Open MPI State of the Union Community Meeting SC ‘15

November 18, 2015

Jeff Squyres

CISCO

George Bosilca

ICL UT

Nathan Hjelm

Los Alamos NATIONAL LABORATORY EST. 1943

First, two quick
public service announcements

MPI Forum BOF

Wednesday, 3:30pm
Room 15

<u>Come hear about:</u>
MPI-3.1
The road to MPI-4 (ongoing activities)

www.EuroMPI2016.ed.ac.uk

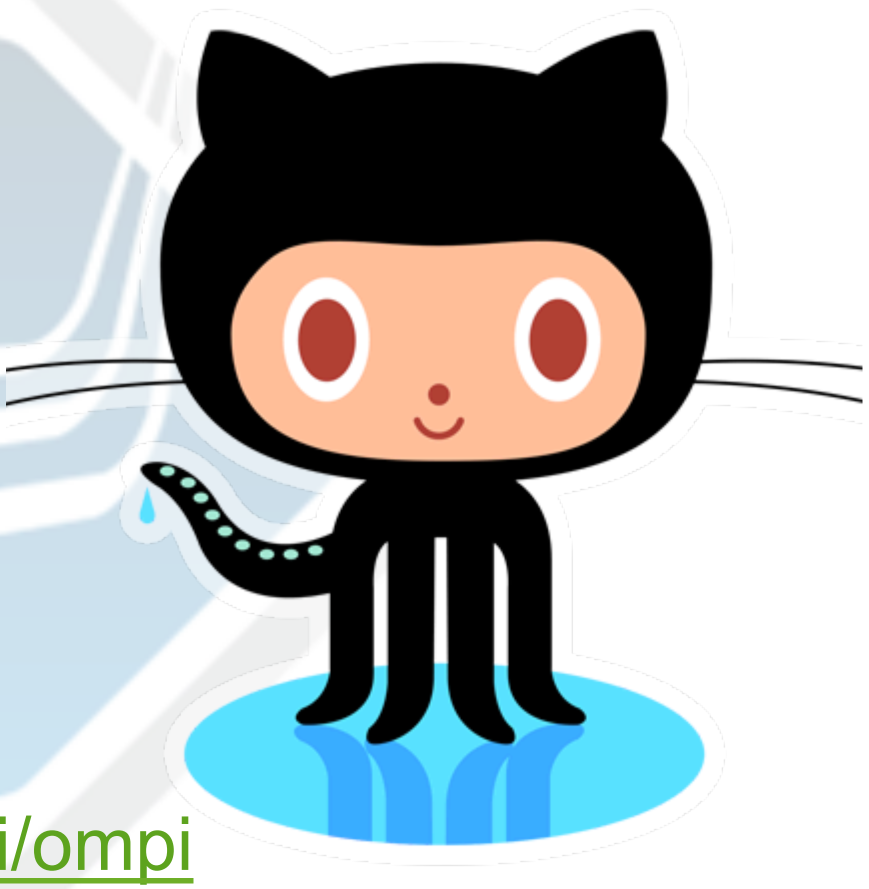
- Call for papers open by end of November 2015
- Full paper submission deadline: 1st May 2016

- Associated events: tutorials, workshops, training
- Focusing on: benchmarks, tools, applications, parallel I/O, fault tolerance, hybrid MPI+X, and alternatives to MPI and reasons for not using MPI

