MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

Designing MPI and PGAS Libraries for Exascale Systems: The MVAPICH2 Approach

Talk at OpenFabrics Workshop (March 2017)

by

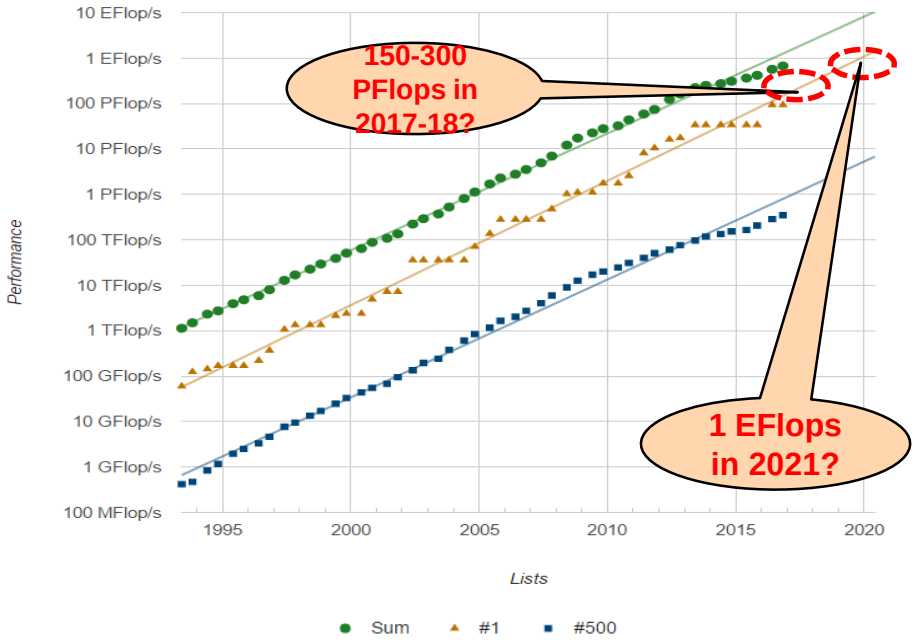
Dhabaleswar K. (DK) Panda

The Ohio State University

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

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

High-End Computing (HEC): Towards Exascale-Era



Expected to have an ExaFlop system in 2021!

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)
- Available on HPC Clouds, e.g., Amazon EC2, NSF Chameleon, Microsoft Azure, etc.

Sunway TaihuLight

K - Computer

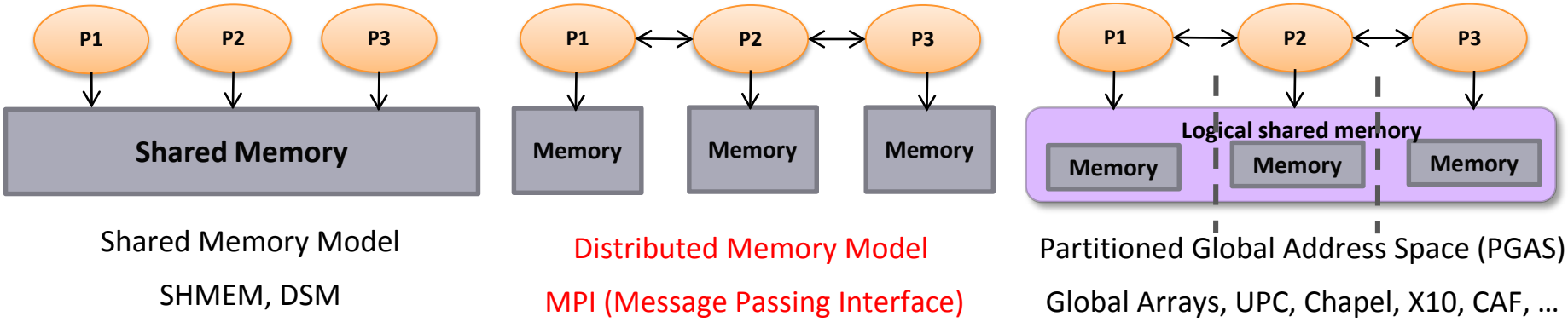
Tianhe – 2

Titan

Three Major Computing Categories

- Scientific Computing
  - Message Passing Interface (MPI), including MPI + OpenMP, is the Dominant Programming Model
  - Many discussions towards Partitioned Global Address Space (PGAS)
    - UPC, OpenSHMEM, CAF, UPC++ etc.
  - Hybrid Programming: MPI + PGAS (OpenSHMEM, UPC)
- Deep Learning
  - Caffe, CNTK, TensorFlow, and many more
- Big Data/Enterprise/Commercial Computing
  - Focuses on large data and data analysis
  - Spark and Hadoop (HDFS, HBase, MapReduce)
  - Memcached is also used for Web 2.0

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, …

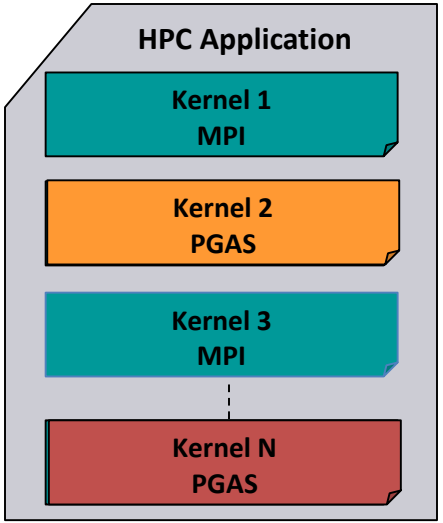
- Programming models provide abstract machine models
- Models can be mapped on different types of systems
  - e.g. Distributed Shared Memory (DSM), MPI within a node, etc.
- PGAS models and Hybrid MPI+PGAS models are gradually receiving importance

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
    - UPC++
    - Global Arrays

Hybrid (MPI+PGAS) Programming

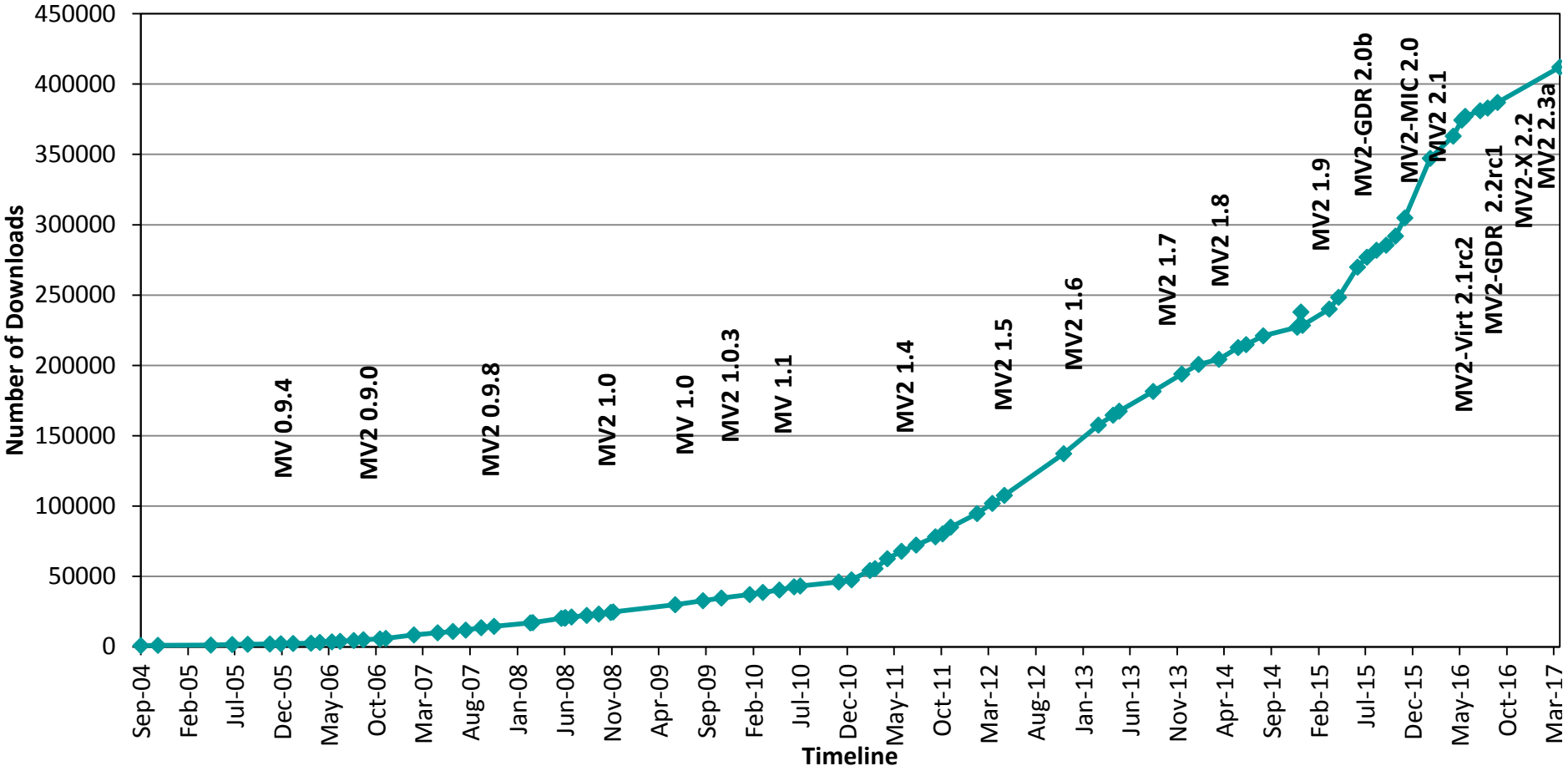
- Application sub-kernels can be re-written in MPI/PGAS based on communication characteristics
- Benefits:
  - Best of Distributed Computing Model
  - Best of Shared Memory Computing Model



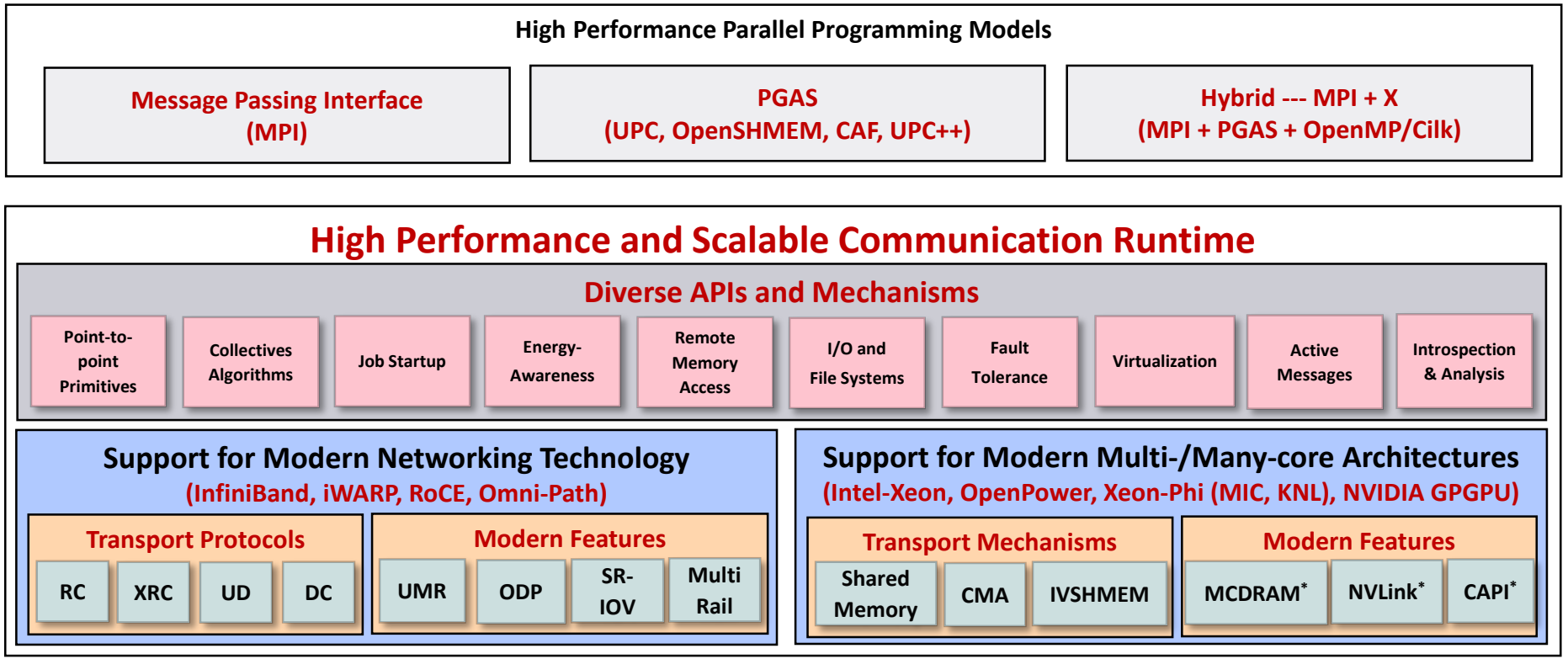
Overview of the MVAPICH2 Project

- 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 2,750 organizations in 83 countries**
  - **More than 412,000 (> 0.4 million) downloads from the OSU site directly**
  - Empowering many TOP500 clusters (Nov '16 ranking)
    - **1st, 10,649,600-core (Sunway TaihuLight) at National Supercomputing Center in Wuxi, China**
    - 13th, 241,108-core (Pleiades) at NASA
    - 17th, 462,462-core (Stampede) at TACC
    - 40th, 74,520-core (Tsubame 2.5) at Tokyo Institute of Technology
  - 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 (1st in Jun'16, 10M cores, 100 PFlops)

