Argonne NATIONAL LABORATORY

MPICH BoF

Supercomputing '14

New Orleans, LA, USA

November 18, 2014

U.S. DEPARTMENT OF ENERGY

The MPICH Project

- MPICH and its derivatives are the world's most widely used MPI implementations
  - Supports all versions of the MPI standard including the recent MPI-3
- Funded by DOE for 22 years (turned 22 this month)
- Has been a key influencer in the adoption of MPI
- Award winning project
  - DOE R&D100 award in 2005

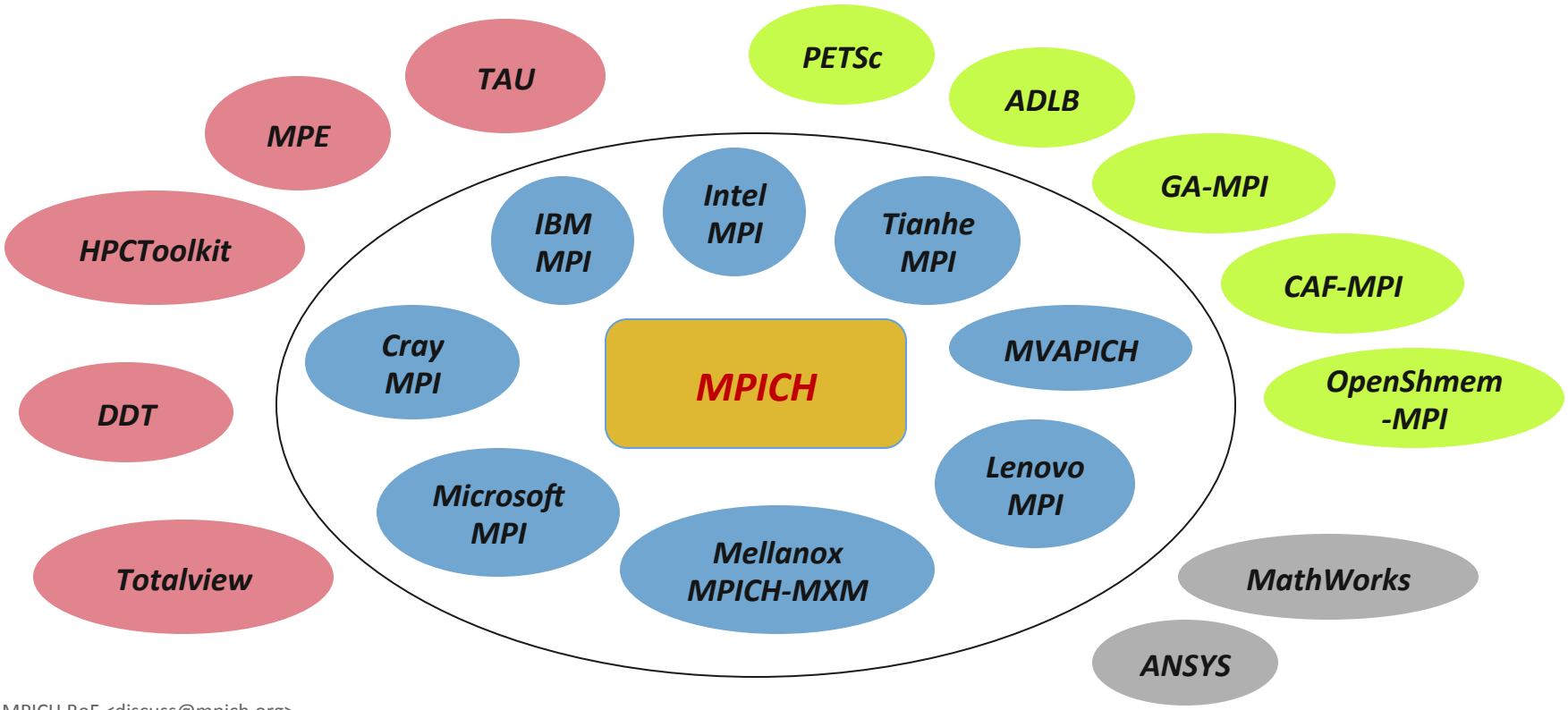
MPICH and it derivatives in the Top 10

1. ***Tianhe-2 (China): TH-MPI***
2. ***Titan (US): Cray MPI***
3. ***Sequoia (US): IBM PE MPI***
4. K Computer (Japan): Fujitsu MPI
5. ***Mira (US): IBM PE MPI***
6. ***Piz Daint (Germany): Cray MPI***
7. ***Stampede (US): Intel MPI and MVAPICH***
8. ***Juqueen (Germany): IBM PE MPI***
9. ***Vulcan (US): IBM PE MPI***
10. ***CS-Storm (US): Cray MPI***

MPICH and its derivatives power 9 of the top 10 supercomputers (Nov. 2014 Top500 rankings)

MPICH: Goals and Philosophy

- MPICH continues to aim to be the preferred MPI implementations on the top machines in the world
- Our philosophy is to create an "MPICH Ecosystem"



Status of MPI-3 Implementations (*)

	MPICH	MVAPICH	Cray MPI	Tianhe MPI	Intel MPI	IBM BG/Q MPI [1]	IBM PE MPICH [2]	MS MPI
NB collectives	✔	✔	✔	✔	✔	✔	Q4 '14	*
Neighborhood collectives	✔	✔	✔	✔	✔	✔	Q4 '14	
RMA	✔	✔	✔	✔	✔	✔	Q4 '14	
Shared memory	✔	✔	✔	✔	✔	✔	Q4 '14	✔
Tools Interface	✔	✔	✔	✔	✔	✔ [3]	Q4 '14	*
Non-collective comm. create	✔	✔	✔	✔	✔	✔	Q4 '14	
F08 Bindings	✔	✔	Q4 '14	✔	Q4 '14	✔	Q4 '14	
New Datatypes	✔	✔	✔	✔	✔	✔	Q4 '14	*
Large Counts	✔	✔	✔	✔	✔	✔	Q4 '14	*
Matched Probe	✔	✔	✔	✔	✔	✔	Q4 '14	*

Release dates are estimates and are subject to change at any time.

Empty cells indicate no *publicly announced* plan to implement/support that feature.

[1] Open source, but unsupported [2] Beta release [3] No MPI_T variables exposed * Under development

(*) Platform-specific restrictions might apply for all supported features

MPICH ABI Compatibility Initiative

- Runtime compatibility for MPI implementations
  - Explicit goal of maintaining ABI compatibility between multiple MPICH derivatives
  - Initial collaborators include:
    - MPICH (starting v3.1, Dec. 2013)
    - IBM PE MPI (starting v1.4, April 2014)
    - Intel MPI Library (starting v5.0, 2014)
    - Cray MPT (starting v7.0, June 2014)
  - More details at http://www.mpich.org/abi
- Open initiative: other MPI implementations are welcome to join

Argonne NATIONAL LABORATORY

IBM

intel

CRAY THE SUPERCOMPUTER COMPANY

Schedule

- Version 3.2 Overview
- User Presentations
  - Jeff Hammond
  - Brad Benton
- Version 3.3 Plans
- Vendor Panel
  - Cray
  - IBM
  - Intel
  - Mellanox
  - Microsoft
  - Riken / University of Tokyo
- Discussion

Argonne NATIONAL LABORATORY

MPICH: Current Status and 3.2 Release

Pavan Balaji

Computer Scientist

Group Lead, Programming Models and Runtime systems

Argonne National Laboratory

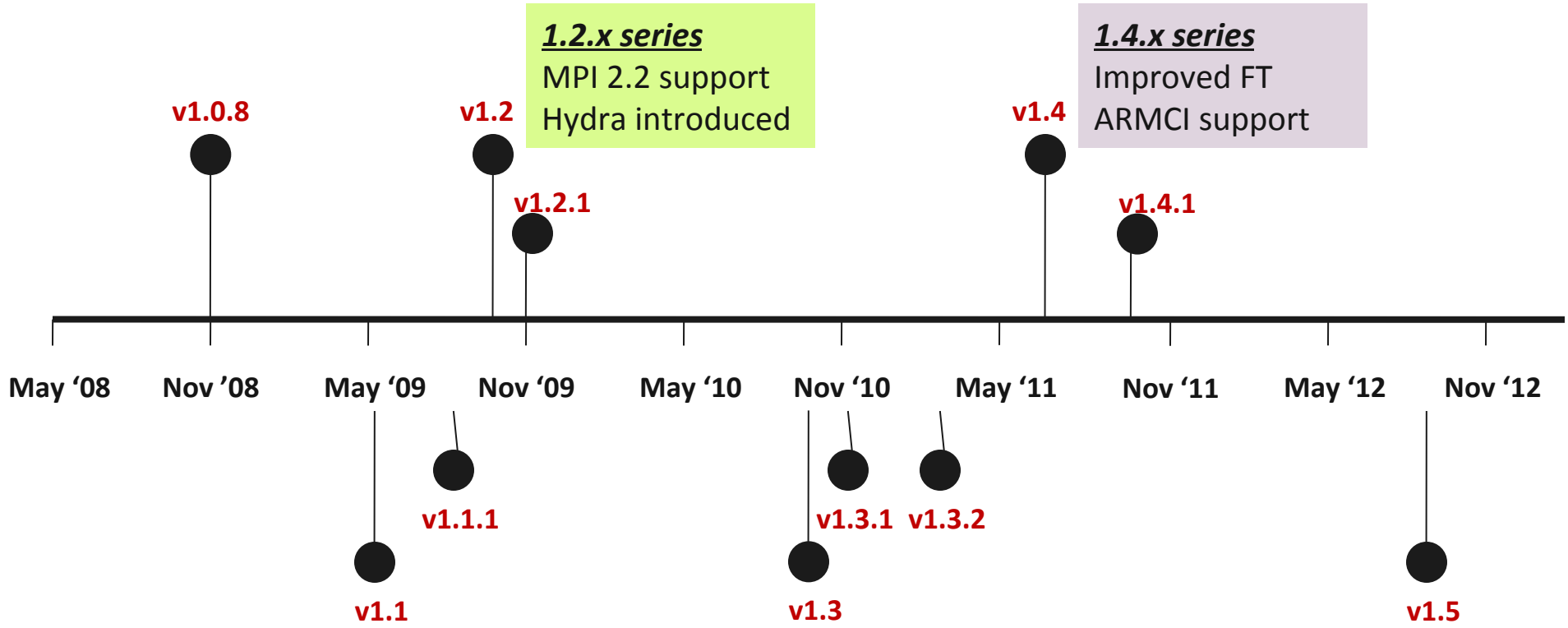
MPICH turns 22

U.S. DEPARTMENT OF ENERGY

MPICH-3.2

- MPICH-3.2 is the latest major release series of MPICH
  - Released mpich-3.2a2 on Saturday
  - mpich-3.2 GA release will be in May 2015
- Primary focus areas for mpich-3.2
  - Support for MPI-3.1 functionality (nonblocking collective I/O and others)
  - Fortran 2008 bindings
  - Support for the Mellanox MXM interface (thanks to Mellanox)
  - Support for the Mellanox HCOLL interface (thanks to Mellanox)
  - Support for OFED IB on Xeon and Xeon Phi (thanks to RIKEN)
  - Improvements to MPICH/Portals4
  - MPI-4 Fault Tolerance (ULFM)
  - Major improvements to the RMA infrastructure

MPICH Past Releases



1.2.x series
MPI 2.2 support
Hydra introduced

1.4.x series
Improved FT
ARMCI support

1.1.x series
MPI 2.1 support
Blue Gene/P
Nemesis
Hierarchical Collectives

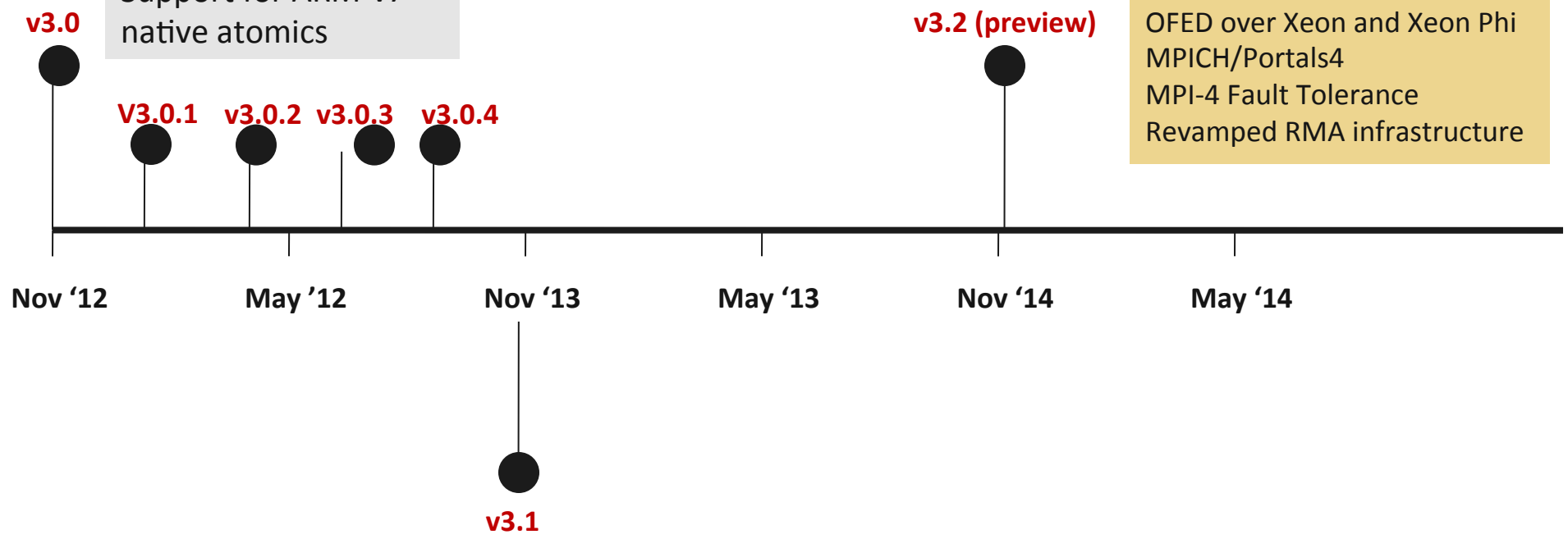
1.3.x series
Hydra default
Asynchronous progress
Checkpointing
FTB