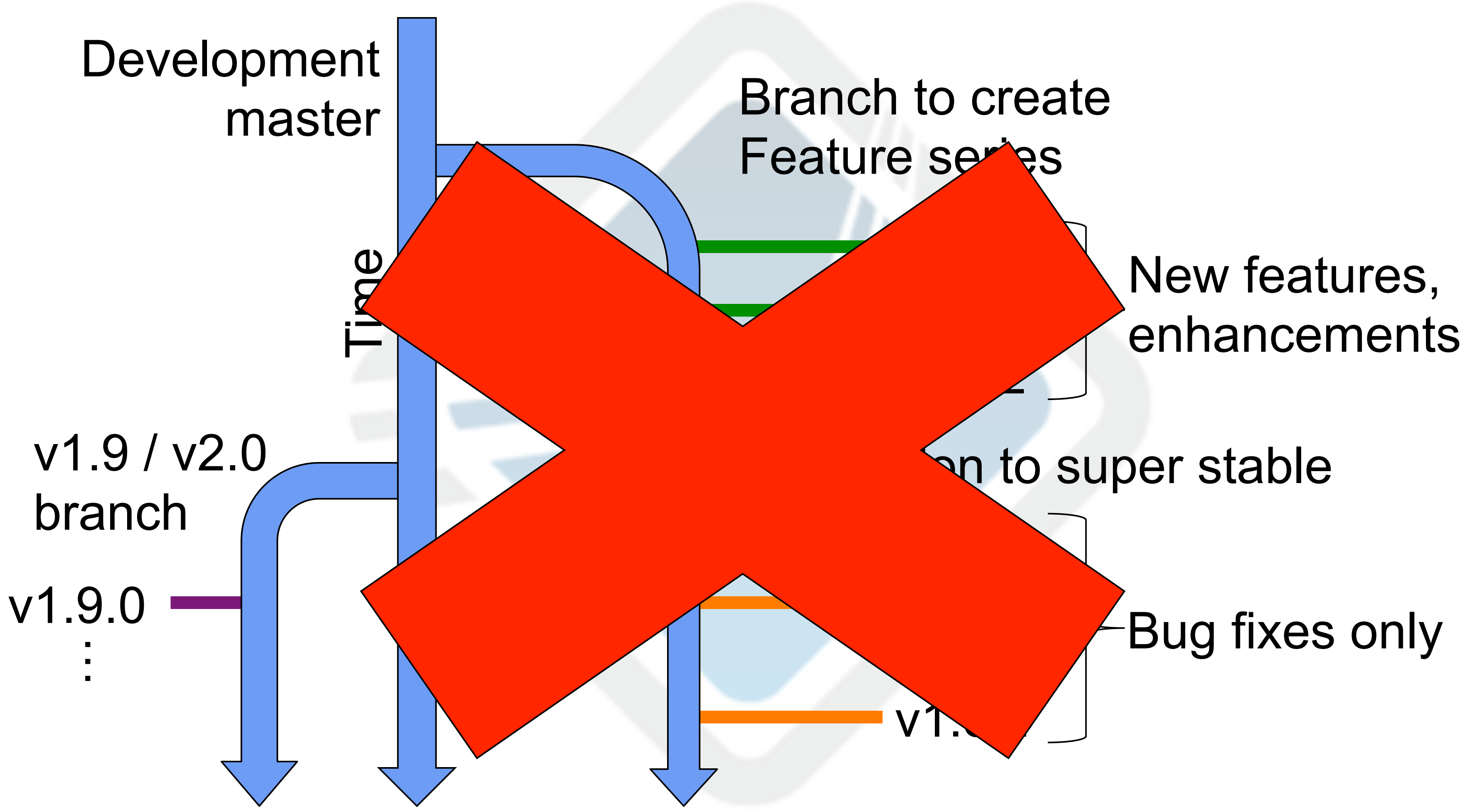
Reminder: Open MPI now hosted on GitHub

- In late 2014, Open MPI code and bug tracking moved to GitHub
- Please file at GitHub:
  - Issues (bugs, feature requests, etc.)
  - Pull requests (code!)

https://github.com/open-mpi/ompi



Open MPI versioning



Open MPI versioning

No more
“odd / even” versioning!