MVAPICH2 Release Timeline and Downloads



Architecture of MVAPICH2 Software Family



* Upcoming

MVAPICH2 Software Family

High-Performance Parallel Programming Libraries	
MVAPICH2	Support for InfiniBand, Omni-Path, Ethernet/iWARP, and RoCE
MVAPICH2-X	Advanced MPI features, OSU INAM, PGAS (OpenSHMEM, UPC, UPC++, and CAF), and MPI+PGAS programming models with unified communication runtime
MVAPICH2-GDR	Optimized MPI for clusters with NVIDIA GPUs
MVAPICH2-Virt	High-performance and scalable MPI for hypervisor and container based HPC cloud
MVAPICH2-EA	Energy aware and High-performance MPI
MVAPICH2-MIC	Optimized MPI for clusters with Intel KNC
Microbenchmarks	
OMB	Microbenchmarks suite to evaluate MPI and PGAS (OpenSHMEM, UPC, and UPC++) libraries for CPUs and GPUs
Tools	
OSU INAM	Network monitoring, profiling, and analysis for clusters with MPI and scheduler integration
OEMT	Utility to measure the energy consumption of MPI applications

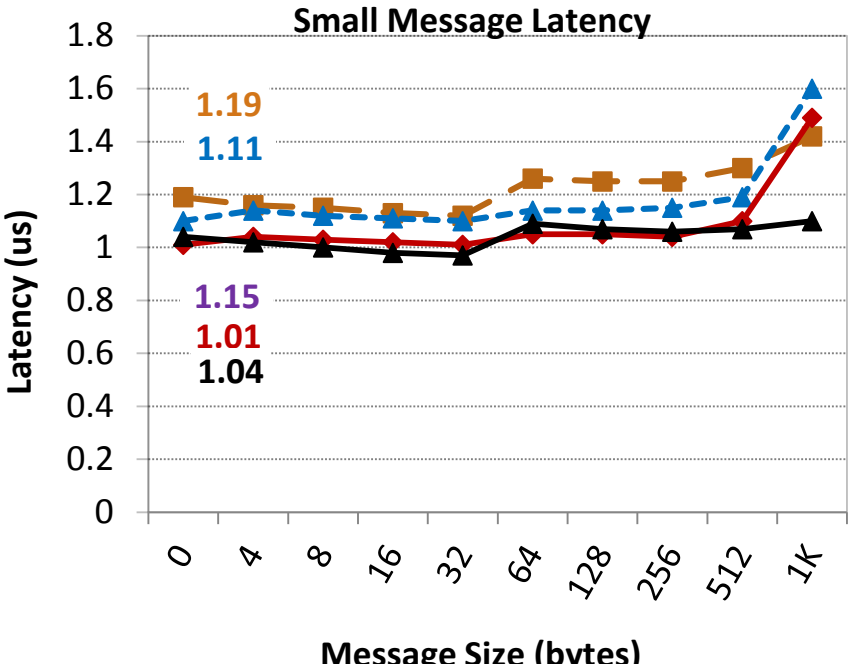
MVAPICH2 2.3a

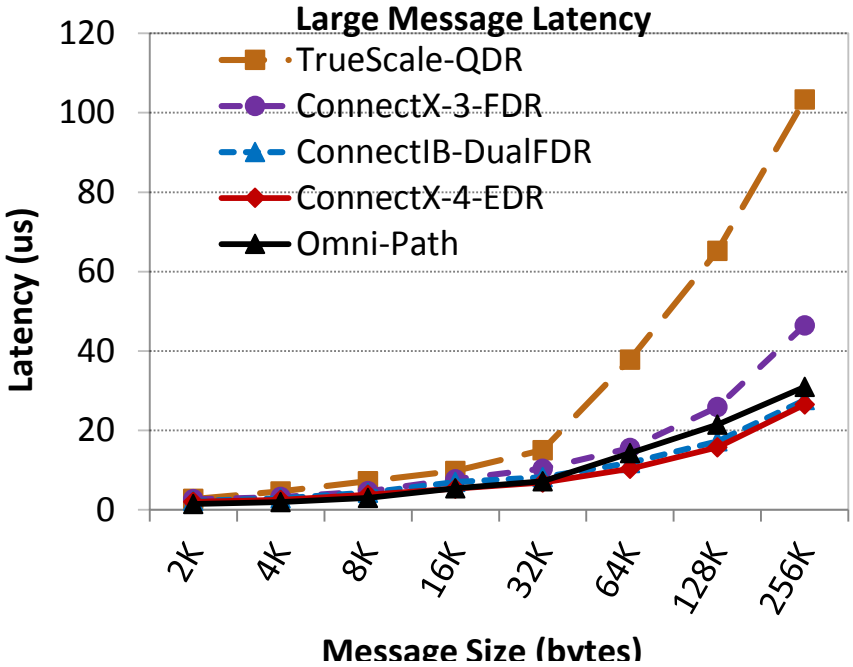
- Released on 03/29/2017
- Major Features and Enhancements
  - Based on and ABI compatible with MPICH 3.2
  - Support collective offload using Mellanox's SHArP for Allreduce
    - Enhance tuning framework for Allreduce using SHArP
  - Introduce capability to run MPI jobs across multiple InfiniBand subnets
  - Introduce basic support for executing MPI jobs in Singularity
  - Enhance collective tuning for Intel Knight's Landing and Intel Omni-path
  - Enhance process mapping support for multi-threaded MPI applications
    - Introduce MV2_CPU_BINDING_POLICY=hybrid
    - Introduce MV2_THREADS_PER_PROCESS
  - On-demand connection management for PSM-CH3 and PSM2-CH3 channels
  - Enhance PSM-CH3 and PSM2-CH3 job startup to use non-blocking PMI calls
  - Enhance debugging support for PSM-CH3 and PSM2-CH3 channels
  - Improve performance of architecture detection
  - Introduce run time parameter MV2_SHOW_HCA_BINDING to show process to HCA binding
    - Enhance MV2_SHOW_CPU_BINDING to enable display of CPU bindings on all nodes
  - Deprecate OFA-IB-Nemesis channel
  - Update to hwloc version 1.11.6

Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication
  - Support for advanced IB mechanisms (UMR and ODP)
  - Extremely minimal memory footprint (DCT)
  - Scalable Job Start-up
  - Dynamic and Adaptive Tag Matching
  - Collective Support with SHArP
- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, UPC++, …)
- Integrated Support for GPGPUs
- Energy-Awareness
- InfiniBand Network Analysis and Monitoring (INAM)
- Virtualization and HPC Cloud

One-way Latency: MPI over IB with MVAPICH2





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

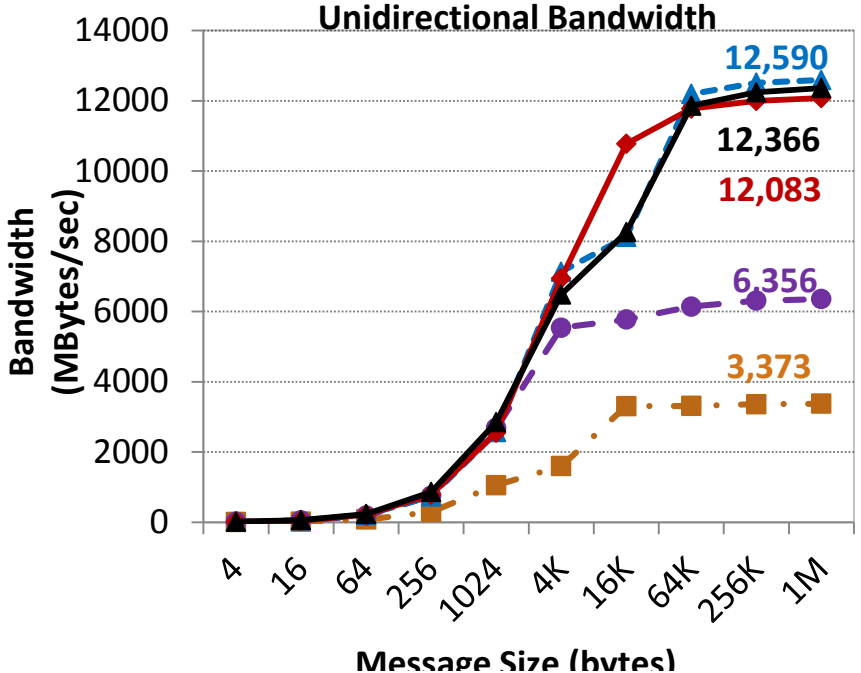
ConnectX-3-FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch

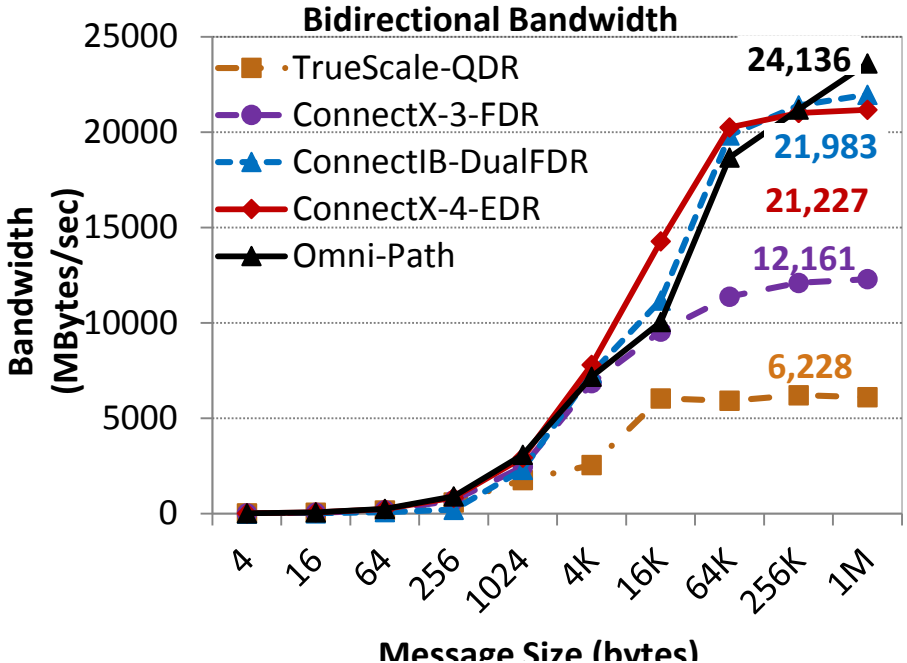
ConnectIB-Dual FDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB switch

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

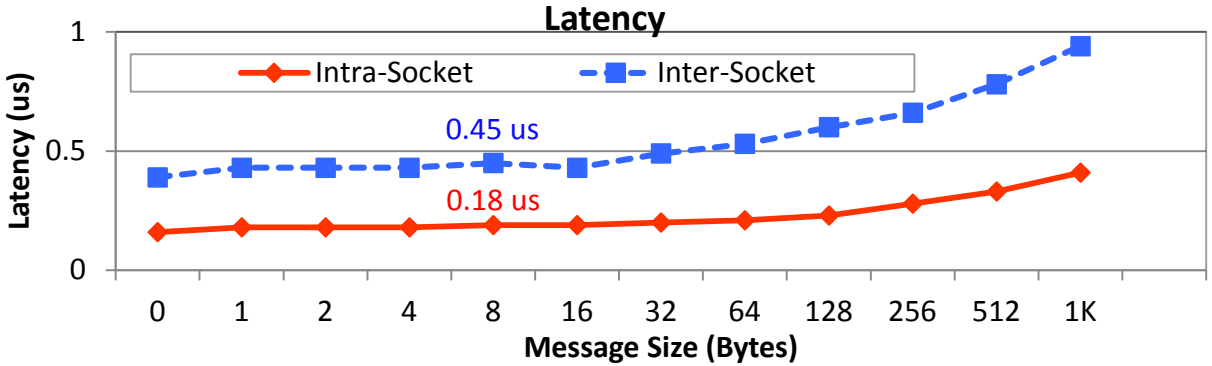
ConnectX-3-FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch

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

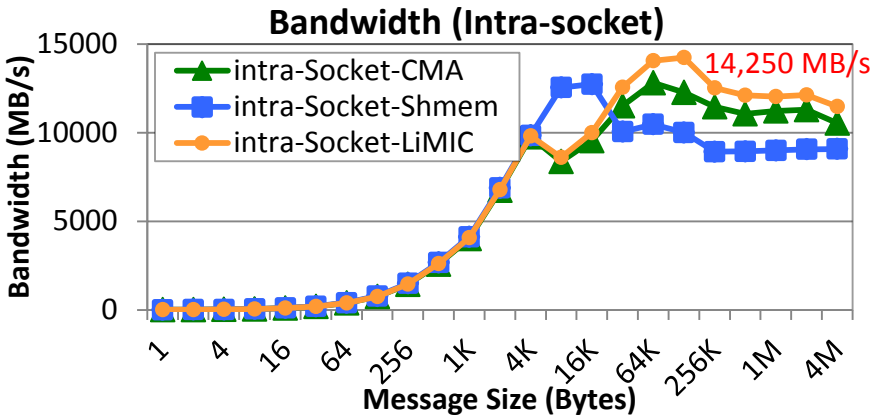
ConnectX-4-EDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 IB switch

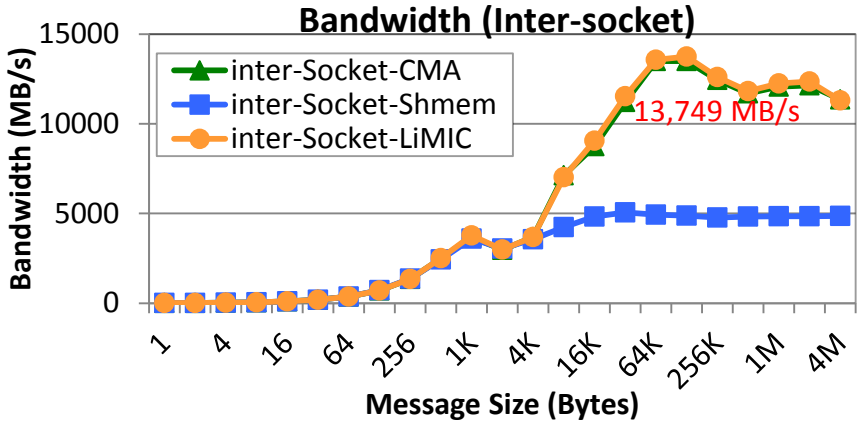
Omni-Path - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with Omni-Path switch

MVAPICH2 Two-Sided Intra-Node Performance
(Shared memory and Kernel-based Zero-copy Support (LiMIC and CMA))



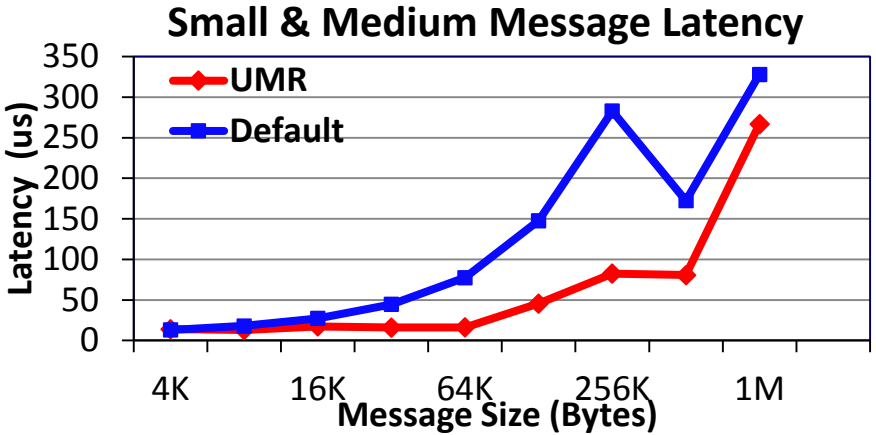
Intel Ivy-bridge

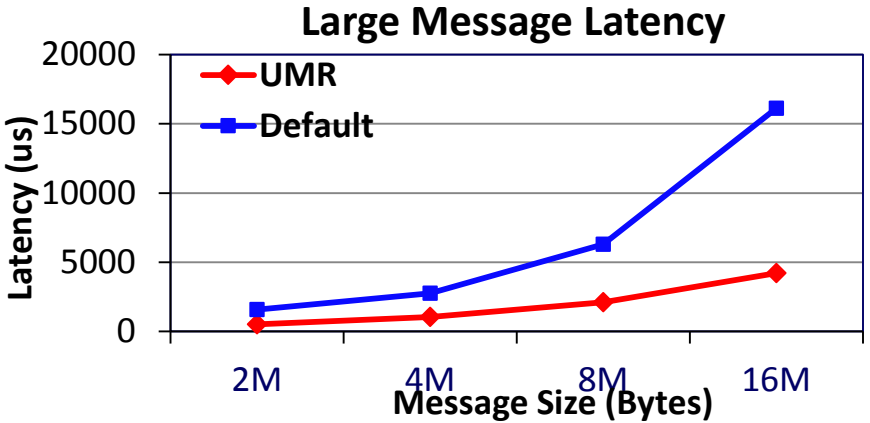




User-mode Memory Registration (UMR)

- Introduced by Mellanox to support direct local and remote noncontiguous memory access
  - Avoid packing at sender and unpacking at receiver
- Available since MVAPICH2-X 2.2b

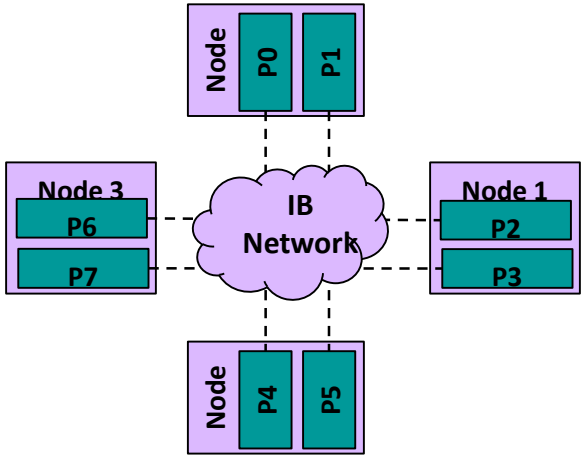




Connect-IB (54 Gbps): 2.8 GHz Dual Ten-core (IvyBridge) Intel PCI Gen3 with Mellanox IB FDR switch

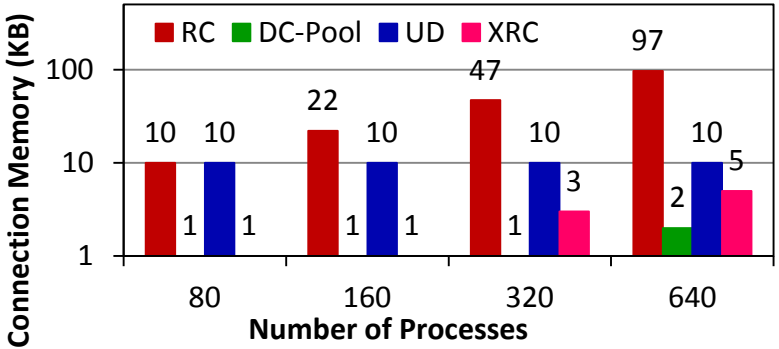
M. Li, H. Subramoni, K. Hamidouche, X. Lu and D. K. Panda, High Performance MPI Datatype Support with User-mode Memory Registration: Challenges, Designs and Benefits, CLUSTER, 2015

Minimizing Memory Footprint by Direct Connect (DC) Transport

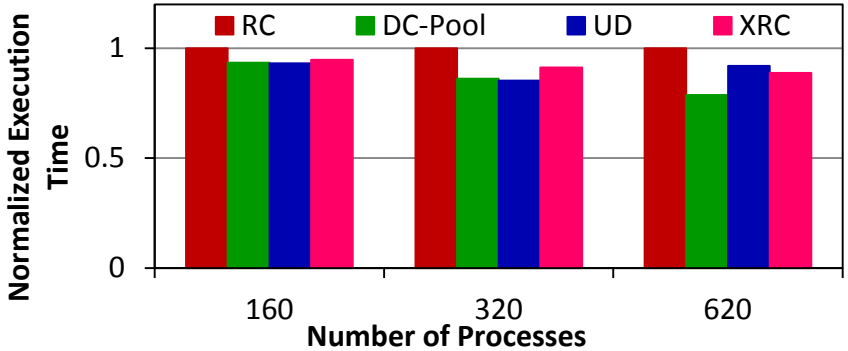


- Constant connection cost (*One QP for any peer*)
- Full Feature Set (RDMA, Atomics etc)
- Separate objects for send (DC Initiator) and receive (DC Target)
  - DC Target identified by "DCT Number"
  - Messages routed with (DCT Number, LID)
  - Requires same "DC Key" to enable communication
- Available since MVAPICH2-X 2.2a

Memory Footprint for Alltoall

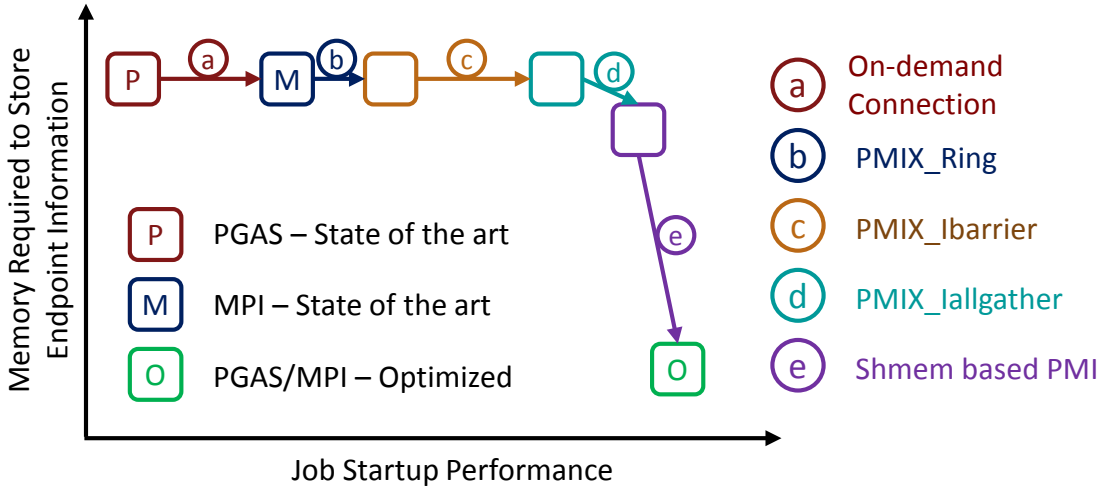


NAMD - Apoa1: Large data set



H. Subramoni, K. Hamidouche, A. Venkatesh, S. Chakraborty and D. K. Panda, Designing MPI Library with Dynamic Connected Transport (DCT) of InfiniBand : Early Experiences. IEEE International Supercomputing Conference (ISC '14)

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

(a) **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)

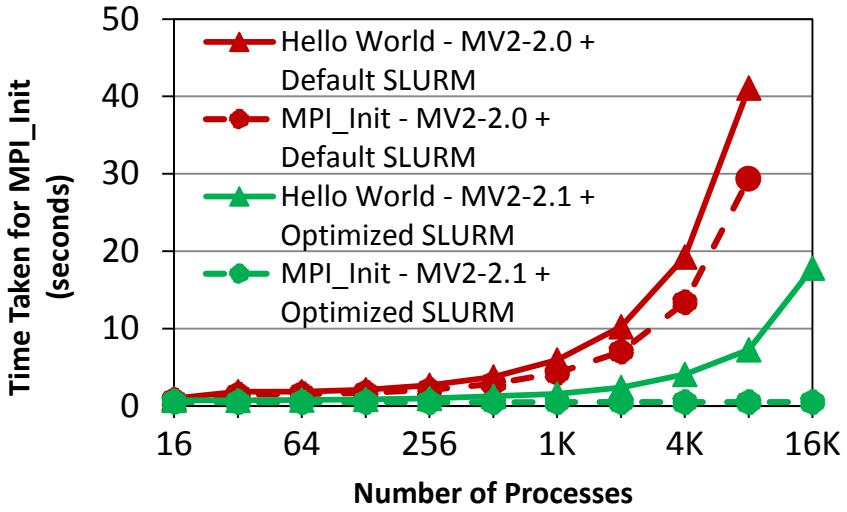
(b) **PMI Extensions for Scalable MPI Startup.** S. Chakraborty, H. Subramoni, A. Moody, J. Perkins, M. Arnold, and D K Panda, Proceedings of the 21st European MPI Users' Group Meeting (EuroMPI/Asia '14)