1.5.x series
Blue Gene/Q
Intel MIC
Revamped Build System
Some MPI-3 features

MPICH 3.1 and Future Plans

3.0.x series
Full MPI-3 support
Support for ARM-v7 native atomics

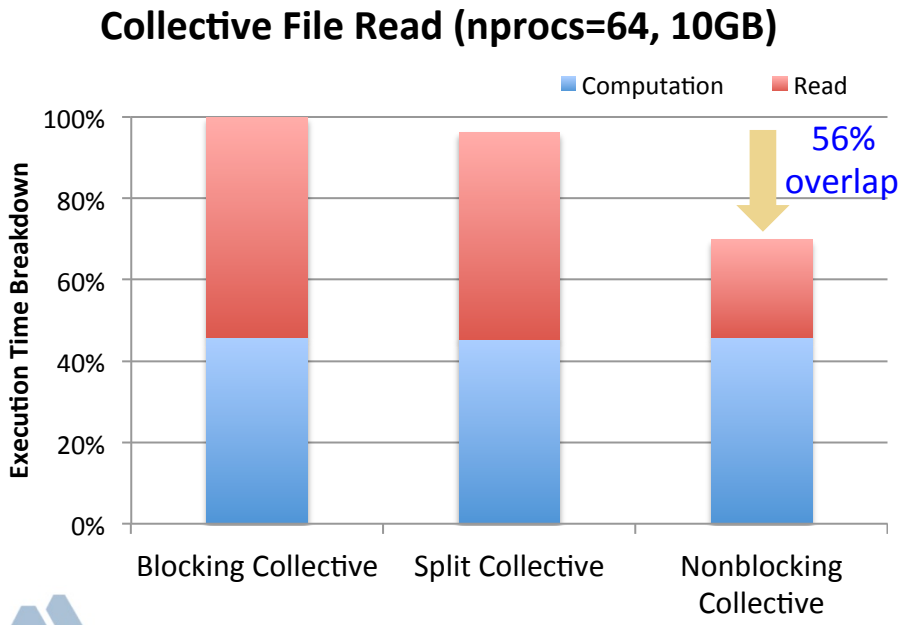
3.2.x series
MPI-3.1 functionality
Fortran 2008 bindings
Mellanox MXM
Mellanox HCOLL
OFED over Xeon and Xeon Phi
MPICH/Portals4
MPI-4 Fault Tolerance
Revamped RMA infrastructure

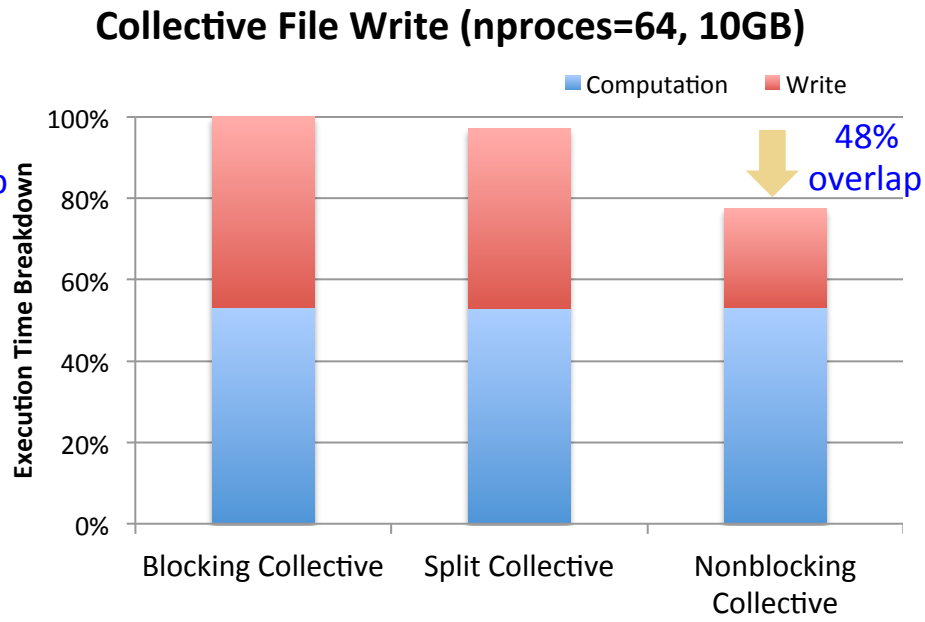


3.1.x series
MPICH ABI initiative
MPICH-PAMI code integration
Intel Xeon Phi improvements

Non-blocking Collective I/O

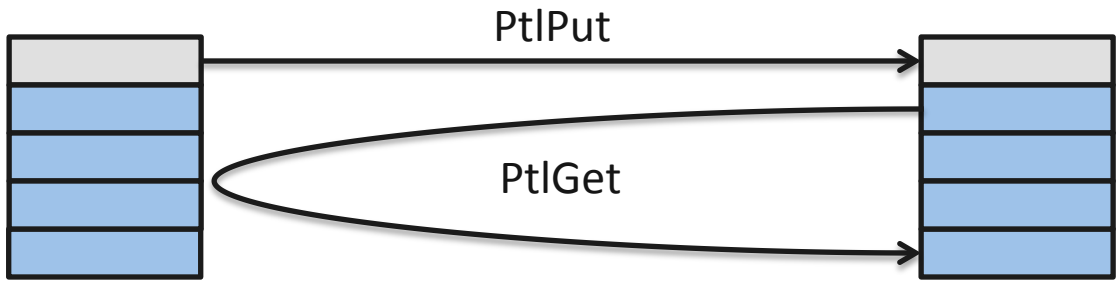
- Proposal for MPI 3.1 standard: Immediate versions of blocking collective I/O
  - **MPI_File_read_all**(MPI_File fh, void *buf, int count, MPI_Datatype datatype, MPI_Status *status)
  - **MPI_File_iread_all**(MPI_File fh, void *buf, int count, MPI_Datatype datatype, MPI_Request *request)
  - Same for **MPI_File_iread_at_all**, **MPI_File_iwrite_all**, and **MPI_File_iwrite_at_all**
- Supported in MPICH 3.2:
  - Implemented as MPIX functions and integrated with ROMIO in MPICH





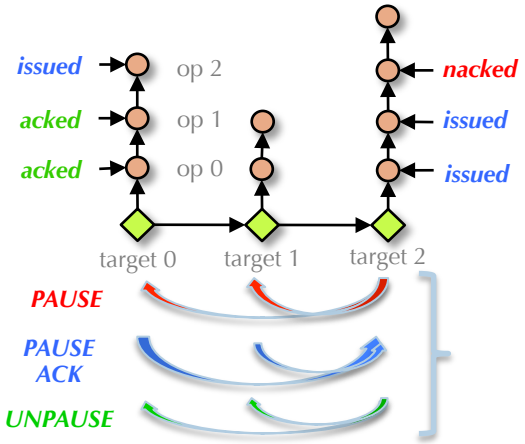
Portals4 netmod in MPICH

- Portals4:
  - Connectionless, hardware independent network API
- Portals4 netmod in MPICH 3.2:
  - Relies on Portals4 MEs (matching list entries)
    - Matched message queues implemented in hardware
  - Messaging design
    - Messages <= eager threshold in single PtlPut operation
    - Hybrid Put/Get protocol for larger messages (receiver gets the rest of the data)
    - Support for messages greater than implementation defined max_msg_size



Portals4 - Reliability

- Portals4 only offers semi-reliability
  - Packets dropped on the network are retransmitted
  - Packets dropped on the end-host are not retransmitted
- Multiple scenarios can cause Portals4 to go into flow control state
  - Event queue overflow
  - Unexpected buffer exhaustion
- MPICH adds a reliability layer (rportals)
  - Mandatory logging of all operations
  - Uses a separate EQ for origin side events
    - Queues operations if they will overflow the local EQ
    - Avoids silently dropping ACK packets
  - Recovery from flow control events
    - Global protocol to quiesce the network in rportals
      - Pause/Pause Ack/Unpause
    - NACKed operations are re-issued



Explicit flow-control when hardware cannot handle messages

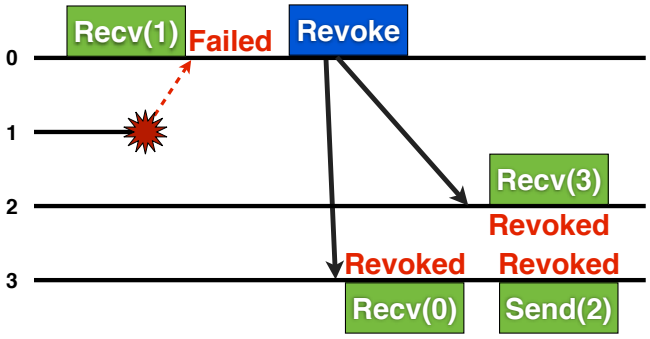
MPI-4 Fault Tolerance (User Level Failure Mitigation)

- Enable application-level recovery by providing minimal FT API to prevent deadlock and enable recovery
- Don't do recovery for the application, but let the application (or a library) do what is best.
- Only handling process failures currently

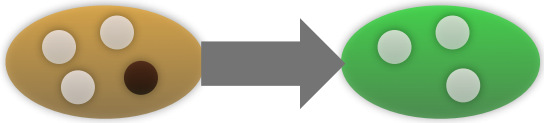
Failure Notification

MPI_ERR_PROC_FAILED

Failure Propagation



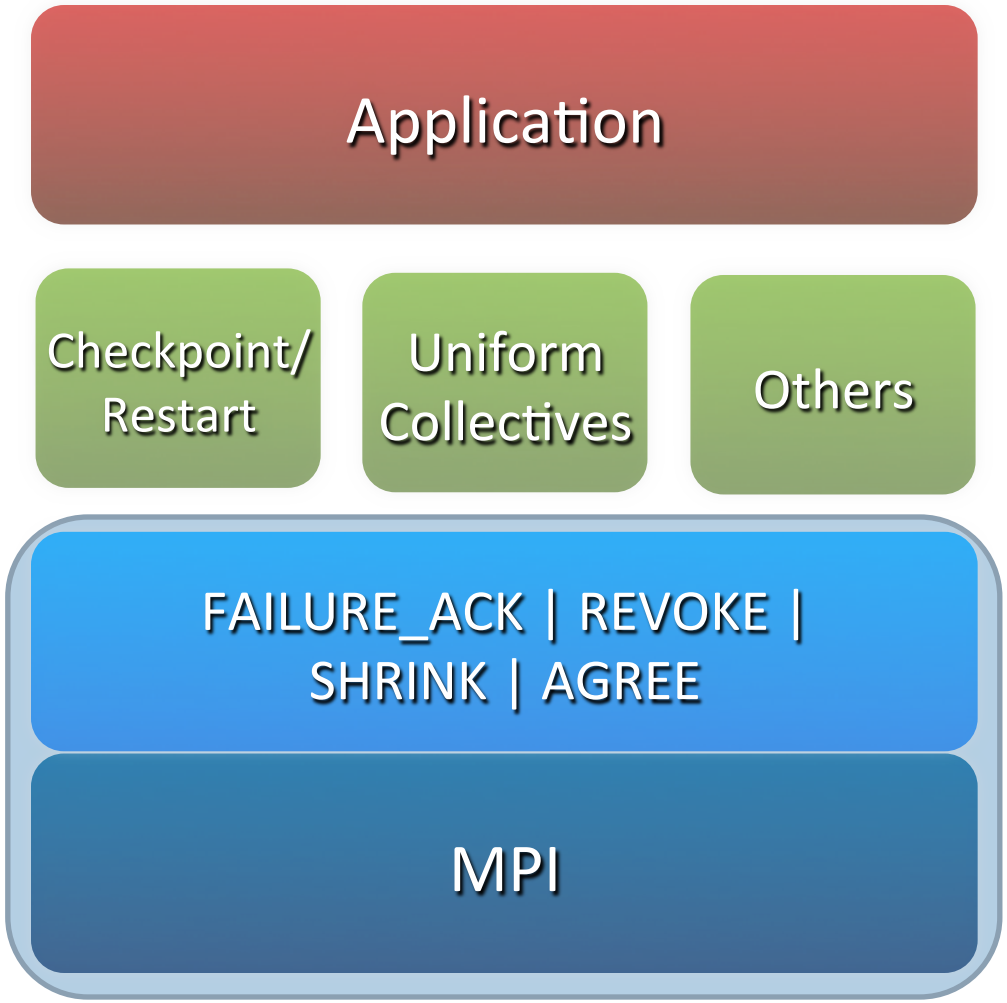
Failure Recovery



MPI_COMM_SHRINK()

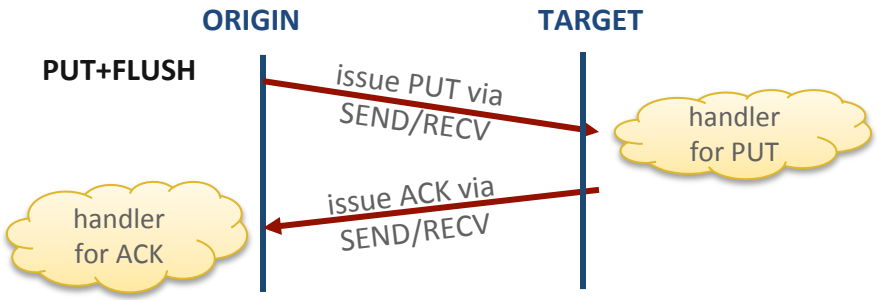
ULFM Continued

- RMA Windows & Files must be recreated after failures
- Minimal additions to encourage FT libraries rather than direct usage
  - Doesn't restrict use cases
- Reference implementation complete
  - MPICH implementation in progress
- Standardization in progress