New version numbering scheme

- Open MPI will (continue to) use a "A.B.C" version number triple
- Each number now has a specific meaning:

A	This number changes when backwards compatibility breaks
B	This number changes when new features are added
C	This number changes for all other releases

Definition

- Open MPI vY is *backwards compatible* with Open MPI vX (where Y>X) if:
  - Users can compile a correct MPI / OSHMEM program with vX
  - Run it with the same CLI options and MCA parameters using vX or vY
  - The job executes correctly

What does that encompass?

- “Backwards compatibility” covers several areas:
  - Binary compatibility, specifically the MPI / OSHMEM API ABI
  - MPI / OSHMEM run time system
  - `mpirun` / `oshrun` CLI options
  - MCA parameter names / values / meanings

What does that *not* encompass?

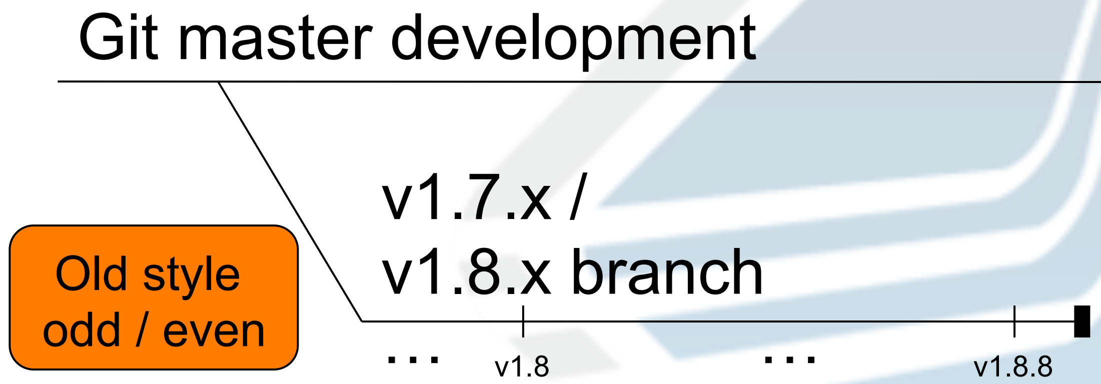
- Open MPI only supports running exactly the same version of the runtime and MPI / OSHMEM libraries in a single job
  - If you mix-n-match vX and vY in a single job…