(c) (d) **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)

(e) **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)

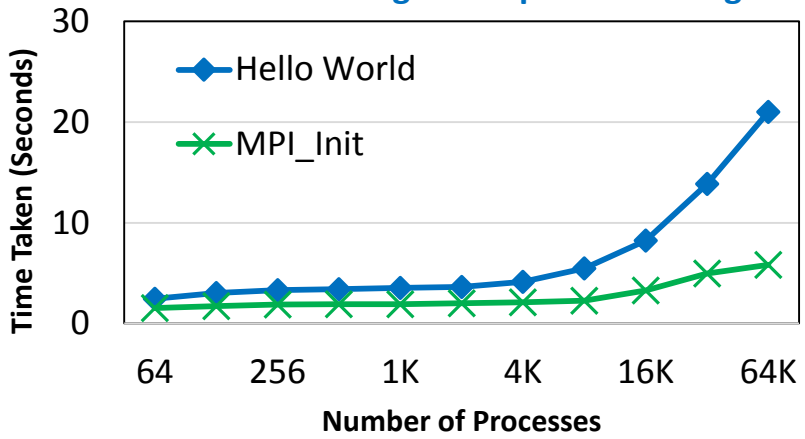
Scalable Job Startup with Non-blocking PMI

Co-designing Resource Manager and MPI library



- Near-constant MPI_Init at any scale
- MPI_Init is 59 times faster
- Hello World (MPI_Init + MPI_Finalize) takes 5.7 times less time with 8,192 processes (512 nodes)
- 16,384 processes, 1,024 Nodes (Sandy Bridge + FDR)

Resource Manager Independent Design

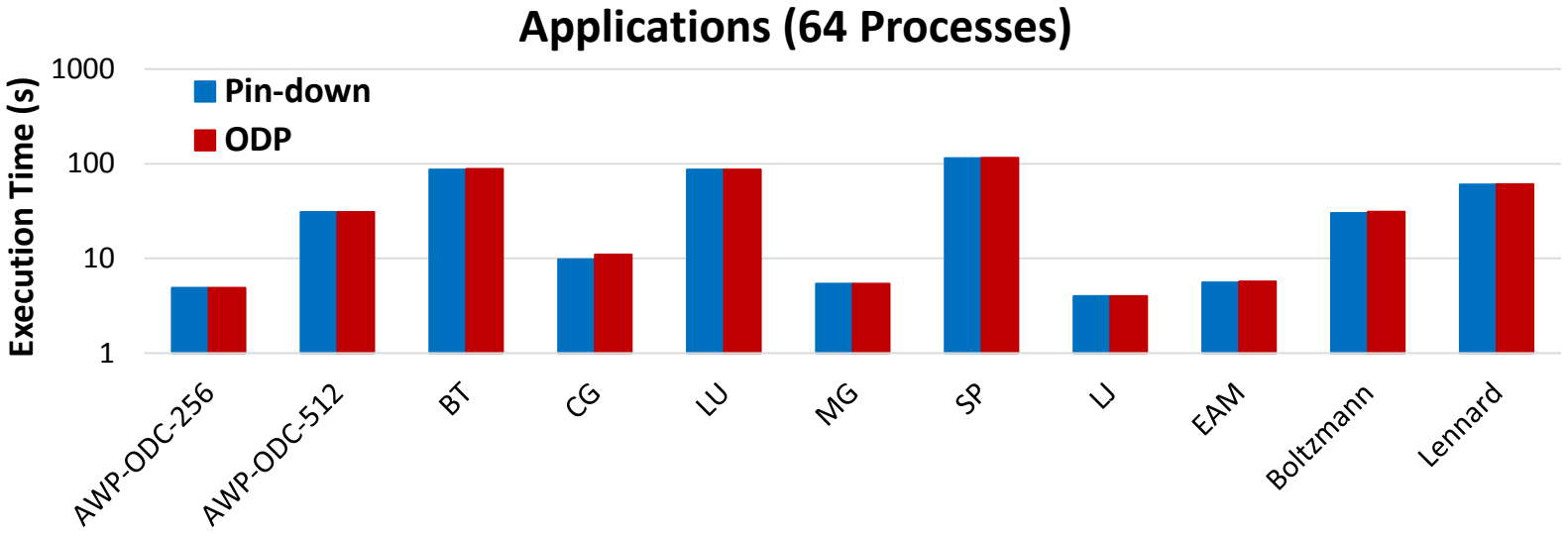


- Non-blocking PMI implementation in mpirun_rsh
- On-demand connection management for PSM and Omni-path
- Efficient intra-node startup for Knights Landing
- MPI_Init takes 5.8 seconds
- Hello World takes 21 seconds
- 65,536 processes, 1,024 Nodes (KNL + Omni-Path)

New designs available in MVAPICH2-2.3a and as patch for SLURM-15.08.8 and SLURM-16.05.1

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
- Available in MVAPICH2-X 2.2



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

Dynamic and Adaptive Tag Matching

Challenge

Tag matching is a significant overhead for receivers

Existing Solutions are

- Static and do not adapt dynamically to communication pattern
- Do not consider memory overhead

Solution

A new tag matching design

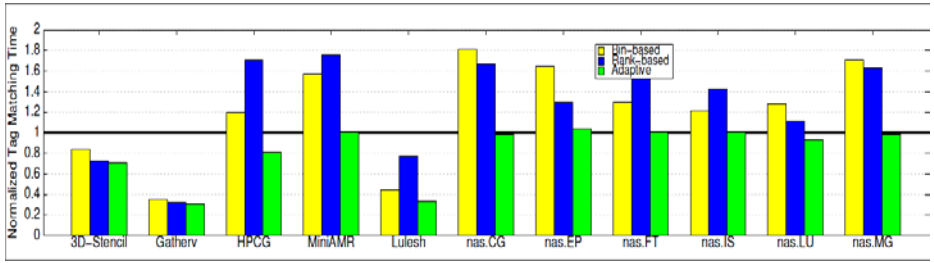
- Dynamically adapt to communication patterns
- Use different strategies for different ranks
- Decisions are based on the number of request object that must be traversed before hitting on the required one

Results

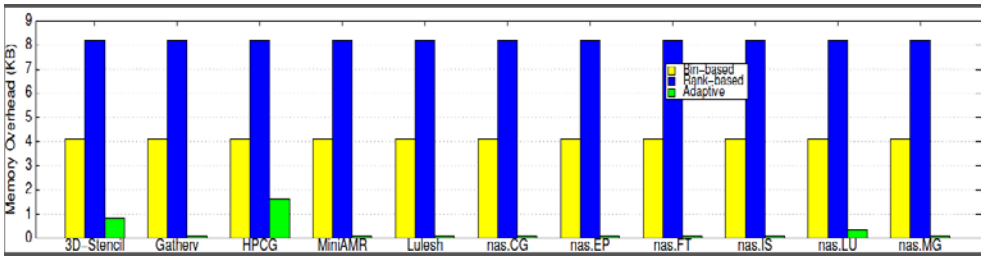
Better performance than other state-of-the art tag-matching schemes

Minimum memory consumption

Will be available in future MVAPICH2 releases



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

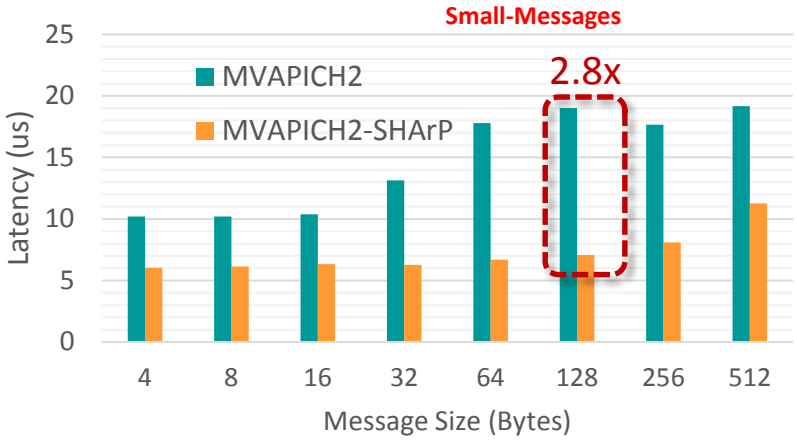


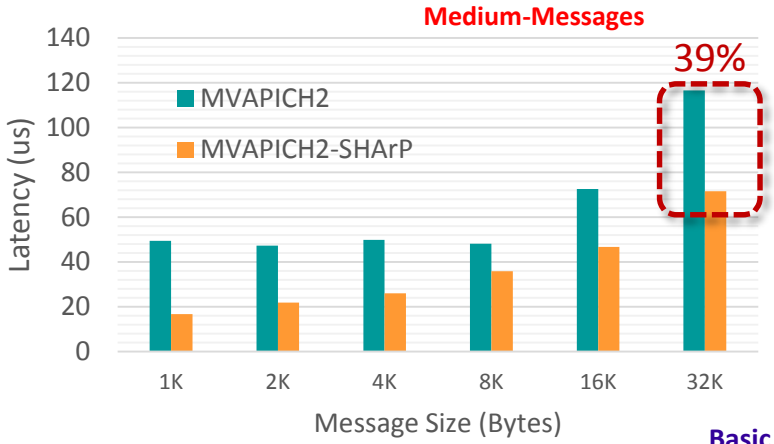
Normalized Memory Overhead per Process at 512 Processes
Compared to Default (Lower is Better)

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

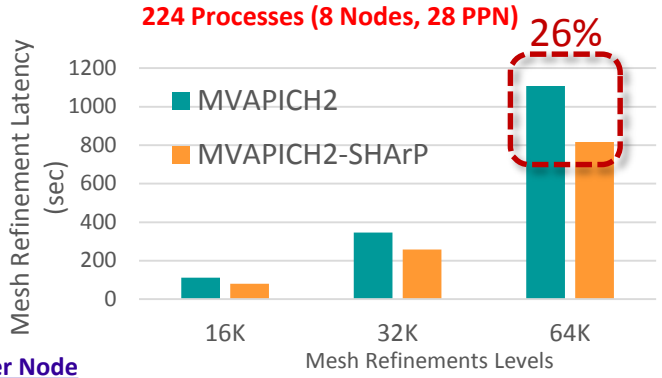
Advanced Allreduce Collective Designs Using Switch Offload Mechanism (SHArP)

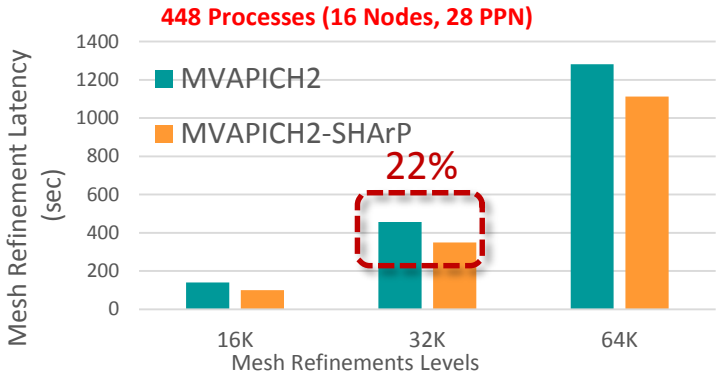
osu_allreduce (OSU Micro Benchmark) with 448 processes (28 *PPN 16 Nodes)





Avg of Mesh Refinement Latency for miniAMR Application (28 PPN)