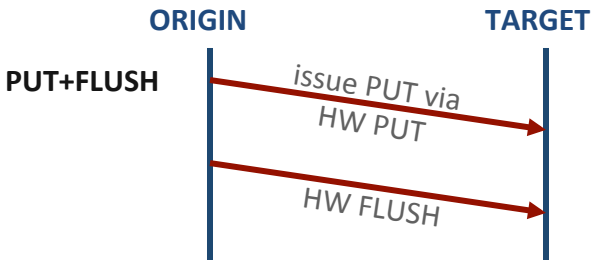


RMA Improvements for MPICH 3.2 (1/3)

- **Depending on how RMA operations are implemented, they can be divided into two categories:**
  - AM-based RMA operations, implemented in software using SEND/RECV in network module ← current release
  - Network-based RMA operations



AM-based operations: network module is not aware of RMA operation, treats it as two-sided communication packet

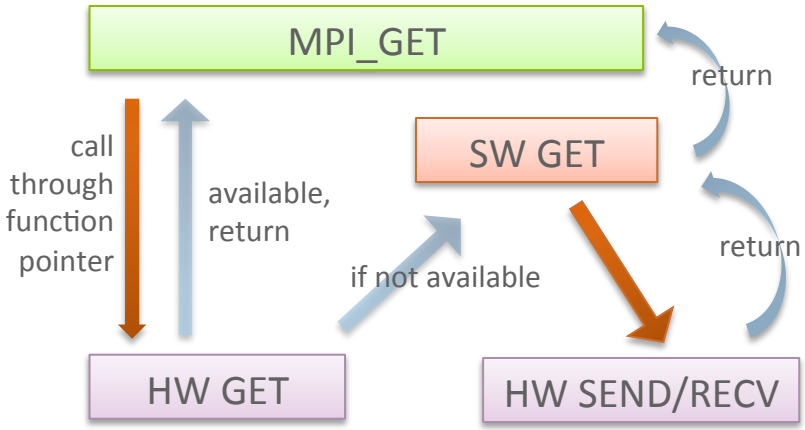


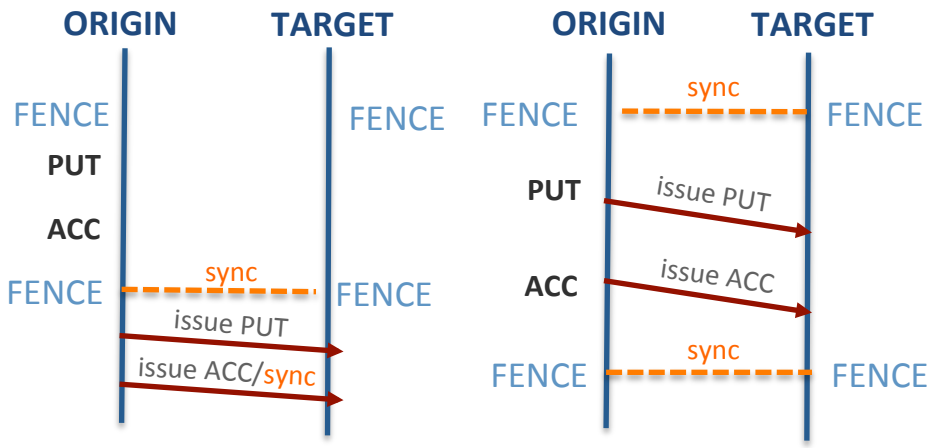
Network-based operations: network module is aware of RMA operation

- **Our improvements for RMA in MPICH 3.2**
  - Supporting network-based RMA operations
  - Improving AM-based operations in scalability and performance

RMA Improvements for MPICH 3.2 (2/3)

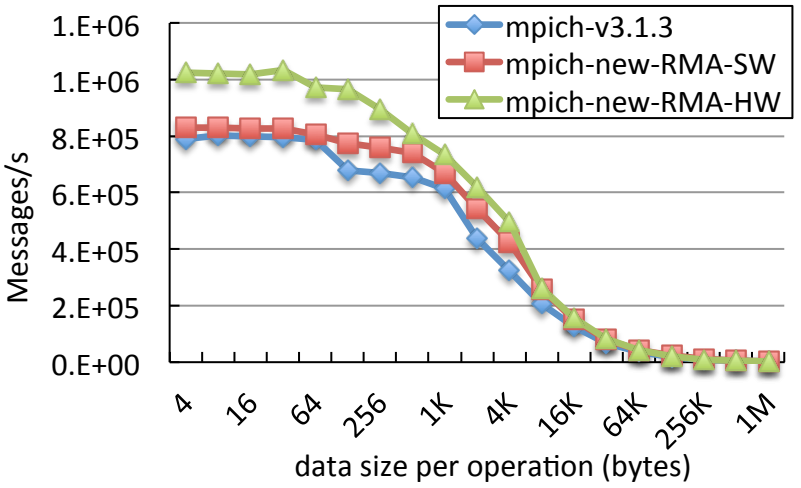
- **Support network-based RMA operations in hardware**
  - Algorithmic improvements
    - Eagerly issuing RMA operations as early as possible
  - Network module overwrites function pointers
  - Using software implementation as a fallback
  - Optimization for network orderings





Original algorithm: queuing up operations, delayed issuing

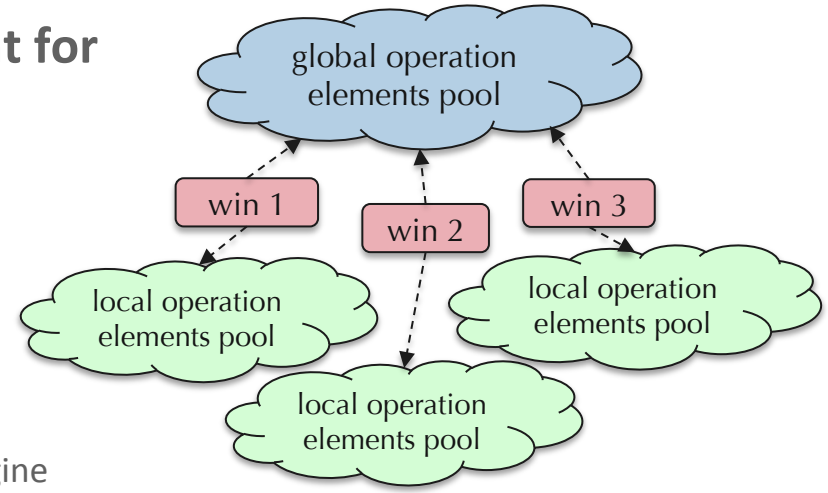
New algorithm: eagerly issuing

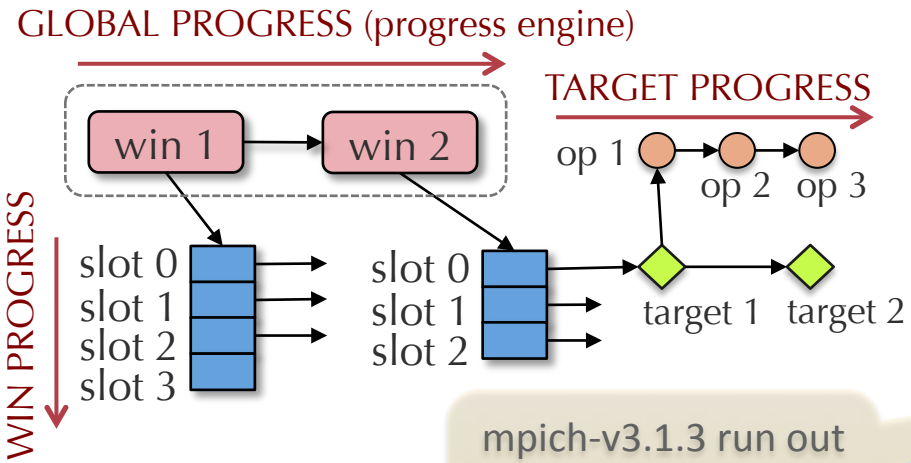


Message rate of MPI_GET (ch3:nemesis:mxm, IB QDR cluster)

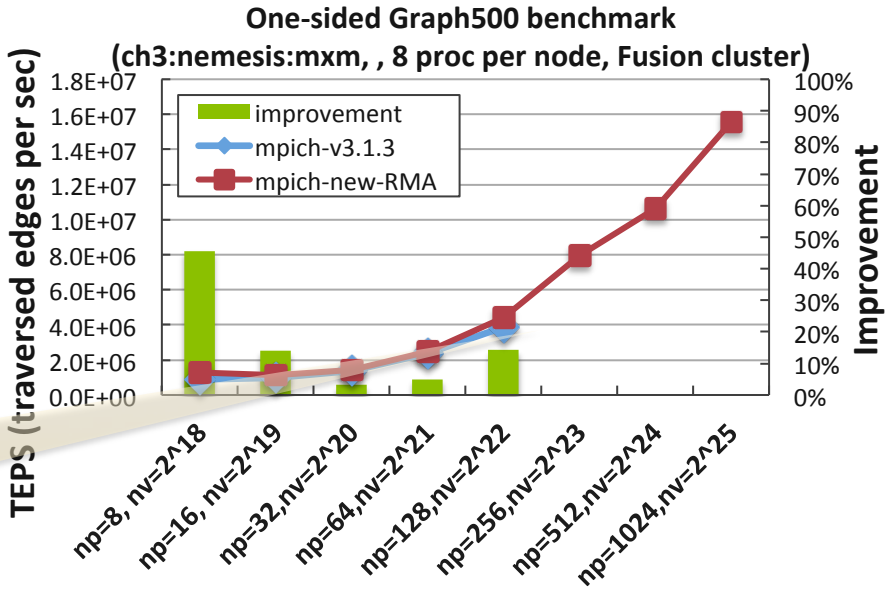
RMA Improvements for MPICH 3.2 (3/3)

- **Scalability and performance improvement for AM-based RMA operations**
  - No O(P) data structures
  - Sustainable resource management
    - Constant resources
  - Multi-level progress making strategy
    - Integrate RMA progress into MPICH progress engine





mpich-v3.1.3 run out of memory after 128 processes



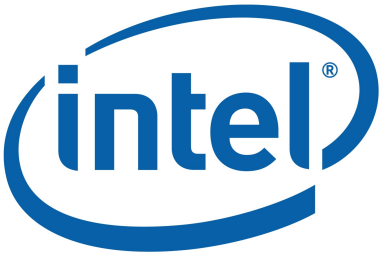
(Fusion cluster: 36GB memory per node with InfiniBand QDR interconnect)

NWChem and Global Arrays Applications using MPI-3 RMA

Jeff Hammond

Extreme Scalability Group
Parallel Computing Lab
Intel Corporation
Portland, OR
jeff_hammond@acm.org

18 November 2014



Legal Disclaimer & Optimization Notice

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

NWChem Overview

- **History:** Began at the dawn of the MPP age, before MPI; first MPP code in quantum chemistry; almost every code imitates it now.
- **Design:** Global Arrays programming model abstracted away explicit communication, was data-centric (i.e. what do you need to do with that matrix?).
- **Capability:** Very diverse collection of quantum chemical methodology and QM/MM.
- **Portability:** Runs on laptops/workstations (Linux, Mac and Cygwin), clusters (e.g. Infiniband) and supercomputers (e.g. Cray and IBM Blue Gene).

Over the past decade, the portability of NWChem has been limited almost entirely by ARMCI, the one-sided communication of Global Arrays...

Global Arrays model

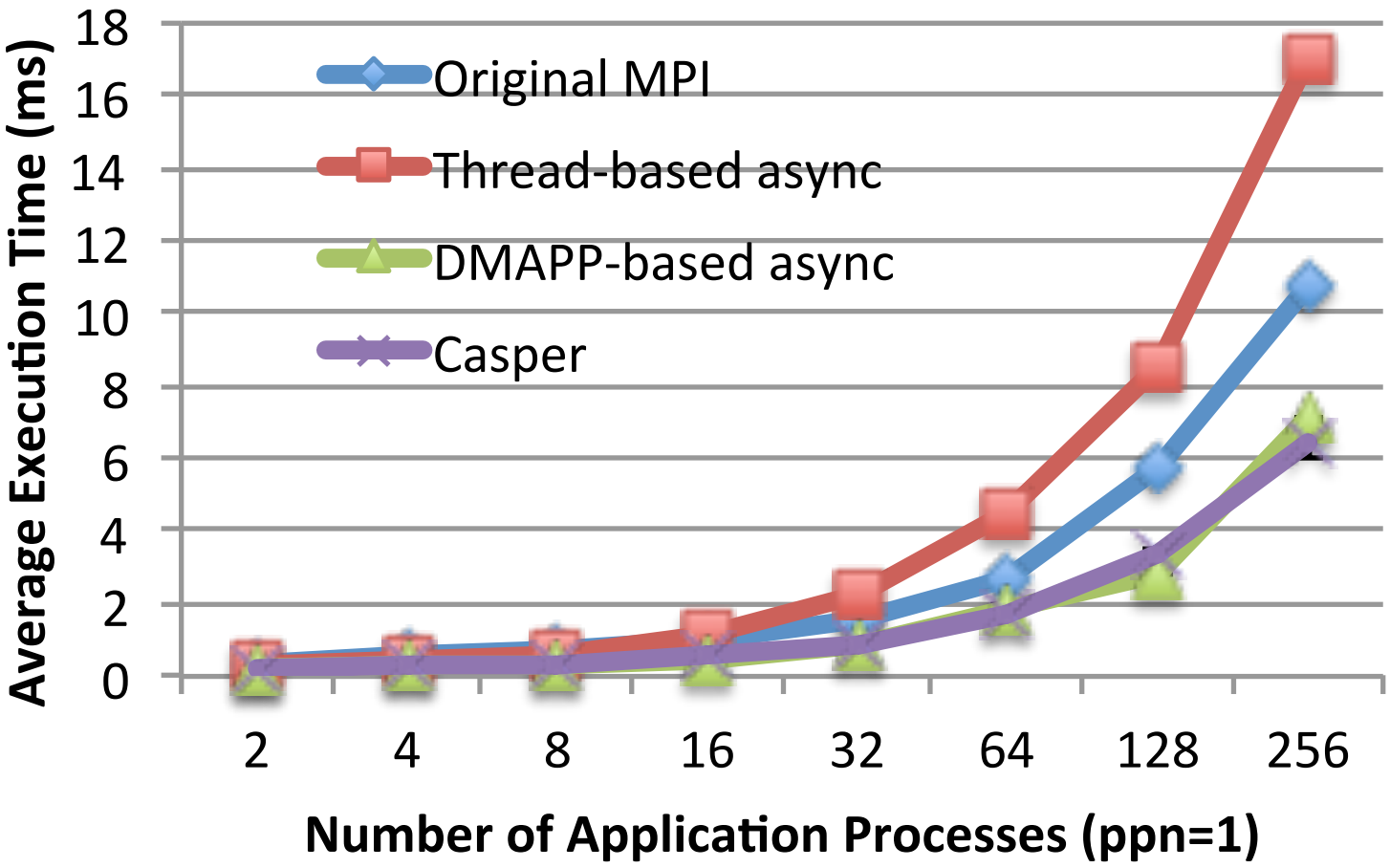
GA supports N-dimensional ($N < 8$) distributed arrays with regular (blocked and block-cyclic) and irregular (user-defined) distributions in all dimensions.