v1.8.x Roadmap

v1.8.x / v1.10.x release manager

- Ralph Castain, Intel

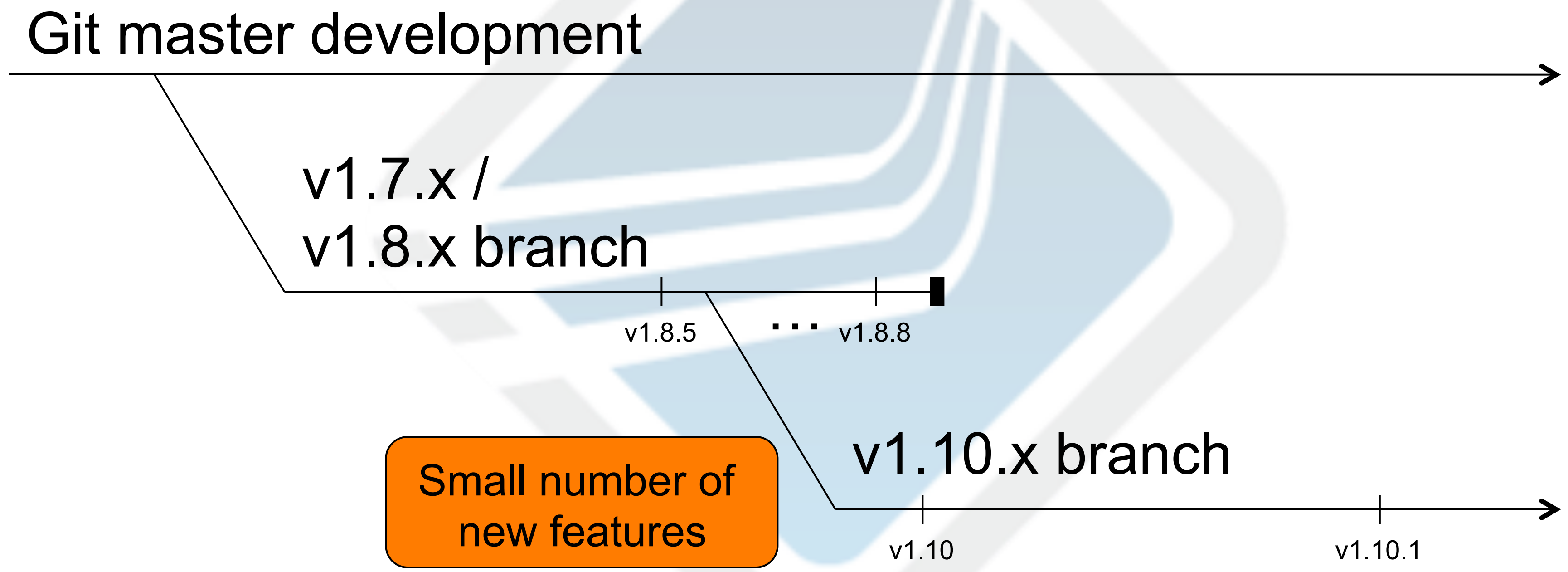
v1.8 roadmap



No further v1.8.x releases planned

v1.10.x Roadmap

v1.10.x roadmap



v1.10 series

- Small number of new features
  - libfabric support for usNIC and PSM
  - Mellanox Yalla PML
  - Intel PSM2 for OmniPath
- Built on top of stable v1.8 series
- "Mostly" the new numbering scheme
  - Example: v1.10.1 could have been called v1.11 (new features, but still ABI compatible)

User question

Does OmniPath work with Open MPI v1.10.x?

Yes, OmniPath should work out-of-the-box with Open MPI v1.10.1 via the PSM2 MTL.

v1.10 series

- Current status:
  - v1.10.0: released August 25, 2015
  - v1.10.1: released October 4, 2015
    - Bug fixes compared to v1.10.0
    - Conformance: MPI-3.1 without non-blocking I/O collectives
  - v1.10.2: <u>estimated</u> January 2016