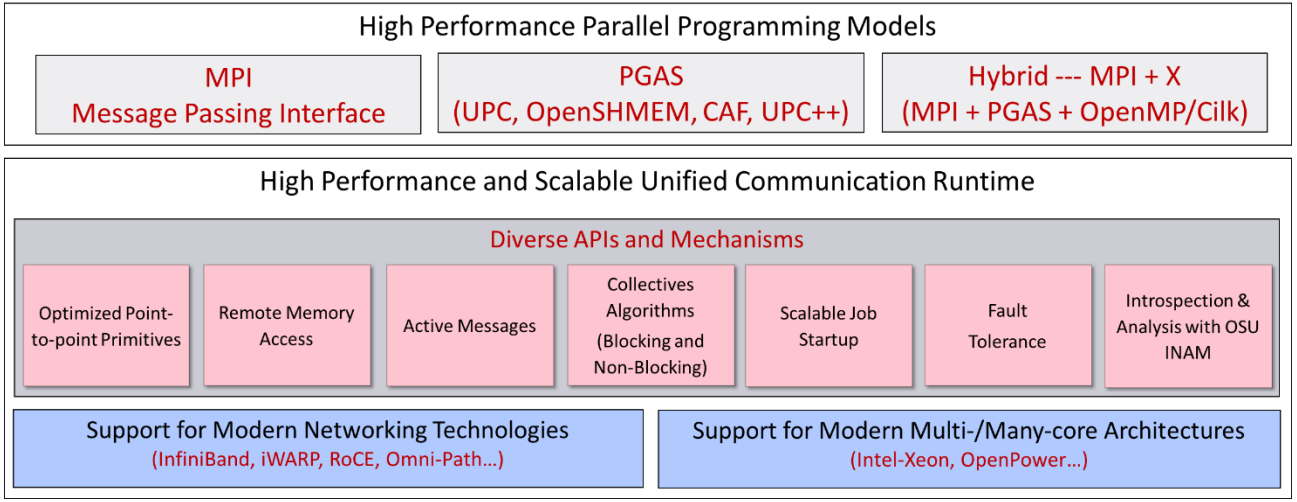
Basic Support Available in MVAPICH2 2.3a

*PPN: Processes Per Node

Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

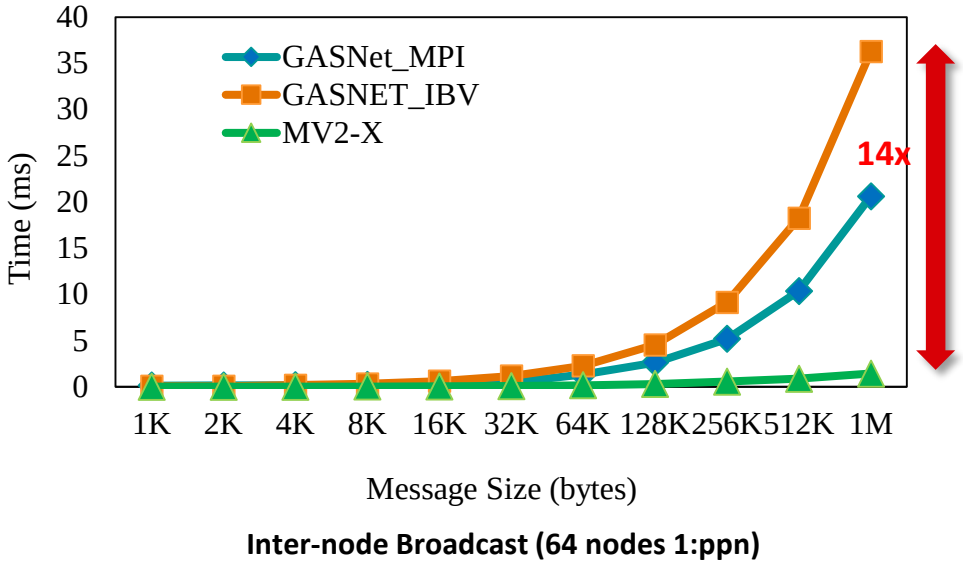
- Scalability for million to billion processors
- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, UPC++, …)
- Integrated Support for GPGPUs
- Energy-Awareness
- InfiniBand Network Analysis and Monitoring (INAM)
- Virtualization and HPC Cloud

MVAPICH2-X for Hybrid MPI + PGAS Applications

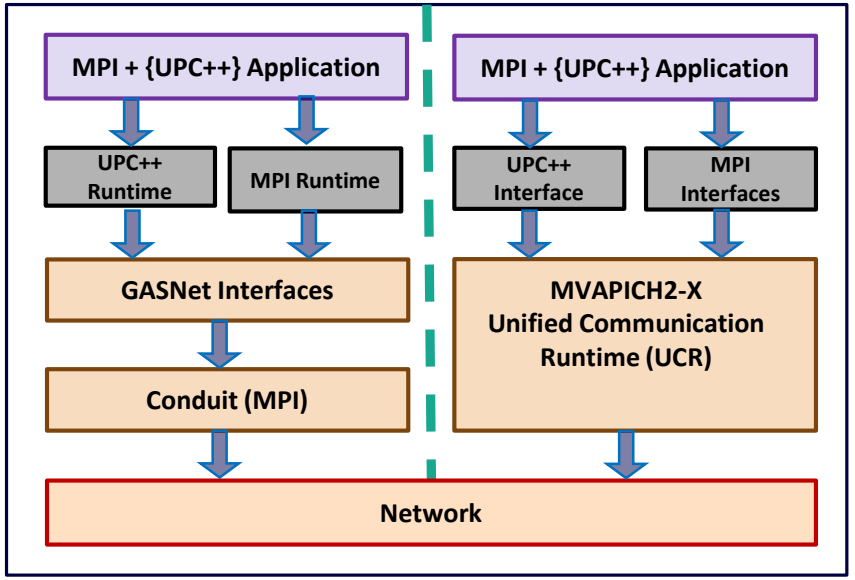


- Current Model – Separate Runtimes for OpenSHMEM/UPC/UPC++/CAF and MPI
  - Possible deadlock if both runtimes are not progressed
  - Consumes more network resource
- Unified communication runtime for MPI, UPC, UPC++, OpenSHMEM, CAF
  - Available with since 2012 (starting with MVAPICH2-X 1.9)
  - http://mvapich.cse.ohio-state.edu

UPC++ Support in MVAPICH2-X

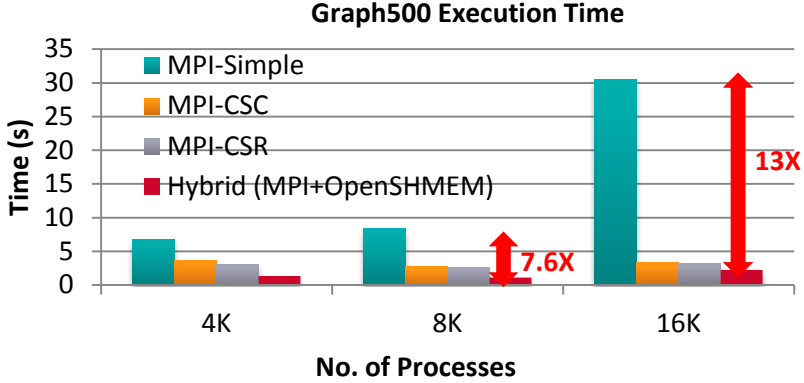


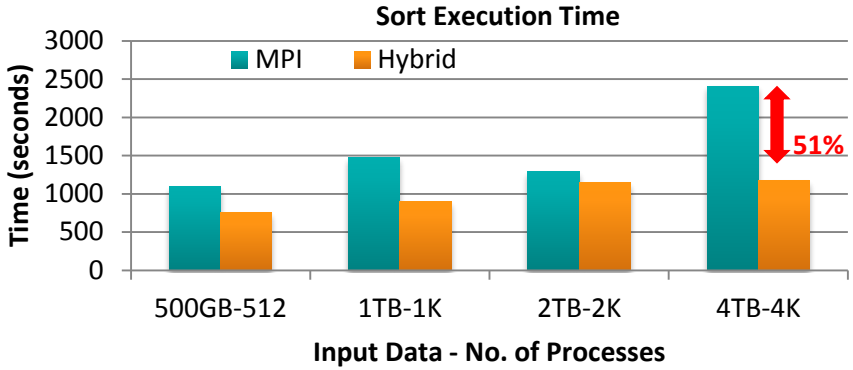
Inter-node Broadcast (64 nodes 1:ppn)



- 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)

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

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

J. Jose, K. Kandalla, M. Luo and D. K. Panda, Supporting Hybrid MPI and OpenSHMEM over InfiniBand: Design and Performance Evaluation, Int'l Conference on Parallel Processing (ICPP '12), September 2012

Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

- Scalability for million to billion processors
- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, UPC++, …)
- Integrated Support for GPGPUs
  - CUDA-aware MPI
  - GPUDirect RDMA (GDR) Support
- Energy-Awareness
- InfiniBand Network Analysis and Monitoring (INAM)
- Virtualization and HPC Cloud

MPI + CUDA - Naive

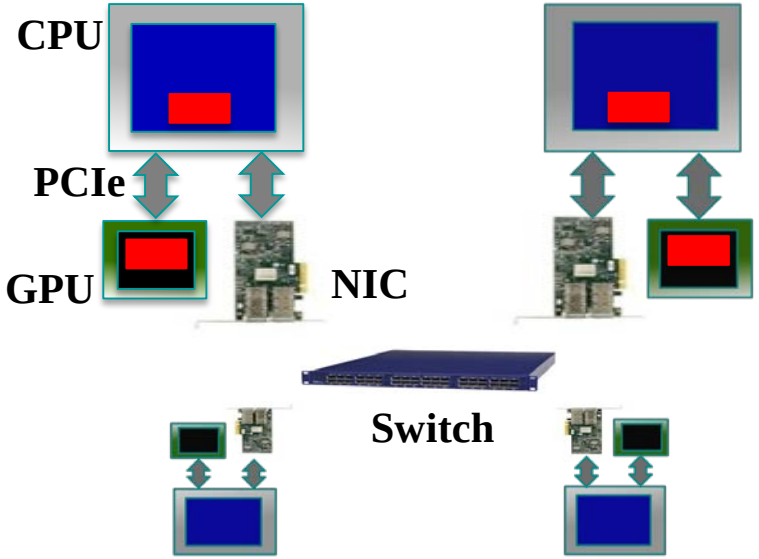
- Data movement in applications with standard MPI and CUDA interfaces

At Sender:

```
cudaMemcpy(s_hostbuf, s_devbuf, . . .);
MPI_Send(s_hostbuf, size, . . .);
```

At Receiver:

```
MPI_Recv(r_hostbuf, size, . . .);
cudaMemcpy(r_devbuf, r_hostbuf, . . .);
```



High Productivity and Low Performance

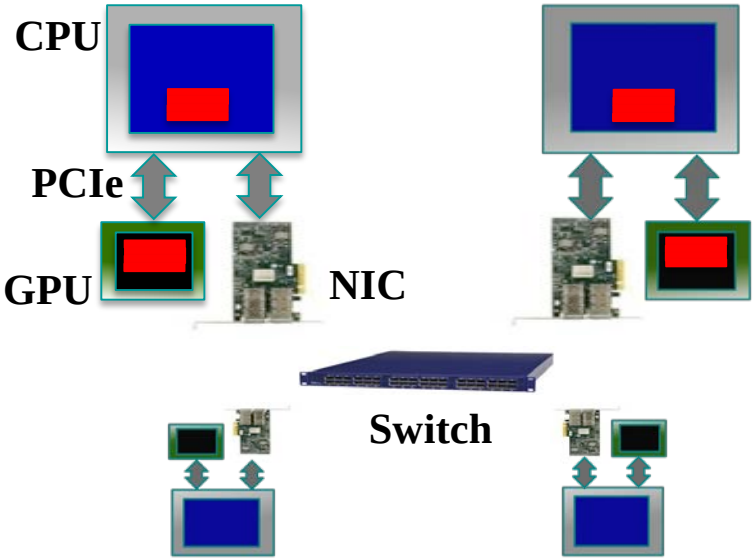
MPI + CUDA - Advanced

- Pipelining at user level with non-blocking MPI and CUDA interfaces

At Sender:

```
for (j = 0; j < pipeline_len; j++)
    cudaMemcpyAsync(s_hostbuf + j * blk, s_devbuf + j * blksz, …);
for (j = 0; j < pipeline_len; j++) {
       while (result != cudaSucess) {
            result = cudaStreamQuery(…);
             if(j > 0) MPI_Test(…);
       }
       MPI_Isend(s_hostbuf + j * block_sz, blksz . . .);
 }
MPI_Waitall();
```

