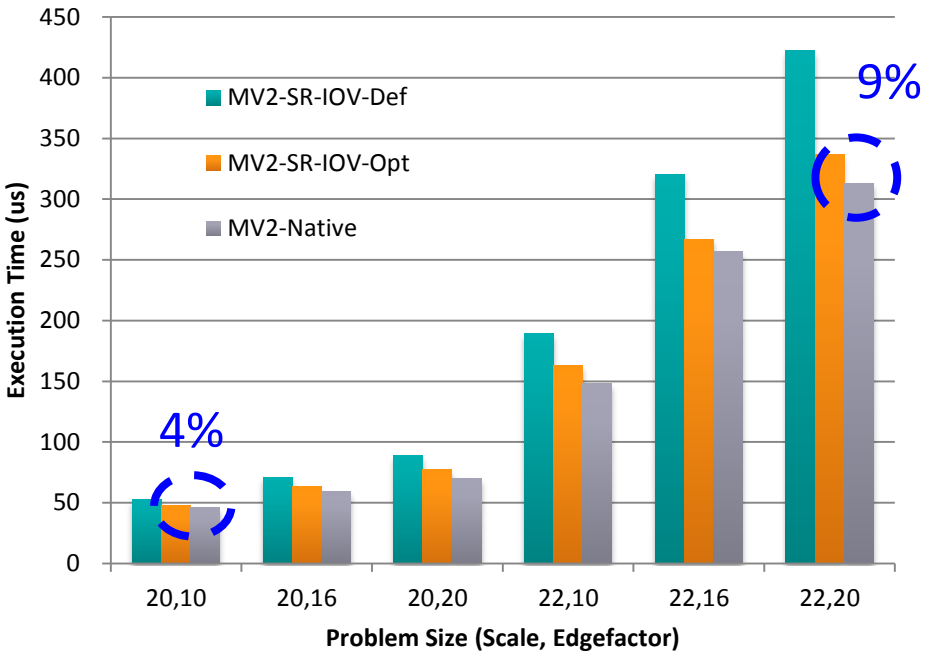
Inter-node Inter-VM pt2pt Bandwidth

- EC2 C3.2xlarge instances
- Similar performance with SR-IOV-Def
- Compared to Native, 2%-8% overhead on Lat & BW for 8KB+ messages
- Compared to EC2, up to 30X and 16X performance speedup on Lat & BW

Application-Level Performance (8 VM * 8 Core/VM)



NAS



Graph500

- Compared to Native, 1-9% overhead for NAS
- Compared to Native, 4-9% overhead for Graph500

MVAPICH2 – Plans for Exascale

- Performance and Memory scalability toward 500K-1M cores
  - Dynamically Connected Transport (DCT) service with Connect-IB
- Hybrid programming (MPI + OpenSHMEM, MPI + UPC, MPI + CAF …)
  - Support for UPC++
- Enhanced Optimization for GPU Support and Accelerators
- Taking advantage of advanced features
  - User Mode Memory Registration (UMR)
  - On-demand Paging
- Enhanced Inter-node and Intra-node communication schemes for upcoming OmniPath enabled Knights Landing architectures
- Extended RMA support (as in MPI 3.0)
- Extended topology-aware collectives
- Energy-aware point-to-point (one-sided and two-sided) and collectives
- Extended Support for MPI Tools Interface (as in MPI 3.0)
- Extended Checkpoint-Restart and migration support with SCR
- Enhanced support with Virtualization

Funding Acknowledgments

Funding Support by

Equipment Support by

Personnel Acknowledgments

Current Students

- A. Augustine (M.S.)
- A. Awan (Ph.D.)
- A. Bhat (M.S.)
- S. Chakraborthy (Ph.D.)
- C.-H. Chu (Ph.D.)
- N. Islam (Ph.D.)
- K. Kulkarni (M.S.)
- M. Li (Ph.D.)
- M. Rahman (Ph.D.)
- D. Shankar (Ph.D.)
- A. Venkatesh (Ph.D.)
- J. Zhang (Ph.D.)

Current Research Scientists

- H. Subramoni
- X. Lu

Current Post-Docs

- J. Lin
- D. Shankar

Current Senior Research Associate

- K. Hamidouche

Current Programmer

- J. Perkins

Current Research Specialist

- M. Arnold

Past Students

- P. Balaji (Ph.D.)
- D. Buntinas (Ph.D.)
- S. Bhagvat (M.S.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- R. Rajachandrasekar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)

Past Post-Docs

- H. Wang
- X. Besseron
- H.-W. Jin
- M. Luo
- E. Mancini
- S. Marcarelli
- J. Vienne

Past Research Scientist

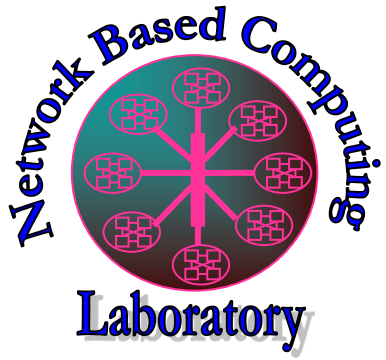
- S. Sur

Past Programmers

- D. Bureddy

Thank You!

panda@cse.ohio-state.edu



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

MVAPICH

The MVAPICH2 Project
http://mvapich.cse.ohio-state.edu/

HiBD
High-Performance
Big Data

The High-Performance Big Data Project
http://hibd.cse.ohio-state.edu/

Multiple events from the Group at SC '15

- Many events
  - Workshop (ESPM2)
  - Tutorials (IB-HSE and Big Data)
  - Talks
  - Presentation-demo-discussion
- Detailed events available from:
  - http://mvapich.cse.ohio-state.edu
- Will be at Ohio Supercomputer Center / OH-TECH Booth (#1209)