Direct local access is permitted but the primary programming model is **Get-Compute-Update**.

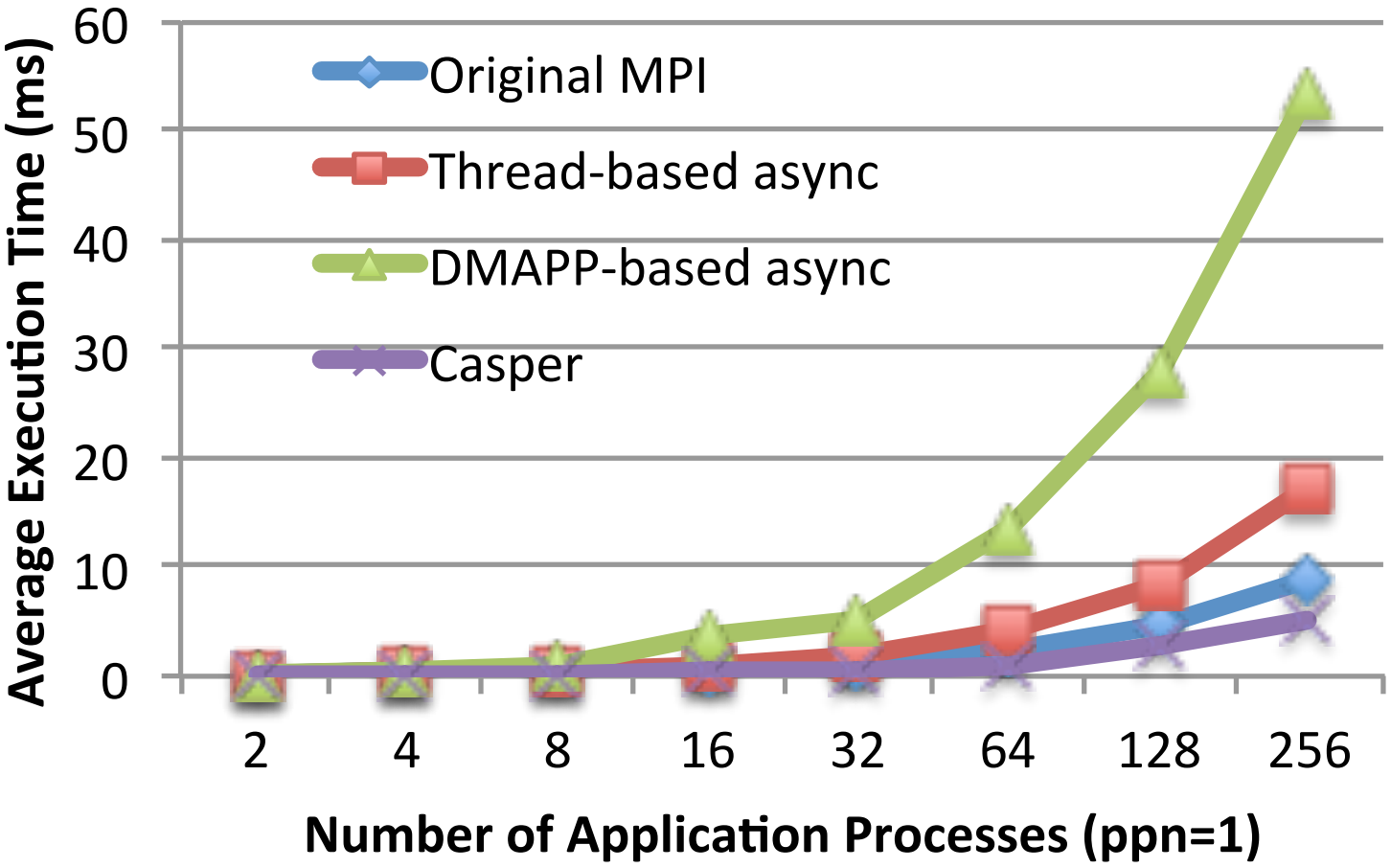
GA Template for Matrix Multiplication

```
Win_allocate(A,B,C)
Win_lock_all(A,B,C)
for i in I blocks:
   for j in J blocks:
      for k in K blocks:
         if Fetch_and_op(1,LONG,SUM,nxtval):
            Get,Win_flush block a(i,k) from A
            Get,Win_flush block b(k,j) from B
            Compute
      Accumulate,Win_flush_local c(i,j) block to C
Win_flush_all(C)
Win_unlock_all(A,B,C)
```

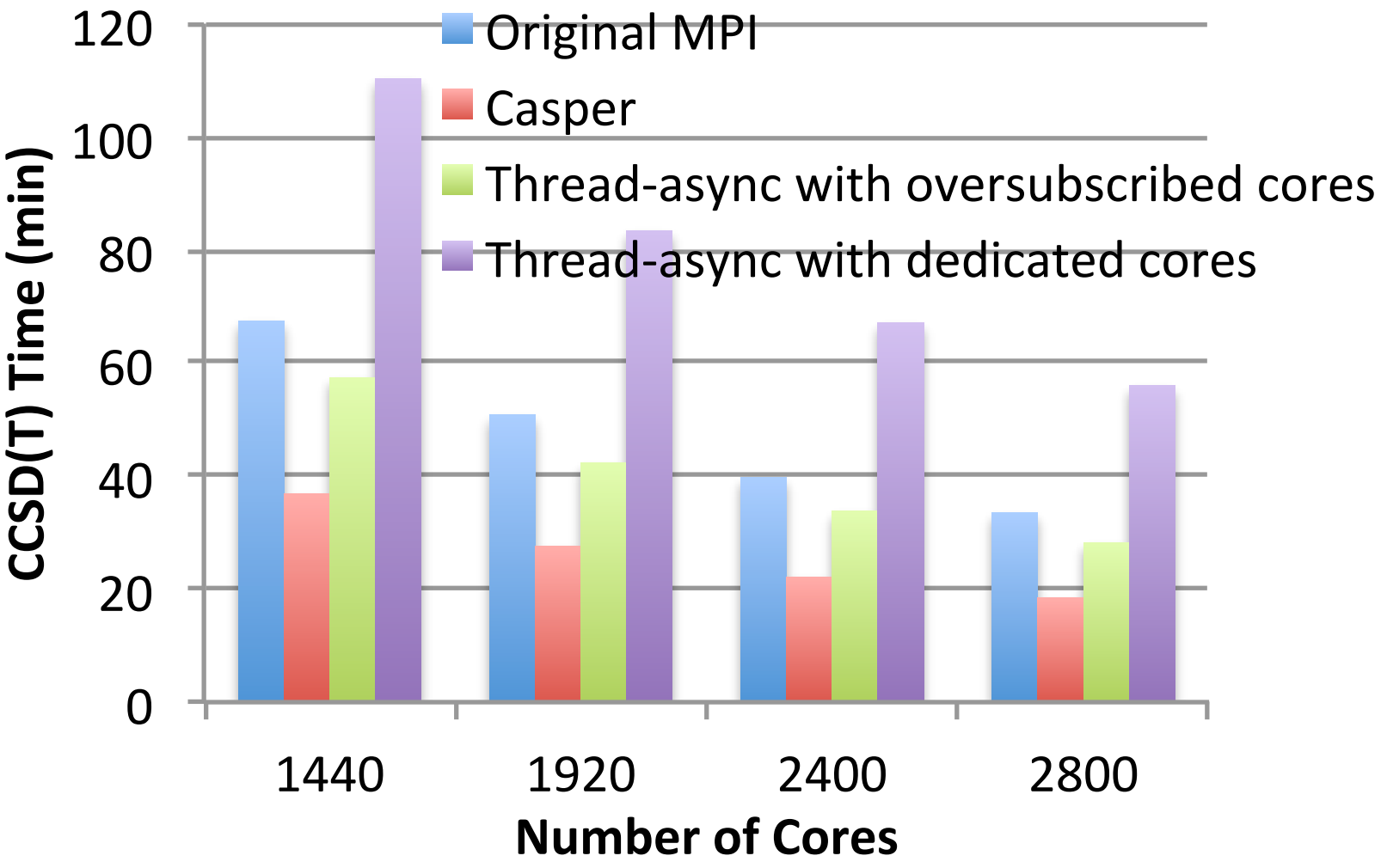
MPI-3 Put (Cray XC30)



MPI-3 Accumulate (Cray XC30)



NWChem with MPI-3 on Cray XC30



"The best performance improvement is the transition from the nonworking state to the working state." – John Osterhout

MPI-3 Osterhout Performance

- No native ARMCI port on Tianhe-2; ARMCI-MPI3 scaled to 8000 nodes.
- No native ARMCI port on Blue Gene/Q; ARMCI-MPI2 scaled to thousands of nodes.
- ARMCI native IB port extremely unstable; segfaults nearly all the time for large-memory jobs. ARMCI-MPI able to run near the memory limit without crashing.
- ARMCI native DMAPP port extremely unstable; fails every time in TCE. ARMCI-MPI running to 80K cores without crashing.

Summary: MPI-3 in MPICH and derivatives (Cray MPI, MVAPICH2, TH-MPI, Intel MPI, ...) is the most portable and robust and therefore scientifically productive way to run NWChem.

More information

NWChem

http://www.nwchem-sw.org

ARMCI-MPI

http://wiki.mpich.org/armci-mpi/index.php/Main_Page

AMD AND MPICH

MPICH BOF SC14

BRAD BENTON, AMD

AMD AND DOE DESIGNFORWARD PROGRAM

AMD

- **DesignForward Goal**

  "Initiate partnerships with multiple companies to accelerate the R&D of critical technologies needed for extreme-scale computing"

- **AMD awarded to explore remote task queuing as a fundamental primitive**

- **Extend Heterogeneous System Architecture (HSA) to multi-node systems**
  - Builds on key HSA features
    - User-level queuing
    - Shared virtual addressing

THE HETEROGENEOUS SYSTEM ARCHITECTURE (HSA)

AMD

- **The Heterogeneous System Architecture (HSA)**
  - Standard platform for heterogeneous computing initiated by AMD
- **HSA Foundation** (http://hsafoundation.com)
  - Independent standards body with over 40 members, including Argonne
- **HSA Features**
  - User-level task queuing:
    No OS or driver in the dispatch path
  - Coherent virtual addressing
    Enables direct data sharing between CPU and device, including pointers
  - Architected Queuing Language
    Consistent across vendors

E**X**TENDED **T**ASK **Q**UEUING (**XTQ**)

AMD

- **HSA+RDMA**
  - Under HSA, enqueuing a task is just a sequence of user-level writes
  - RDMA enables user-level writes across the network
  - Put them together and you have user-level, cross-node task queuing

- **Extended Task Queuing (XTQ)**
  - Extends by adding a "Task Put" operation
  - Similar to an RDMA Put
  - Node A can enqueue a task on node B's GPU without involving node B's CPU
  - Applying task queues to CPU scheduling → active messages

AMD AND MIPCH

AMD

- **Simulation Research using Portals 4**
  - Using Portals 4 API for NIC interface
  - Supports both MPI & PGAS models
- **Extend Portals 4 for XTQ Capability**
  - Provides for both GPU acceleration and dynamic tasking
- **MPI and Portals 4 is key to our research**
  - Using Portals 4 API as the basis for our XTQ-NIC interface (in simulation)
  - Open source MPI library necessary
  - MPI-3 compliant library with Portals 4 support is crucial
  - MPICH satisfies these requirements
  - Argonne team very receptive to mutual collaboration

AMD AND MPICH COLLABORATION

AMD

- **Solidifying Portals 4 MPICH Network Module**
  - Have a dedicated mailing list dedicated to AMD/Argonne Portals 4 discussions
  - MPICH team very responsive to resolving implementation issues
    - Informs us of progress and fixes
    - Provides access to latest versions through nightly builds

- **Areas of Research Collaboration**
  - HSA-Aware MPICH:
    GPU exploitation for computational functions, e.g., reductions, accumulate, etc.
  - Dynamic tasking:
    Exploring XTQ as an infrastructure component in support of Active Messages, following the work of Xin Zhao, Pavan Balaji, *et al.*

DISCLAIMER & ATTRIBUTION

AMD

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION

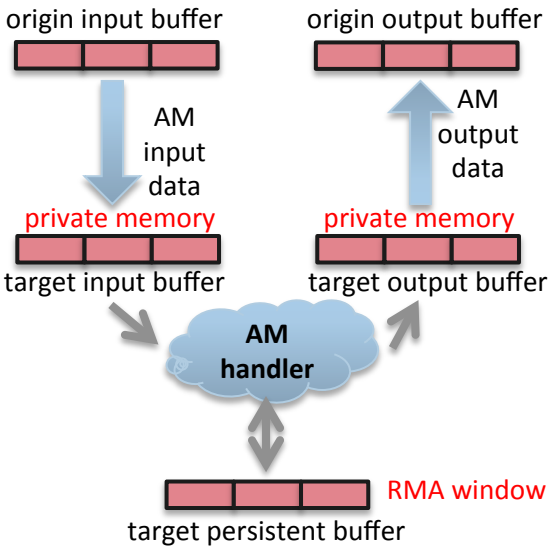
MPICH 3.3

- General Availability release planned for SC '16
  - Prerelease versions available throughout 2016
- New features planned (subject to change)
  - Active Messages
  - Topology Improvements
  - GPU Support
  - Improve CPU idling while blocking
  - Support for PMI-3
  - Scalable VC Infrastructure
  - libfabric Support
  - MPI+Threads Improvments

Generalized and MPI-Interoperable Active Messages

MPI-AM: an MPI-interoperable framework that can dynamically manage data movement and user-defined remote computation.