<<Similar at receiver>>



Low Productivity and High Performance

GPU-Aware (CUDA-Aware) MPI Library: MVAPICH2-GPU

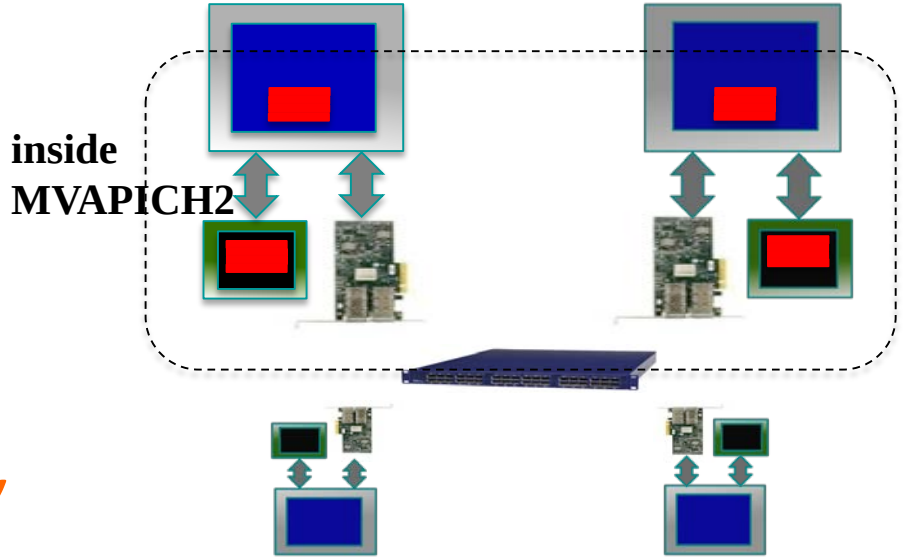
- Standard MPI interfaces used for unified data movement
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

```
MPI_Send(s_devbuf, size, …);
```

At Receiver:

```
MPI_Recv(r_devbuf, size, …);
```

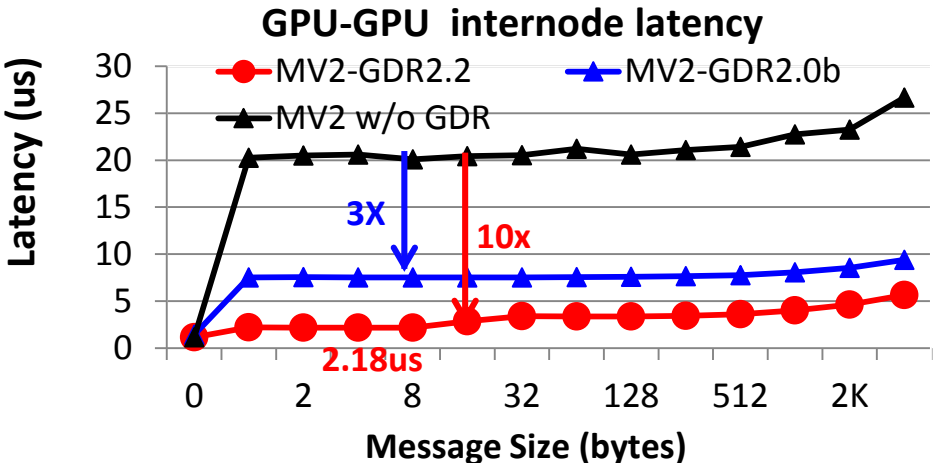


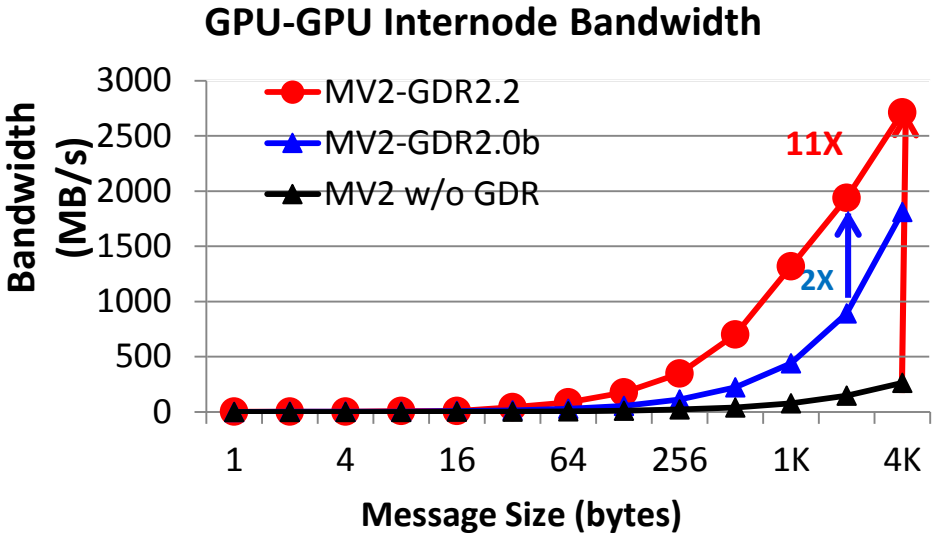
High Performance and High Productivity

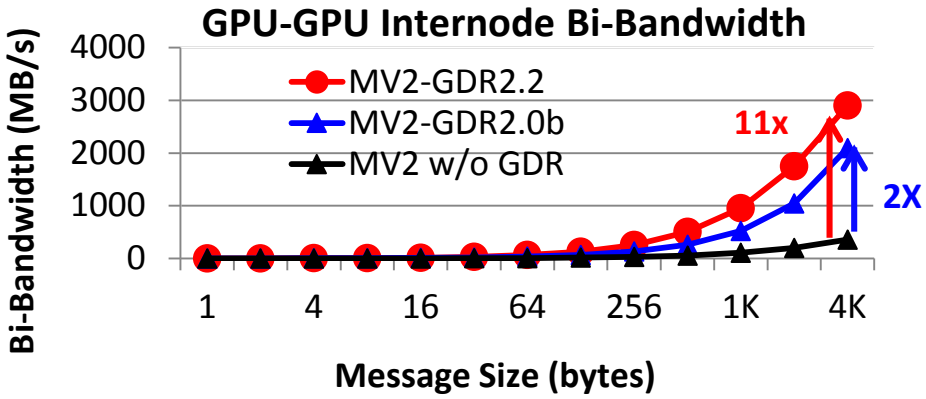
CUDA-Aware MPI: MVAPICH2-GDR 1.8-2.2 Releases

- 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
- Unified memory

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

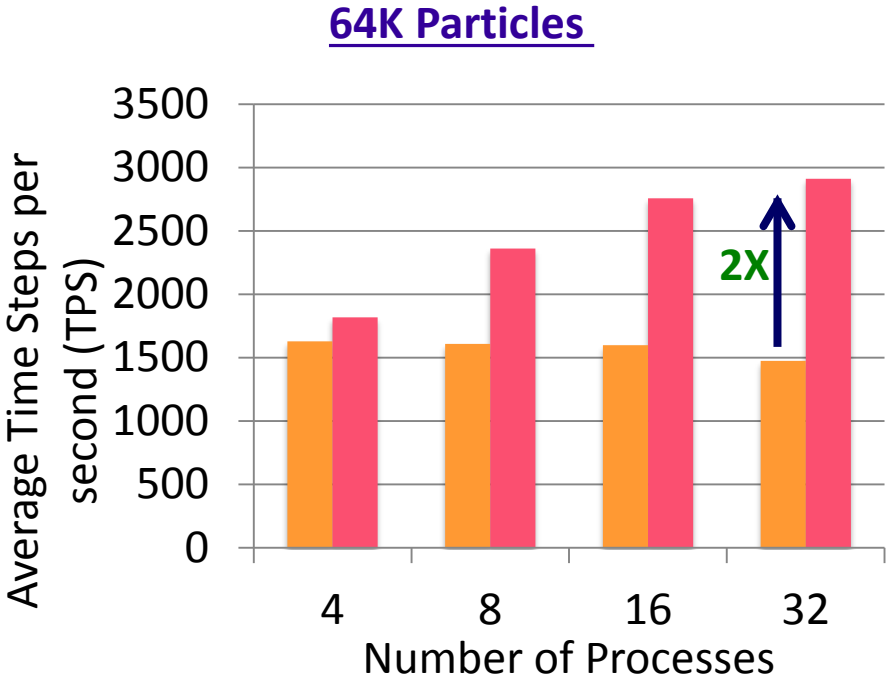


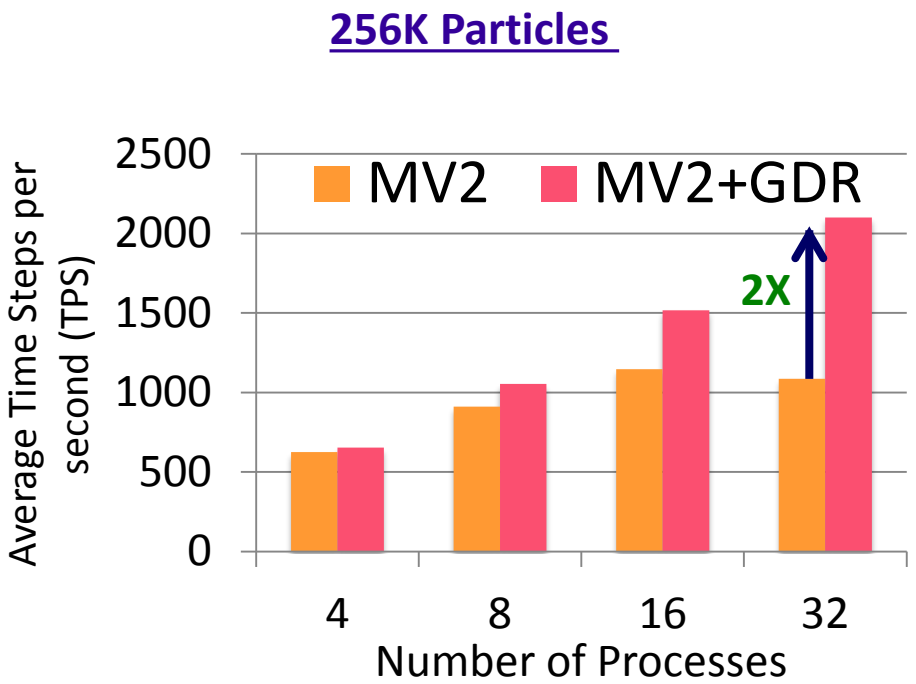




MVAPICH2-GDR-2.2
Intel Ivy Bridge (E5-2680 v2) node - 20 cores
NVIDIA Tesla K40c GPU
Mellanox Connect-X4 EDR HCA
CUDA 8.0
Mellanox OFED 3.0 with GPU-Direct-RDMA

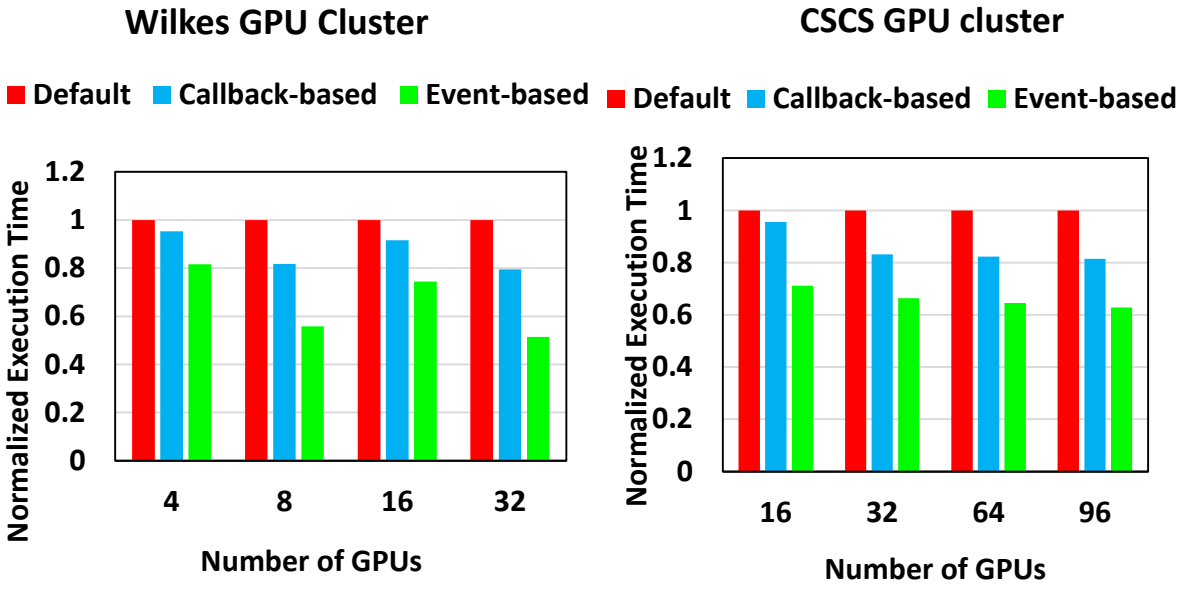
Application-Level Evaluation (HOOMD-blue)

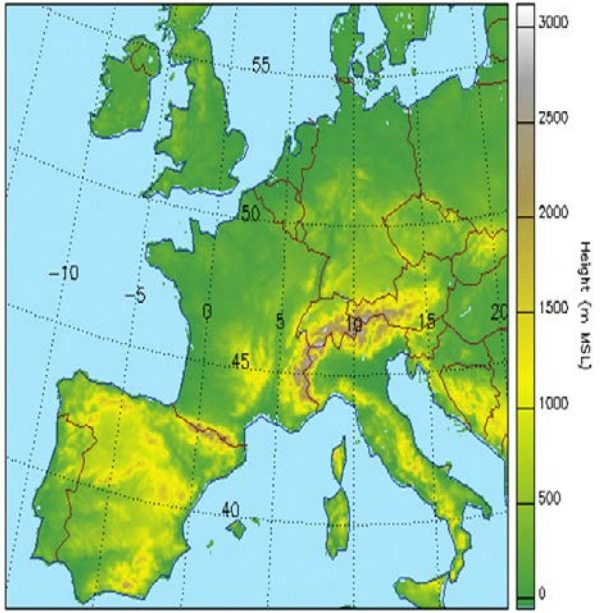




- Platform: Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- HoomdBlue Version 1.0.5
  - GDRCOPY enabled: MV2_USE_CUDA=1 MV2_IBA_HCA=mlx5_0 MV2_IBA_EAGER_THRESHOLD=32768 MV2_VBUF_TOTAL_SIZE=32768 MV2_USE_GPUDIRECT_LOOPBACK_LIMIT=32768 MV2_USE_GPUDIRECT_GDRCOPY=1 MV2_USE_GPUDIRECT_GDRCOPY_LIMIT=16384

Application-Level Evaluation (Cosmo) and Weather Forecasting in Switzerland





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

Cosmo model: http://www2.cosmo-model.org/content/tasks/operational/meteoSwiss/

On-going collaboration with CSCS and MeteoSwiss (Switzerland) 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

Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

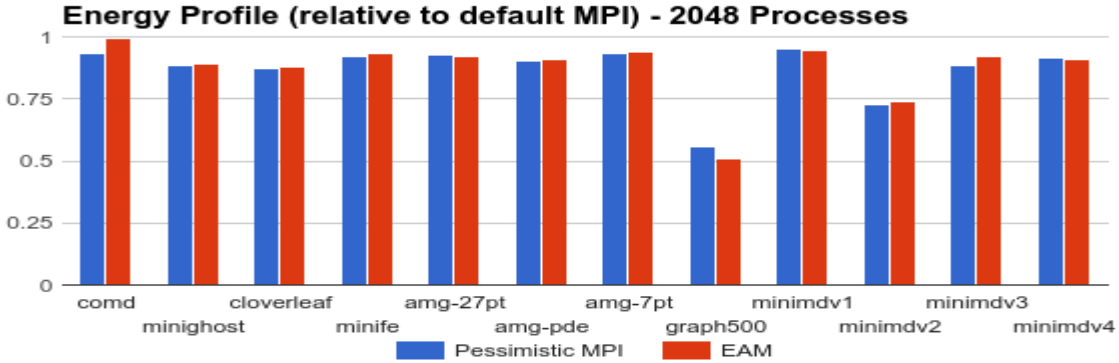
- Scalability for million to billion processors
- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, UPC++, …)
- Integrated Support for GPGPUs
- Energy-Awareness
- InfiniBand Network Analysis and Monitoring (INAM)

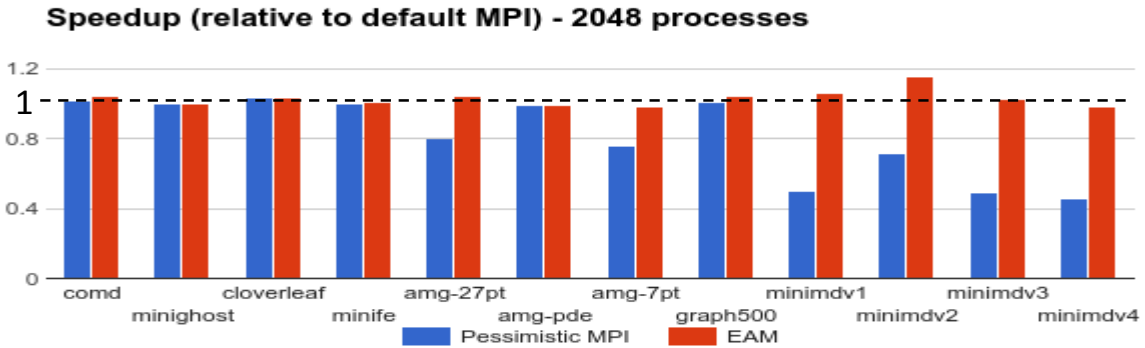
Energy-Aware MVAPICH2 & OSU Energy Management Tool (OEMT)

- 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
- Publicly available since August ‘15

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]*

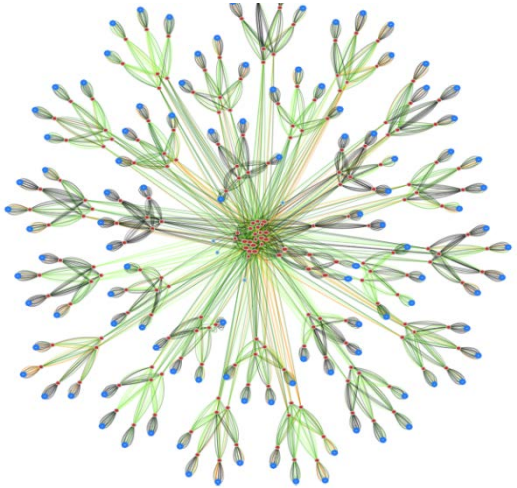
Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

- Scalability for million to billion processors
- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, UPC++, …)
- Integrated Support for GPGPUs
- Energy-Awareness
- InfiniBand Network Analysis and Monitoring (INAM)

Overview of OSU INAM

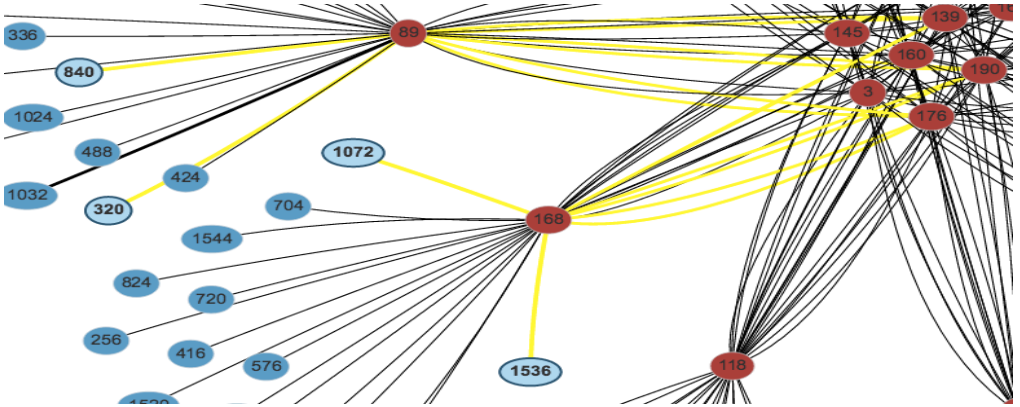
- A network monitoring and analysis tool that is capable of analyzing traffic on the InfiniBand network with inputs from the MPI runtime
  - http://mvapich.cse.ohio-state.edu/tools/osu-inam/
- Monitors IB clusters in real time by querying various subnet management entities and gathering input from the 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

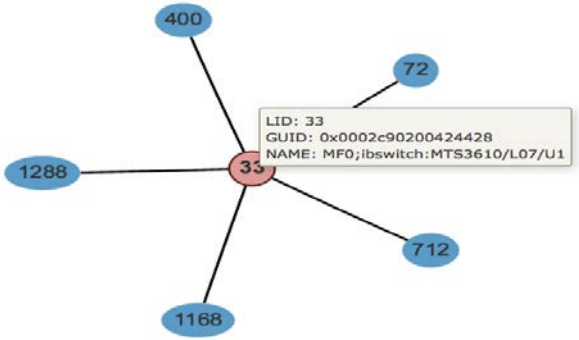
(1,879 nodes, 212 switches, 4,377 network links)



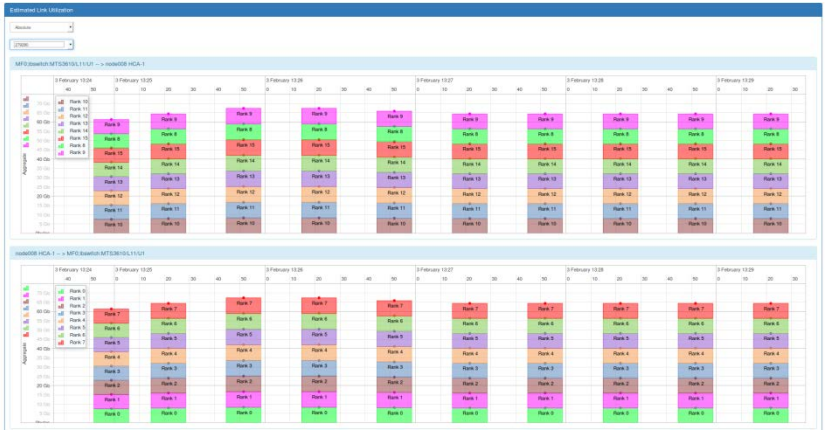
Finding Routes Between Nodes

- 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

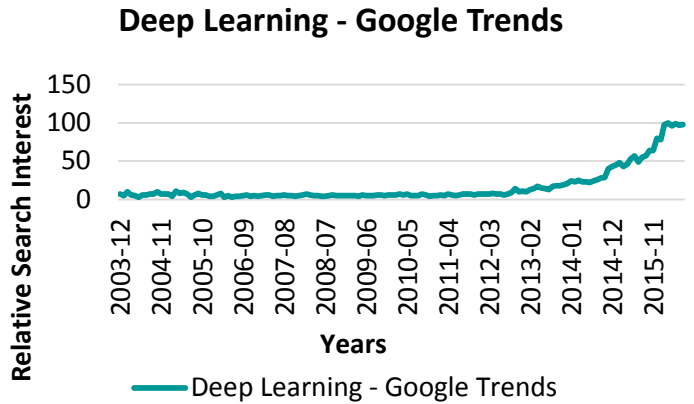
- Job level view
  - Show different network metrics (load, error, etc.) for any live job
  - Play back historical data for completed jobs to identify bottlenecks
- Node level view - details per process or per node
  - CPU utilization for each rank/node
  - Bytes sent/received for MPI operations (pt-to-pt,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

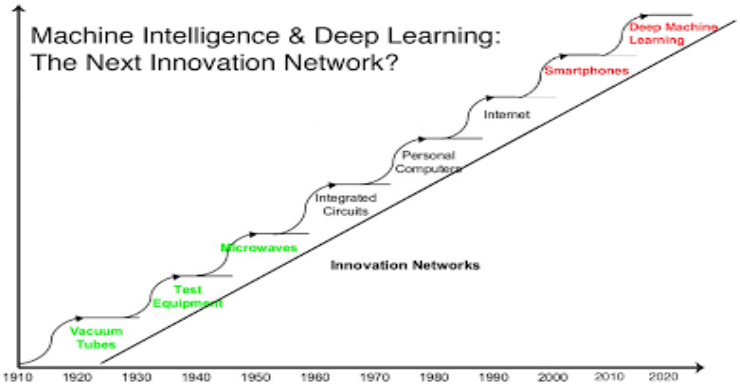
Major Computing Categories

- Scientific Computing
  - Message Passing Interface (MPI), including MPI + OpenMP, is the Dominant Programming Model
  - Many discussions towards Partitioned Global Address Space (PGAS)
    - UPC, OpenSHMEM, CAF, UPC++ etc.
  - Hybrid Programming: MPI + PGAS (OpenSHMEM, UPC)
- Deep Learning
  - Caffe, CNTK, TensorFlow, and many more

Deep Learning and MPI: State-of-the-art

- Deep Learning is going through a resurgence
  - Excellent accuracy for deep/convolutional neural networks
  - Public availability of versatile datasets like MNIST, CIFAR, and ImageNet
  - Widespread popularity of accelerators like Nvidia GPUs
- DL frameworks and applications
  - Caffe, Microsoft CNTK, Google TensorFlow and many more..
  - Most of the frameworks are exploiting GPUs to accelerate training
  - Diverse range of applications – Image Recognition, Cancer Detection, Self-Driving Cars, Speech Processing etc.
- Can MPI runtimes like MVAPICH2 provide efficient support for Deep Learning workloads?
  - MPI runtimes typically deal with
    - relatively small-sizes message (order of kilobytes)
    - CPU-based communication buffers





http://www.computervisionblog.com/2015/11/the-deep-learning-gold-rush-of-2015.html