- **Streaming AMs**
  - Define "**segment**"--- minimum number of elements for AM execution
  - Achieve pipeline effect and reduce temporary buffer requirement
- **Explicit and implicit buffer management to guarantee correct execution and improve performance**
  - **System buffers**: eager protocol, not always enough
  - **User buffers**: rendezvous protocol, guarantee correct execution
- **Correctness semantics**
  - **Memory consistency**
    - MPI runtime ensures consistency of window memory
  - **Flexible ordering and concurrency supports for performance**
- **Compatible with MPI-3 standard**

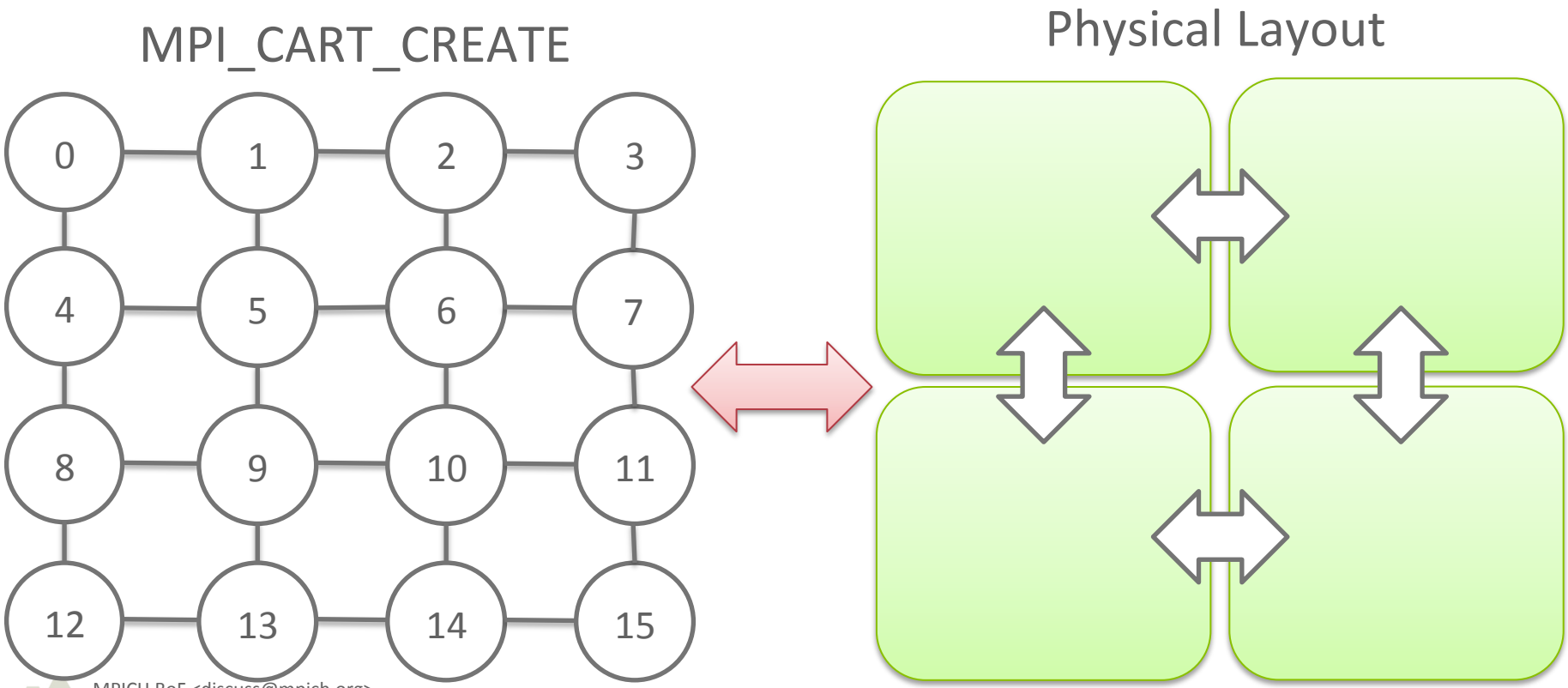


MPI-AM workflow

- [ICPADS'13] MPI-Interoperable Generalized Active Messages. Xin Zhao, Pavan Balaji, William Gropp, Rajeev Thakur. In proceedings of the 19th IEEE International Conference on Parallel and Distributed Systems.
- [ScalCom'13] Optimization Strategies for MPI-Interoperable Active Messages. Xin Zhao, Pavan Balaji, William Gropp, Rajeev Thakur. In proceedings of the 13th IEEE International Conference on Scalable Computing and Communications. Best Paper Award.
- [CCGrid'13] Towards Asynchronous and MPI-Interoperable Active Messages. Xin Zhao, Darius Buntinas, Judicael Zounmevo, James Dinan, David Goodell, Pavan Balaji, Rajeev Thakur, Ahmad Afsahi, William Gropp. In proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.

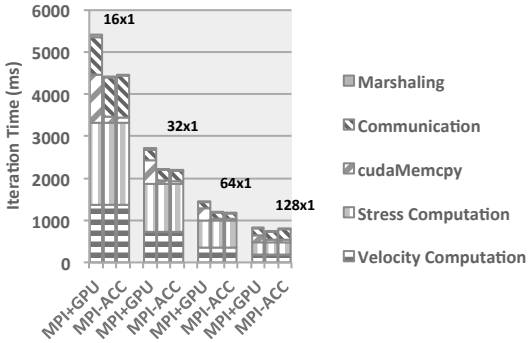
Topology Improvements

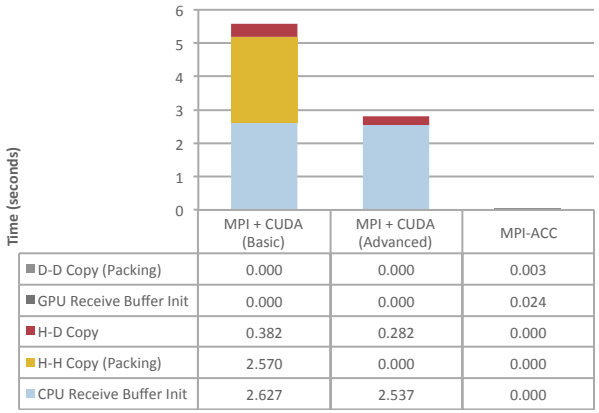
- Improve MPI topology mapping to physical hardware
  - Early Work by Ahmad Afsahi
- Integrate with topology discovery tools:
  - hwloc, ibtracert, scotch



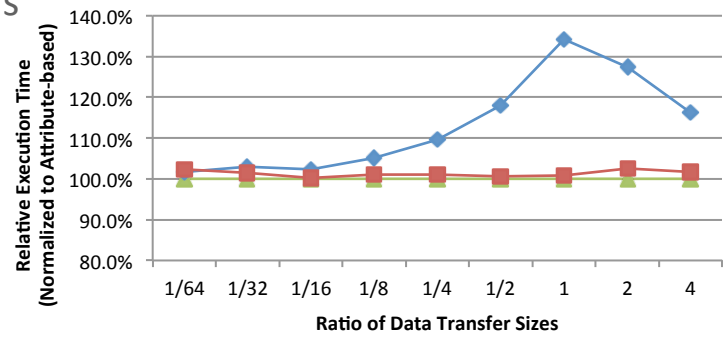
Accelerator/Coprocessor Support

- Exchange data among coprocessors' memory
  - Integration and extension from our MPI-ACC related research
  - Integrated and extensible approach
  - Enables accelerator pointers in MPI calls
  - Coprocessors as "first-class citizens" in hybrid clusters
  - Generic approach, feature independent
- Coding productivity + performance
  - DMA-assisted intranode communication
  - Automatic overlapped GPU & network transfers
  - Balancing of communication
  - Naturally expressing communication targets
  - Decouple application logic from GPU communications
  - Improve scalability and portability
- Datatype and communicator attributes
  - Richer synchronization semantics vs. UVA approach
  - Performance and correctness
  - Alleviates user management
  - Best with interleaved MPI and GPU operations



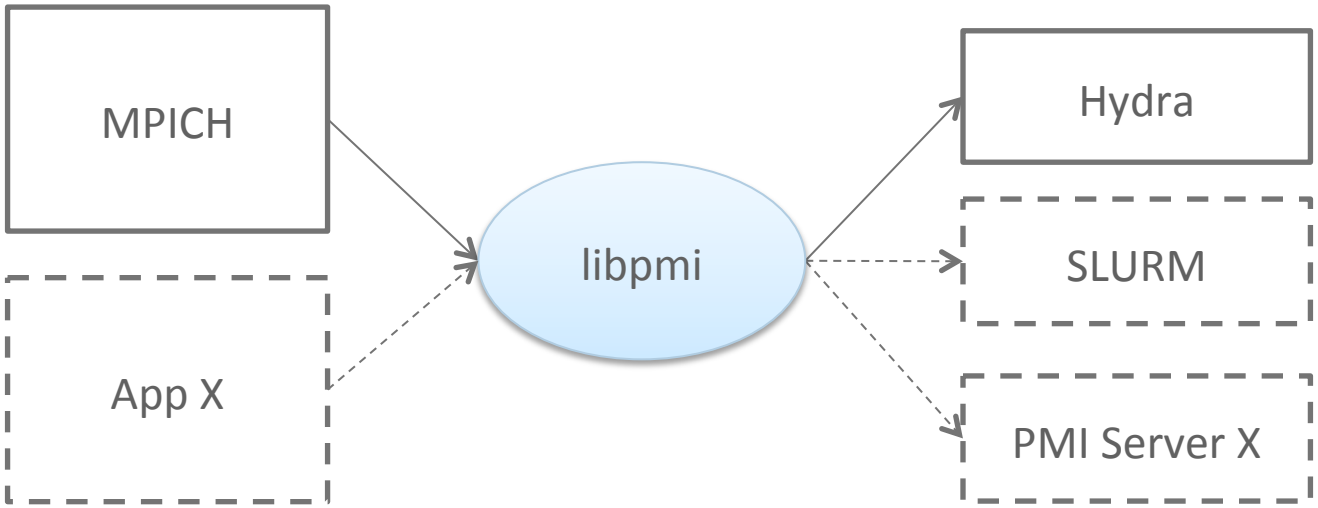


	MPI + CUDA (Basic)	MPI + CUDA (Advanced)	MPI-ACC
D-D Copy (Packing)	0.000	0.000	0.003
GPU Receive Buffer Init	0.000	0.000	0.024
H-D Copy	0.382	0.282	0.000
H-H Copy (Packing)	2.570	0.000	0.000
CPU Receive Buffer Init	2.627	2.537	0.000



PMI-3 (PMI Next)

- Goal of a new, community defined PMI standard
  - Unified API (no more PMI vs. PMI2)
  - Backwards compatibility with existing PMI servers
- Independent PMI client library (libpmi)
  - Buildable outside of MPICH
  - Support for multiple PMI servers (Hydra, SLURM, etc.)

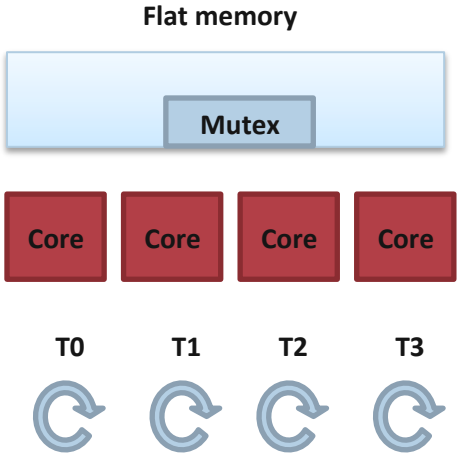


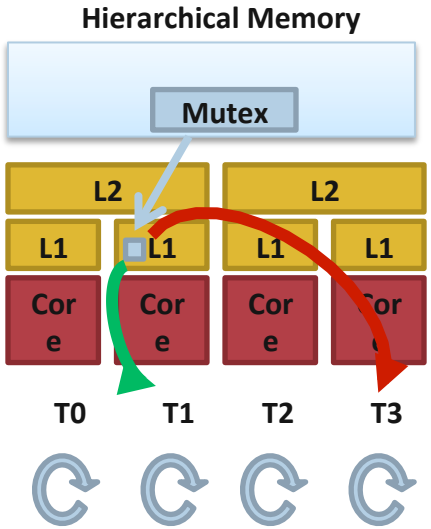
Scalable VCs

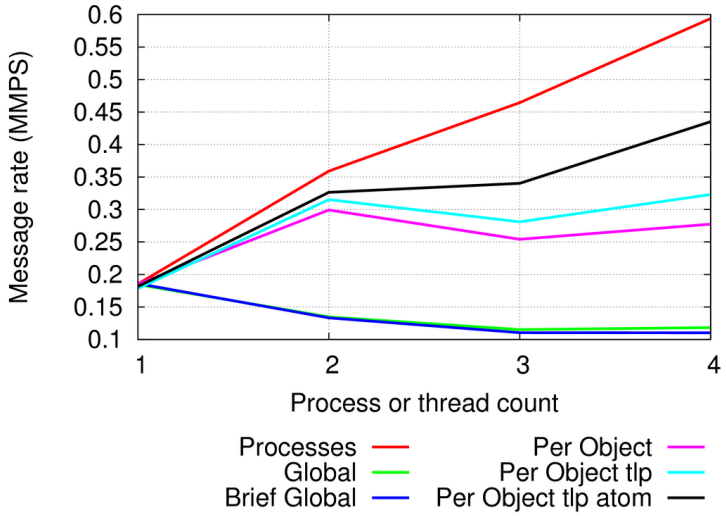
- What are VCs?
  - VCs or "virtual connections" in MPICH are data structures that contain the information needed for one MPI process to communicate with another
- Scalability challenges
  - All processes maintain a table of connection information for all other processes
  - A single VC in MPICH requires about 4KB of memory
    - 1,000,000 processes = 4GB of VC data
- Possible solutions
  - Locating VC data in a shared memory segment accessible to all ranks on a node
  - In-memory compression