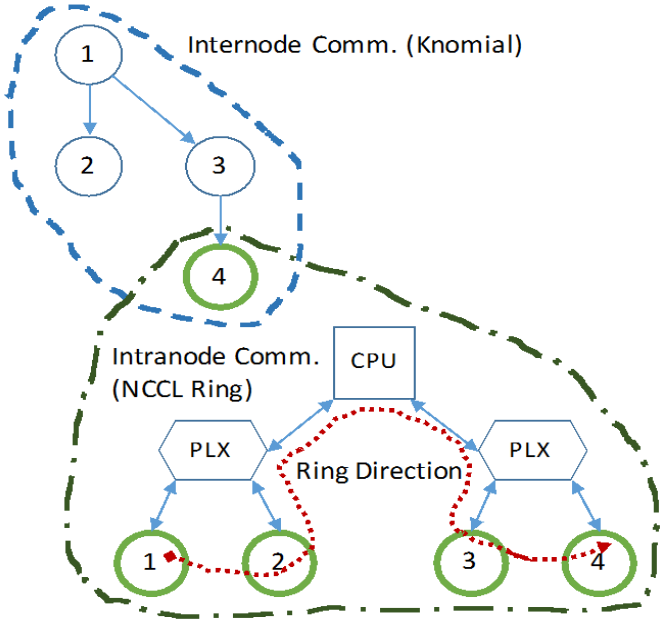
https://www.tensorflow.org

https://github.com/BVLC/caffe

https://cntk.ai

Deep Learning: New Challenges for MPI Runtimes

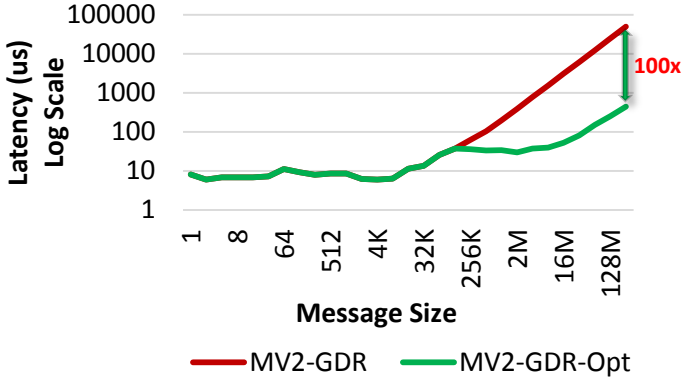
- Deep Learning frameworks are a different game altogether
  - Unusually large message sizes (order of megabytes)
  - Most communication based on GPU buffers
- How to address these newer requirements?
  - GPU-specific Communication Libraries (NCCL)
    - NVidia's NCCL library provides inter-GPU communication
  - CUDA-Aware MPI (MVAPICH2-GDR)
    - Provides support for GPU-based communication
- Can we exploit CUDA-Aware MPI and NCCL to support Deep Learning applications?



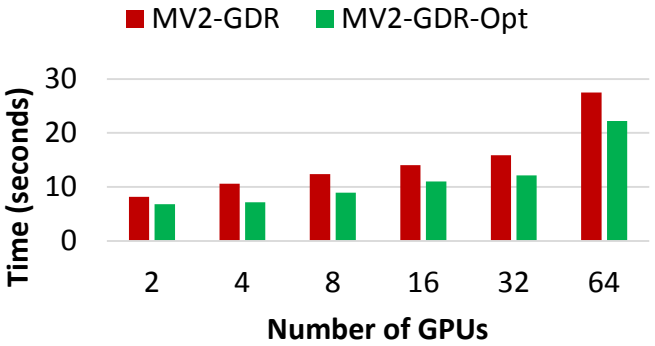
Hierarchical Communication (Knomial + NCCL ring)

Efficient Broadcast: MVAPICH2-GDR and NCCL

- 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)
- We propose optimized MPI_Bcast
  - Communication of very large GPU buffers (order of megabytes)
  - Scale-out on large number of dense multi-GPU nodes
- Hierarchical Communication that efficiently exploits:
  - CUDA-Aware MPI_Bcast in MV2-GDR
  - NCCL Broadcast primitive



Performance Benefits: OSU Micro-benchmarks

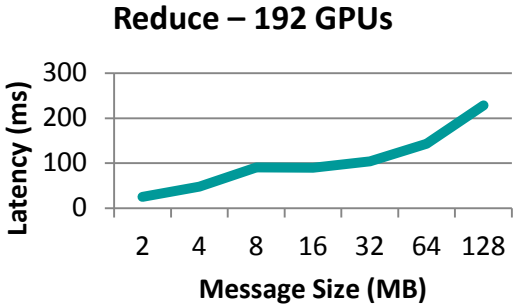


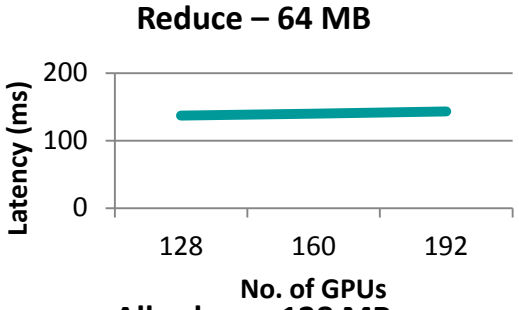
Performance Benefits: Microsoft CNTK DL framework
(25% avg. improvement)

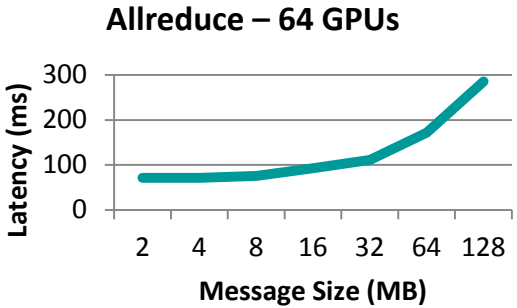
Efficient Large Message Broadcast using NCCL and CUDA-Aware MPI for Deep Learning,
A. Awan , K. Hamidouche , A. Venkatesh , and D. K. Panda,
The 23rd European MPI Users' Group Meeting (EuroMPI 16), Sep 2016 [Best Paper Runner-Up]

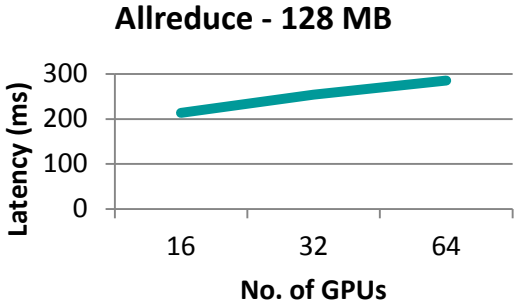
Large Message Optimized Collectives for Deep Learning

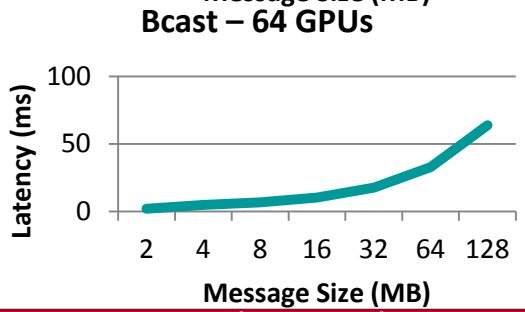
- MV2-GDR provides optimized collectives for **large message sizes**
- Optimized Reduce, Allreduce, and Bcast
- **Good scaling with large number of GPUs**
- **Available with MVAPICH2-GDR 2.2GA**

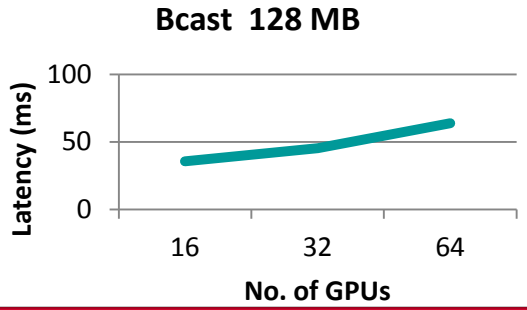










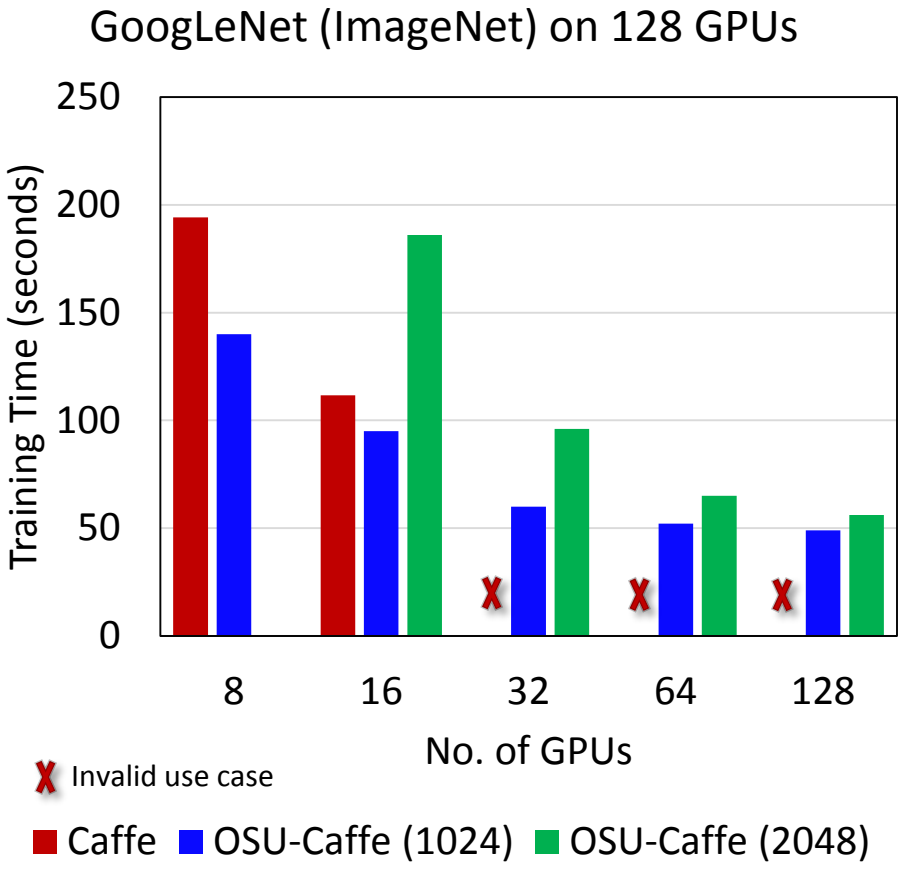


OSU-Caffe: Scalable Deep Learning

- 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
  - Limited Scale-out
- OSU-Caffe: MPI-based Parallel Training
  - Enable Scale-up (within a node) and Scale-out (across multi-GPU nodes)
  - network on ImageNet dataset

Awan, K. Hamidouche, J. Hashmi, and D. K. Panda, S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters, PPoPP, Sep 2017

OSU-Caffe is publicly available from:
http://hidl.cse.ohio-state.edu



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
- 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 3.1)
- Extended Checkpoint-Restart and migration support with SCR
- Support for * features will be available in future MVAPICH2 Releases

Two More Presentations

- Thursday (03/30/17) at 9:00am

  Building Efficient HPC Clouds with MVAPICH2 and RDMA-Hadoop over SR-IOV IB Clusters

- Friday (03/31/17) at 11:00am

  NVM-aware RDMA-Based Communication and I/O Schemes for High-Perf Big Data Analytics

Funding Acknowledgments

Personnel Acknowledgments

Current Students

- A. Awan (Ph.D.)
- R. Biswas (M.S.)
- M. Bayatpour (Ph.D.)
- S. Chakraborthy (Ph.D.)
- C.-H. Chu (Ph.D.)
- S. Guganani (Ph.D.)
- J. Hashmi (Ph.D.)
- H. Javed (Ph.D.)
- M. Li (Ph.D.)
- D. Shankar (Ph.D.)
- H. Shi (Ph.D.)
- J. Zhang (Ph.D.)

Current Research Scientists

- X. Lu
- H. Subramoni

Current Research Specialist

- J. Smith

Past Students

- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- S. Bhagvat (M.S.)
- A. Bhat (M.S.)
- D. Buntinas (Ph.D.)
- 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.)
- K. Kulkarni (M.S.)
- 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 Research Scientist

- K. Hamidouche
- S. Sur

Past Programmers

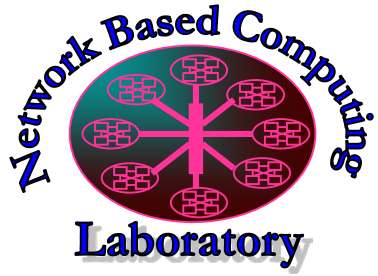
- D. Bureddy
- M. Arnold
- J. Perkins

Past Post-Docs

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

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/

HiDL
High-Performance Deep Learning

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/