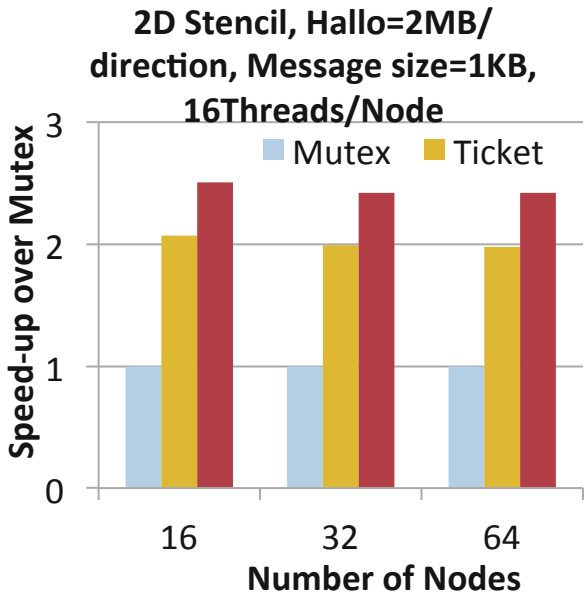
Concurrency Optimizations to MPI

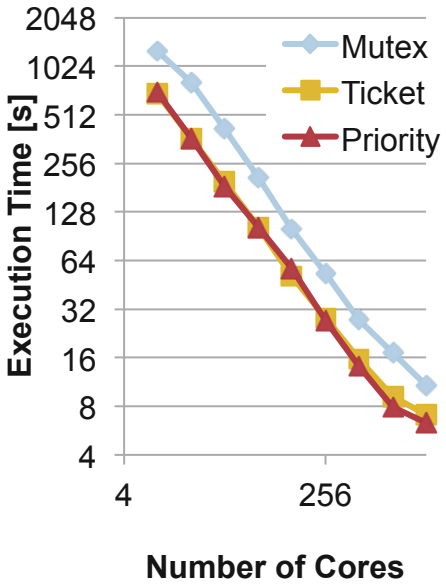
- Lock granularity:
  - Smaller critical sections
  - Lock-free algorithms
- Lock scheduling
  - Lock scheduling fairness (instead of OS scheduling)
  - Prioritized lock scheduling





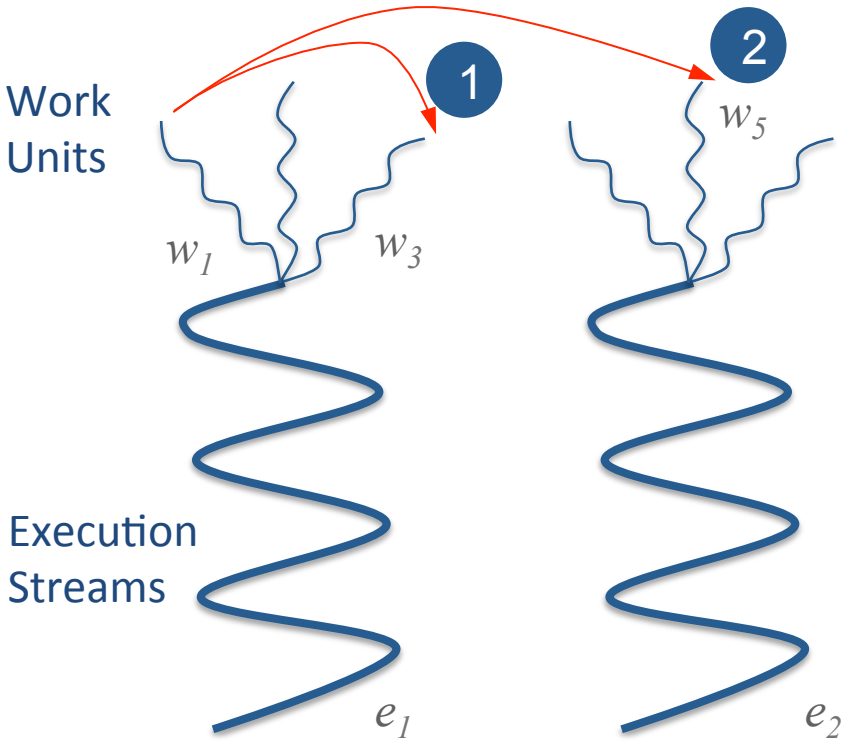






User-Level Threads Ongoing Works: Fine-grained Context-aware Thread Library

- **Two Level of Threads**
  - Execution stream: a normal thread
  - Work unit: a user level thread or a tasklet with a function pointer
- **Avoid Unnecessary lock/unlock**
  - Case 1: switch the execution to another work unit in the same execution stream **without unlock**
  - Case 2: switch to another execution stream, call unlock
- **Scheduling Work Units in Batch Order**
  - Work units in the same execution stream will be batch executed



Vendor Panel

- Cray – Heidi Poxon
- IBM – Craig Stunkel
- Intel – Bill Magro
- Lenovo – Chulho Kim
- Mellanox – Rich Graham
- Microsoft – Fab Tiller
- Riken / University of Tokyo – Yutaka Ishikawa

Cray MPT Update

SC14 - MPICH BOF

Safe Harbor Statement

This presentation may contain forward-looking statements that are based on our current expectations. Forward looking statements may include statements about our financial guidance and expected operating results, our opportunities and future potential, our product development and new product introduction plans, our ability to expand and penetrate our addressable markets and other statements that are not historical facts. These statements are only predictions and actual results may materially vary from those projected. Please refer to Cray's documents filed with the SEC from time to time concerning factors that could affect the Company and these forward-looking statements.

MPT 7.0 released in June 2014

- **Based on MPICH 3.1**
- **MPI-3 compliant**
- **Additional content:**
  - Initial MPICH ABI compatibility release
  - Optimizations for MPI_Bcast and MPI_Alltoallv
  - Optimizations for MPI-3 RMA
  - Support for Intel Haswell, Intel Xeon Phi (Knights Corner) and NVIDIA Tesla K40 on Cray XC systems

What's Next . . .

- **Plan to release MPT 7.1 in December 2014**
  - Based on MPICH 3.1.2
  - Includes Fortran 2008 bindings
- **Working on Cray-MPICH Fault Tolerance prototype**
  - Based on ANL MPICH implementation (master branch)
  - Use "xtnmi" from the SMW to simulate compute node failures
  - Relies on Fault Tolerance capabilities of Cray PMI to detect "fail stop" nodes
- **Plans for next year**
  - Incrementally add implementation specific information for MPI-3 Tools Interface
  - Upgrade to MPICH 3.2 with MPI-3.1 support
    - Hoping ANL's implementation will be binary compatible

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA, and YARCDATA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.

IBM MPI Directions

Craig Stunkel,
Robert Blackmore
Alan Jea
Nysal Jan
Sameer Kumar
Sameh Sharkawi

IBM Messaging Future Plans

- New IBM PE Runtime Edition for Linux on Power V2.1 GA in 12/2014
  - MPI-3 RMA and Non blocking collectives
  - Optimized MPICH and PAMI on Power8 over IB
  - Performance enhancements for Collectives and GPFS optimizations in MPI IO
- Develop a fast and scalable messaging stack for CORAL

Long Term Goals

- Scalable process managers that use PMI-x
  - Minimize MPI startup time
  - Be compatible with open source process managers
- GPU Support
  - Optimized GPU direct support when message is allocated in GPU memory
    - Minimize latency
    - Minimize copying when RDMA is available directly from GPU memory
- In general, enhance OpenPOWER systems
  - DCT routing on IB
  - Leverage/optimize on other future enhancements from Mellanox

Intel® MPI Library @SC14 update

William R. Magro

Chief Technologist, Technical Computing Software

Intel Corporation

SC'14 MPICH BoF

November 18, 2014

Intel® MPI Library at SC14

Intel MPI Library 5.0.2 released

- 256k ranks with HPLINPACK
- Performance tuning on latest Intel® Xeon E5 processors with Mellanox* and Intel® True Scale fabrics
- Multi-threaded optimized library used by default
- Linux* Cross-Memory Attach support
- More collective algorithms
- Improved start-up on Intel True Scale fabric
- Tools support on selected ranks (-gtool)
- Performance snapshot support

HPCG on Stampede submission with Intel MPI Library in Top-10

Now running on Microsoft* Azure – see it in the Microsoft* booth

Intel® MPI Library & MPICH collaboration

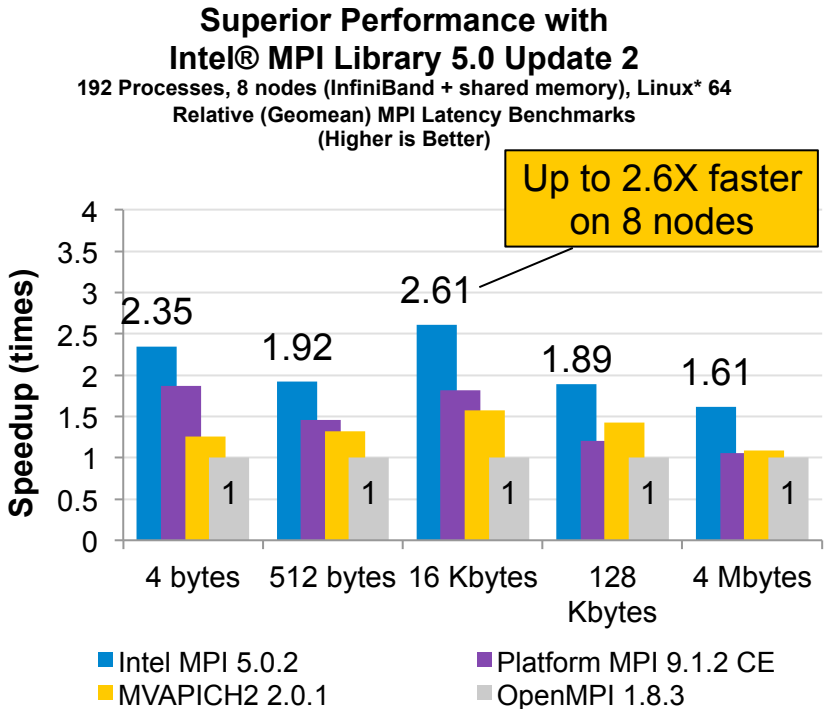
MPICH ABI compatibility initiative

- Agreed on binary names format, dependencies
- Established ABI compatibility regular testing

Code upstream

- Created intel-mpich repository; contribution process is in place
- Bugfix patches upstreamed with the new process
- More contributions in the pipeline:
  - Netmod supporting OpenFabrics* libfabric (Scalable Fabric Interface)
  - Hydra hierarchical launch
  - Wait mode

Intel® MPI Library 5.0.2 Performance Intel® Xeon® processor



Configuration Info:

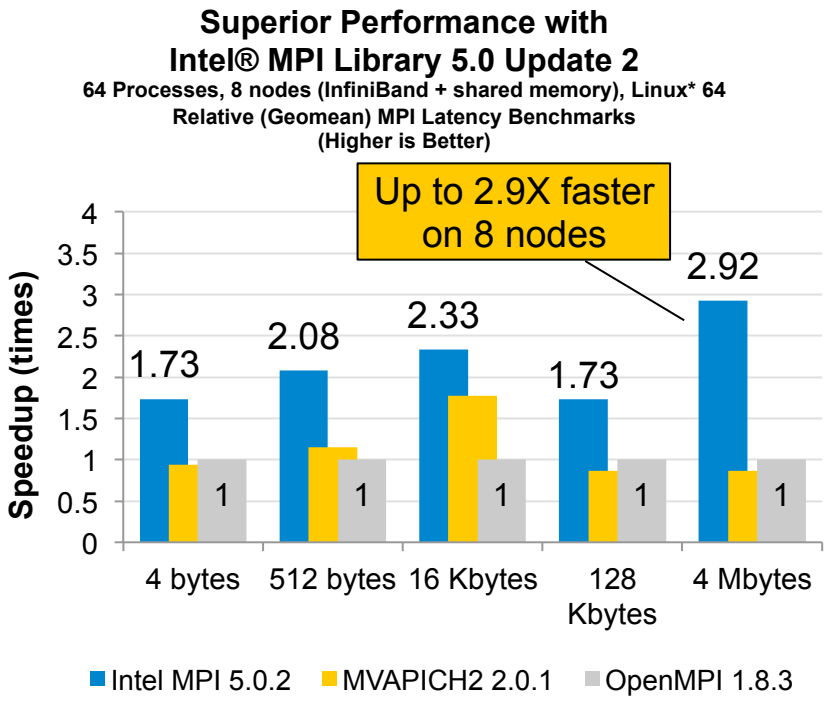
Hardware: CPU: Dual Intel® Xeon E5-2697v2@2.70Ghz; 64 GB RAM. Interconnect: Mellanox Technologies* MT27500 Family [ConnectX*-3] FDR.

Software: RedHat* RHEL 6.2; OFED 3.5-2; Intel® MPI Library 5.0.2; Intel® MPI Benchmarks 4.0 (built with Intel® C++ Compiler XE 15.0.0.090 for Linux*);

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804

Intel® MPI Library 5.0.2 Performance Intel® Xeon Phi™ processor



Configuration Info:

Hardware: Intel® Xeon® CPU E5-2680 @ 2.70GHz, RAM 64GB; Interconnect: InfiniBand, ConnectX adapters; FDR. MIC: C0-KNC 1238095 kHz; 61 cores. RAM: 15872 MB per card.

Software: RHEL 6.5, OFED 3.5.2-MIC, MPSS Version: 3.4.1, Intel® C/C++ Compiler XE 15.0.0.090, Intel® MPI Benchmarks 4.0.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804

OFA (Open Fabric Alliance): Enabling High-Performance Networking

- **Open-Source** Fabric Software
- Launched around InfiniBand verbs
- Created **OFIWG**:

"Develop an extensible, open source framework and interfaces aligned with ULP and application needs for high-performance fabric services"

- Implementation: libfabric + Scalable Fabrics Interface (SFI)
- Bootstrapping the community effort: MPICH Netmod with tag matching interface support

- **Join the community:** http://lists.openfabrics.org/mailman/listinfo/ofiwg
- **Get the code:** https://github.com/ofiwg/libfabric

Legal Disclaimer & Optimization Notice

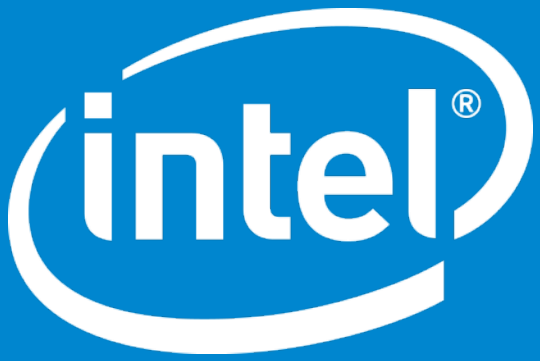
INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804


intel®

Mellanox – MPICH

Rich Graham

SC14 MPICH BOF | 2014

Mellanox TECHNOLOGIES

Connect. Accelerate. Outperform.™

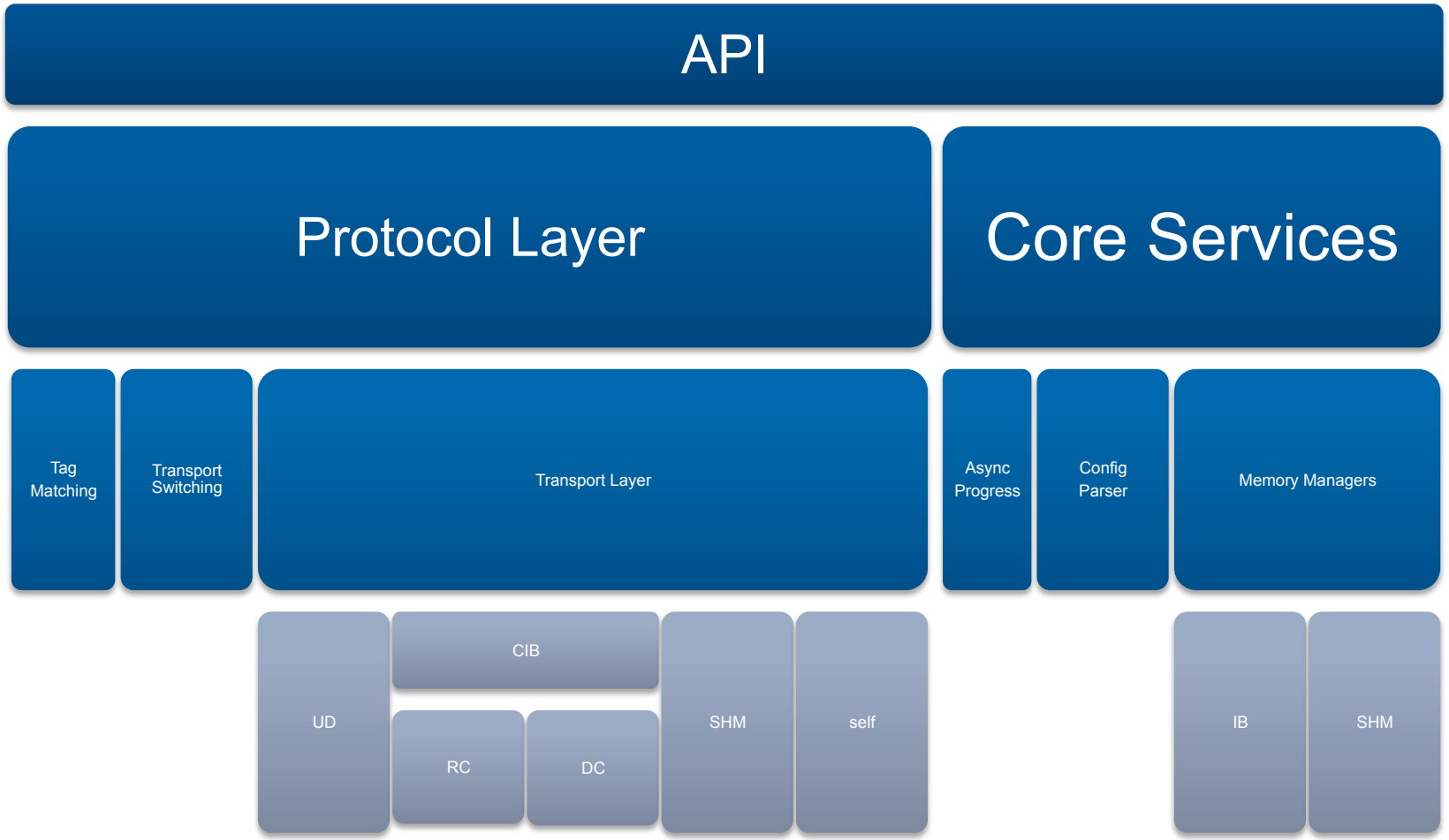
Mellanox Messaging - MXM

Point-to-Point Communication Library

Highlights

- Reliable Communication
- Multiple transport types supported
  - UD, RC, DC, Shared memory, Loopback
- Hybrid Transport Mechanism
- Communication library agnostic (standalone)
  - MPI
  - PGAS – OpenSHMEM, UPC
- Optimize for Mellanox specific capabilities
  - UMR, ODP, Accelerated verbs, Dynamically Connected Transport
- Tag matching

High-Level Architecture



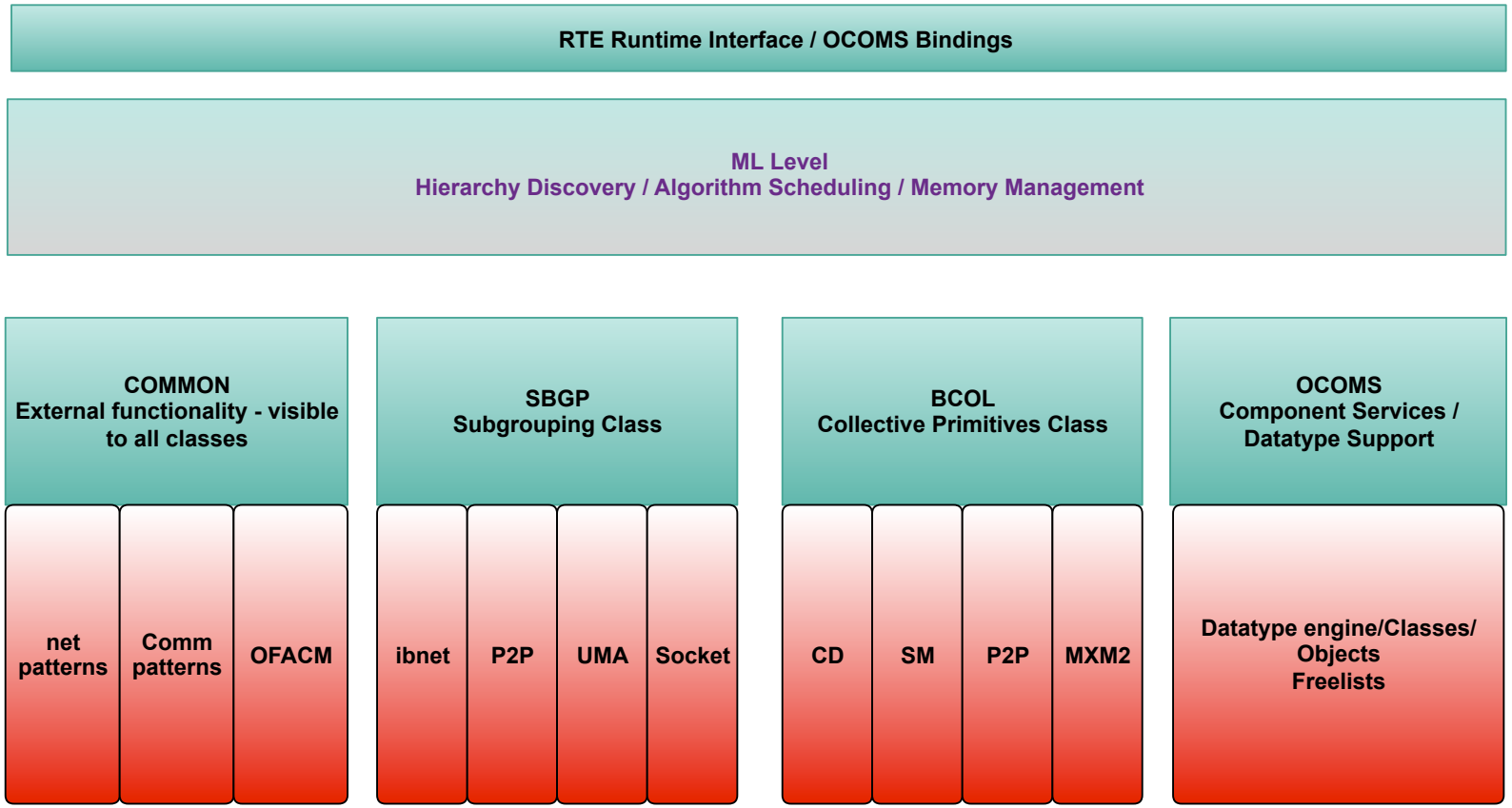
Fabric Collective Acceleration (FCA)

Collective Communication

FCA - Highlights

- Blocking and non-blocking collective routines
- Modular component architecture
- Scalable runtime interface (RTE)
  - Successfully integrated into Open MPI & MPICH
- Host level hierarchy awareness
  - Socket groups
  - UMA groups
- Exposes Mellanox and InfiniBand specific capabilities
  - CORE-*Direct*
  - MXM 2.0
    - UD, RC, DC
  - Hardware multicast

FCA 3.0 Software Architecture



FCA 3.0 Algorithms

- Supported collectives in 3.0
  - MPI_Allgather
  - MPI_Allreduce
  - MPI_Barrier
  - MPI_Bcast
  - MPI_Iallgather
  - MPI_Iallreduce
  - MPI_Ibarrier
  - MPI_Ibcast
- To be supported in 3.5
  - MPI_Allgather(v)
  - MPI_Alltoall(v)
  - MPI_Scatter
  - MPI_Gather
  - MPI_IAllgather(v)
  - MPI_IAlltoall(v)
  - MPI_IScatter
  - MPI_IGather
- All other collectives fallback to runtime (MPICH) provided algorithms

MPICH & HPC-X

- Integrated Mellanox MXM and FCA into MPICH
  - MXM as a new network module (netmod)
  - FCA as a network independent collective accelerator
  - Available in MPICH-3.2a2 as beta form.

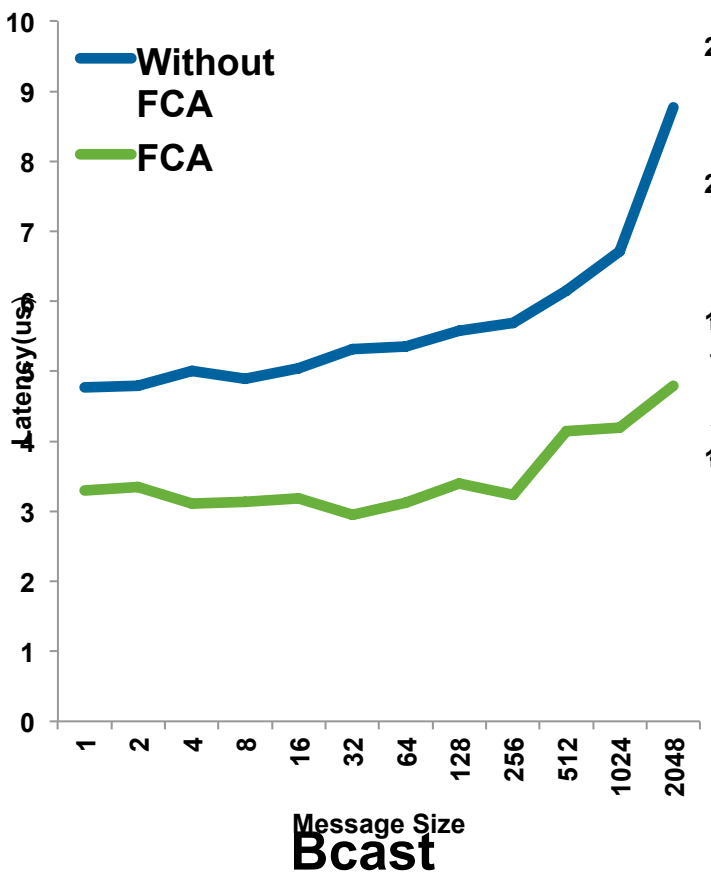
Performance Data

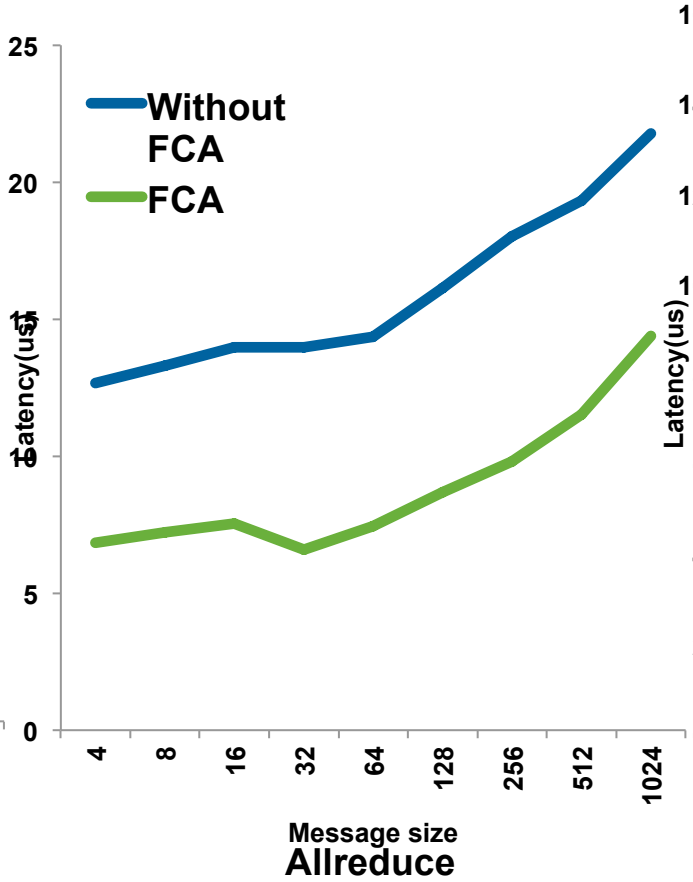
MPICH/MXM Performance

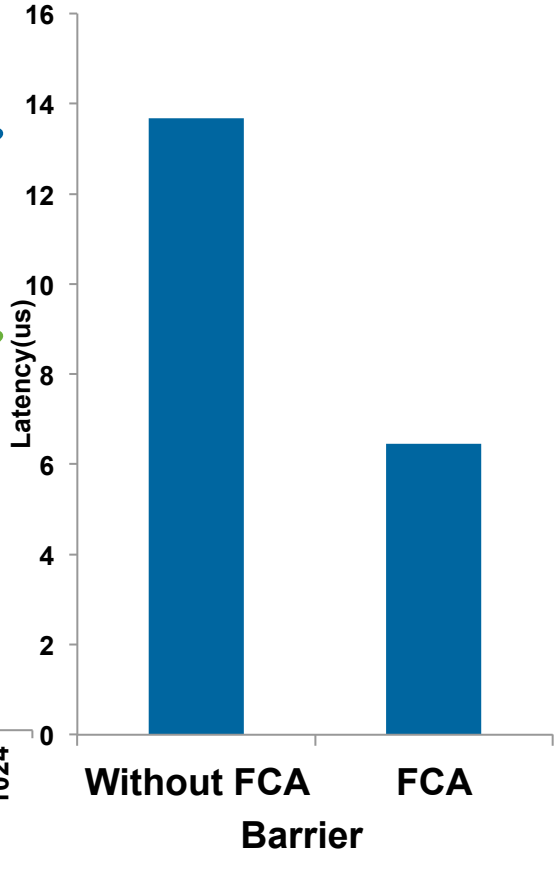
MPI Point-to-point Performance						
Transport	**Verbs UD***	**Verbs RC***	**MPICH-MXM UD**	**MPICH MXM RC**	**MVAPICH**	**OpenMPI**
send/recv latency 2b (us)	0.79	0.76	1.26	1.17	1.28	1.16
bandwidth 4k (Mbyte/sec)	6164	6178	5188	5528	5479	5203
bandwidth 1M (Mbyte/sec)	6168	6200	6398	6422	6301	6460
packet rate 1b (million/sec)	8.08	9.35	3.34	3.52	2.91	3.34

*Setup : Connect-IB FDR, Intel Ivy Bridge CPU E5-2680 v2 @ 2.80GHz
MXM/MICH : MPICH trunk, hpcx-v1.2.0
MVAPICH : mvapich2-2.1a
OpenMPI : ompi-v1.8, hpcx-v1.2.0, openib BTL
Verbs : Ib_send_lat, ib_send_bw, ib_write_bw

MPICH/FCA Performance







- Setup : Connect-IB FDR with switch, Intel Ivy Bridge CPU E5-2697 v2 @ 2.70GHz **, 384 ranks (16 nodes, 24 ppn**)
- Without FCA : MPICH trunk with MXM netmod
- FCA : MPICH trunk with MXM netmod and FCA
- Benchmark : IMB

Thank You

Mellanox TECHNOLOGIES

Connect. Accelerate. Outperform.™

MS-MPI

SC|14 Update

Microsoft is invested in HPC

- Microsoft is, and will continue, investing in HPC
  - All on-premises (HPC Pack evolution, SMB Direct)
  - Hybrid (burst via HPC Pack)
  - All in the cloud (Head Node in IaaS VM)
- Azure HPC Virtual Machines

	CPU	Memory	Network
A8	Intel® Xeon® E5-2670 8 cores @ 2.6 GHz	DDR3-1600 MHz 56 GB	Frontend = 10 Gbps Ethernet Backend = InfiniBand QDR, RDMA-capable
A9	Intel® Xeon® E5-2670 16 cores @ 2.6 GHz	DDR3-1600 MHz 112 GB	Frontend = 10 Gbps Ethernet Backend = InfiniBand QDR, RDMA-capable

- We aim to provide the best commercial platform for compute-intensive parallel workloads

HPC VMs availability

Seven Regions (red):

- West US
- South Central US
- North Central US
- East US
- West Europe
- North Europe
- Japan East

Committed to supporting the ecosystem

- MS-MPI is supported in all Windows environments:
  - Windows Vista
  - Windows 7
  - Windows 8
  - Windows 8.1
  - Windows Server (2008, 2008 R2, 2012, 2012R2)
  - Azure PaaS
- Free to use
- Free to redistribute with 3rd party applications

Customer driven

- MS-MPI v5:
  - http://go.microsoft.com/fwlink/?LinkID=389556
  - MPI-3 Shared Memory Windows
  - Improved Debugging support
    - Partner with Intel MPI
  - Configurable Network Direct memory footprint
  - Improved Fortran support
  - Enable static code analysis for MPI applications
- MS-MPI v6:
  - Targeting Q2 2015
  - MPI-3 Features
    - Non-blocking Collectives
    - Matched Probe
    - MPI Tools interface
    - Large Count
  - Performance Enhancements

We won’t bite!

- Let us know what you need
  - Features
  - Bugs
  - Help
- MS-MPI team eager to connect with Windows HPC community
  - Home page (http://msdn.microsoft.com/en-us/library/bb524831.aspx)
  - Contact askmpi@microsoft.com
  - Blog (http://blogs.technet.com/b/windowshpc/)
  - Web Forum http://social.microsoft.com/Forums/en-US/home?forum=windowshpcmpi

MPICH on K, PostT2K, and PostK

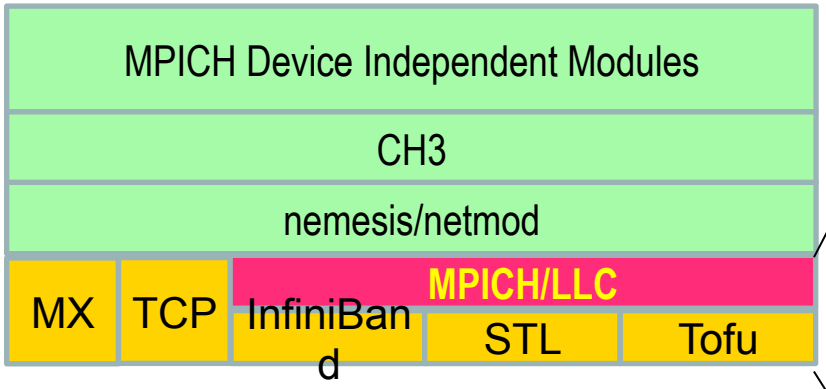
Yutaka Ishikawa, Masamichi Takagi, Norio Yamaguchi, Masayuki Hatanaka

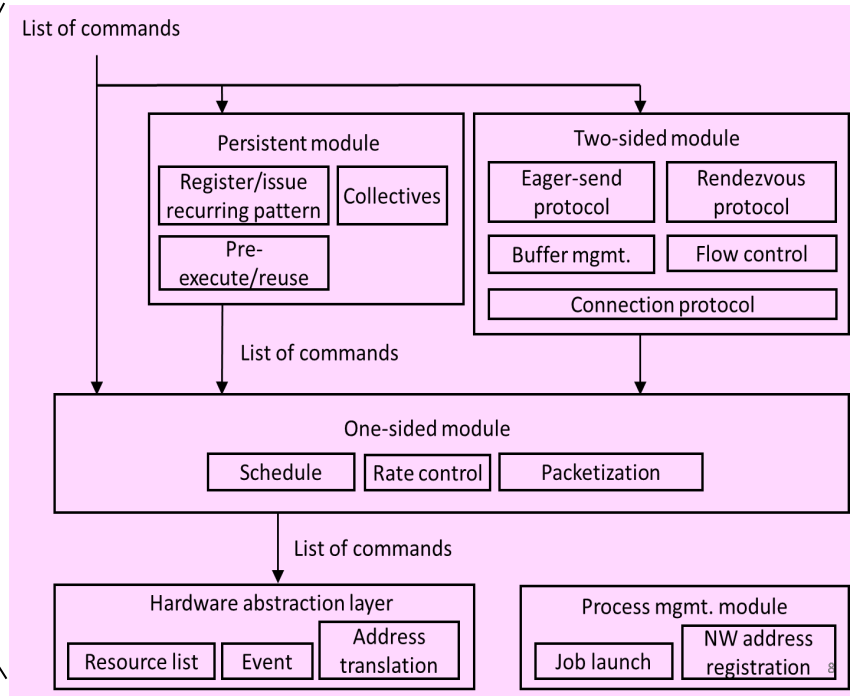
RIKEN AICS

MPICH BoF at SC14

2014/11/18

MPICH/LLC



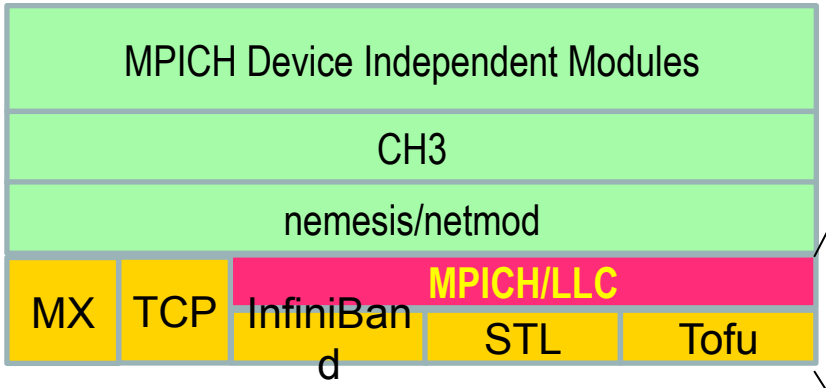


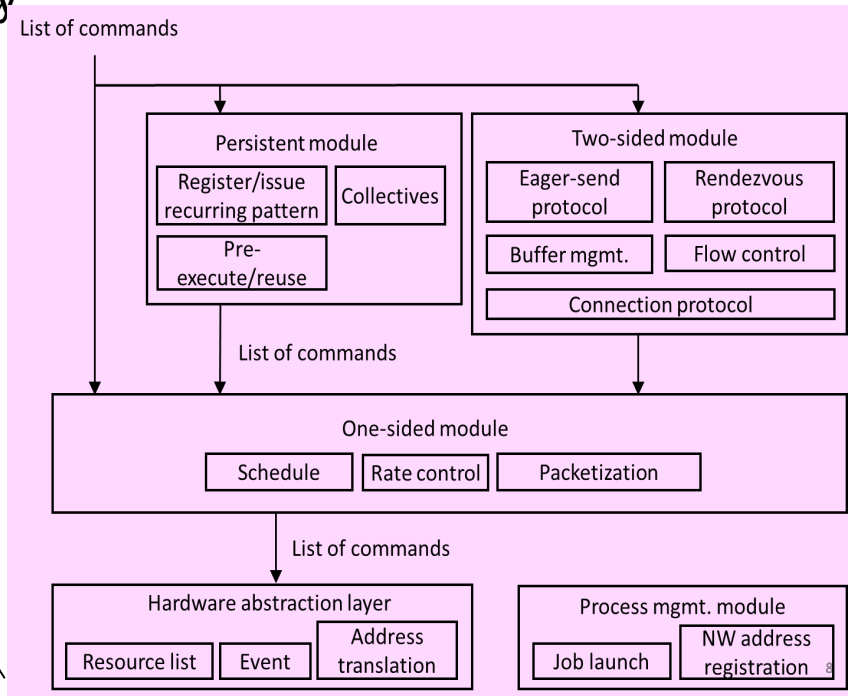
- ✓ MPICH for KNL developed by Masamichi Takagi and Norio Yamaguchi
- ✓ MPICH for K developed by Masayuki Hatanaka

- MPICH/LLC/Infiniband
  - PostT2K, 30+PF manycore-based cluster operated in 2016
    - University of Tsukuba and University of Tokyo
- MPICH/LLC/Tofu
  - K and PostK, Japanese FLAGSHIP supercomputers
    - AICS RIKEN

Why is an additional layer designed in MPICH ?

- Management of communication protocol, e.g., eager/ rendezvous, and buffering is almost the same implementation in NICs with RDMA capabilities, e.g., InfiniBand and Tofu, and maybe STL.
- If the additional layer were efficiently implemented for those NICs, it would be used in wide-range of parallel computers and contribute to maintainability/portability





- ✓ MPICH for KNL developed by Masamichi Takagi and Norio Yamaguchi
- ✓ MPICH for K developed by Masayuki Hatanaka

Current Status

- MPICH/netmod/InfiniBand
  - It is in the MPICH working repository
- MPICH/LLC/Tofu 0.X
  - Will be available in Q1 of 2015
- MPICH/LLC/Infiniband 1.0
  - Specification will be finalized in Q1 of 2015
  - Implementation will be done in Q3 of 2015

Panel Discussion/Questions

See you at SC '15 in Austin!

Thanks For Coming