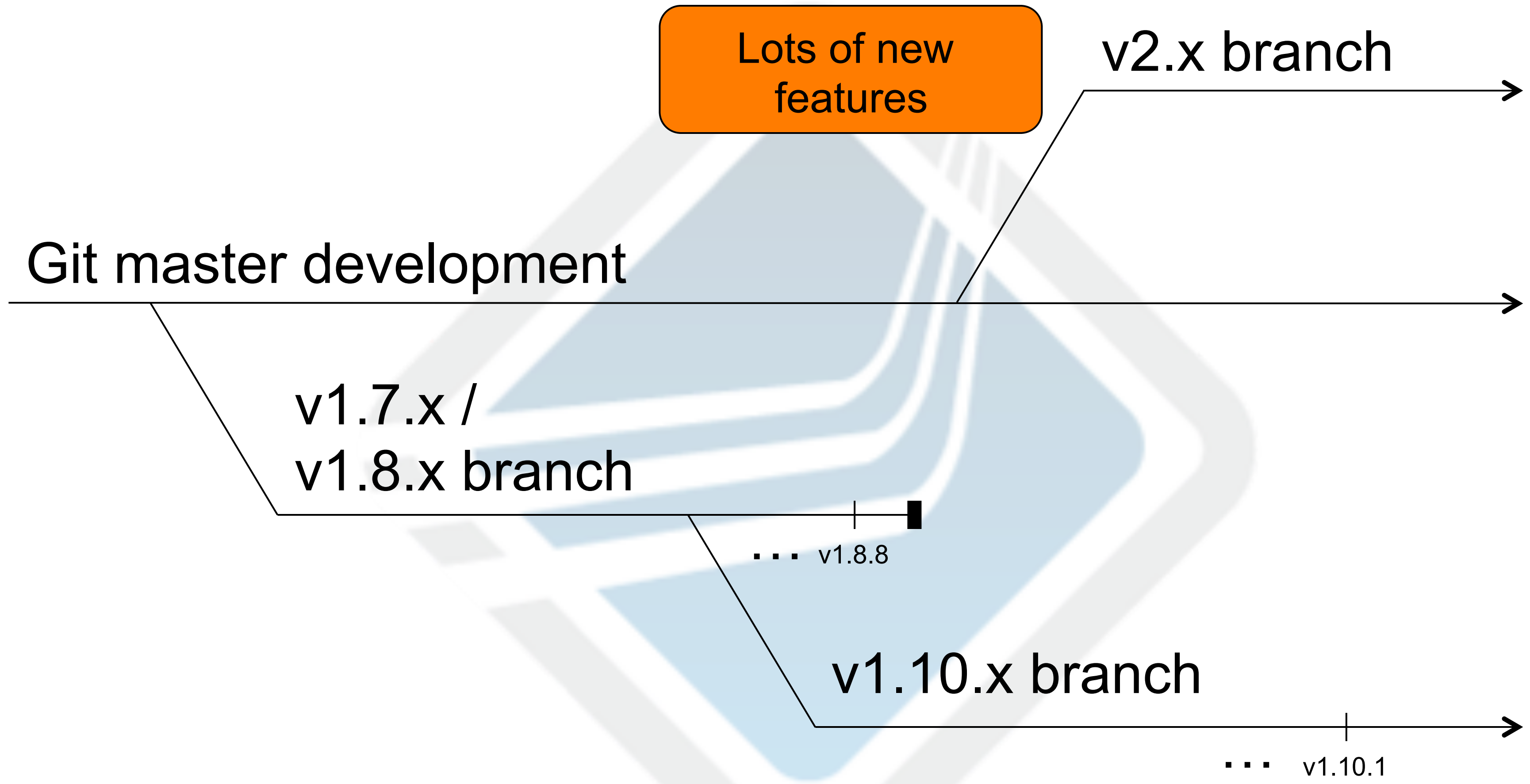
- There will likely be future bug-fix releases as needed

v2.x Roadmap

v2.x release managers

- Howard Pritchard, Los Alamos National Lab
- Jeff Squyres, Cisco Systems, Inc.

v2.x roadmap


Lots of new features
v2.x branch
Git master development
v1.7.x / v1.8.x branch
. . . v1.8.8
v1.10.x branch
. . . v1.10.1

v2.x timeline

- Branched in June, 2015
- v2.0.0 has taken longer than expected
  - v1.10.x releases have consumed resources
  - *Many* new features in v2.0.0

- Estimate v2.0.0 release Q1 2016
  - Currently focusing on stabilizing new features
  - Will be MPI-3.1 compliant

IBM Platform MPI

- IBM Platform MPI to be based on Open MPI v2.x
  - Many proprietary features to be committed back to Open MPI
  - First release planned Q3, 2016
- Integration with IBM/Platform software
  - Customer support available
  - See the IBM booth for more details

Miscellaneous v2.x features

- Compliance level:
  - MPI-3.0 + errata
  - MPI-3.1
- Performance and scalability improvements
  - Lots of internal plumbing enhancements
- Improved Cray XE/XC support
- Unified Communication X (UCX) support
- Process Management Interface – Exascale (PMIx)

PMIx

- Goal: Work with the HPC community to define and implement new APIs that support evolving programming model requirements for application/resource manager interactions.
- Support the *Instant On* initiative
  - For rapid startup of applications at exascale+
- Standalone library (non-copy-left licensed)

BoF: Thurs @ 12:15-1:15pm, Room 15

Usability issues

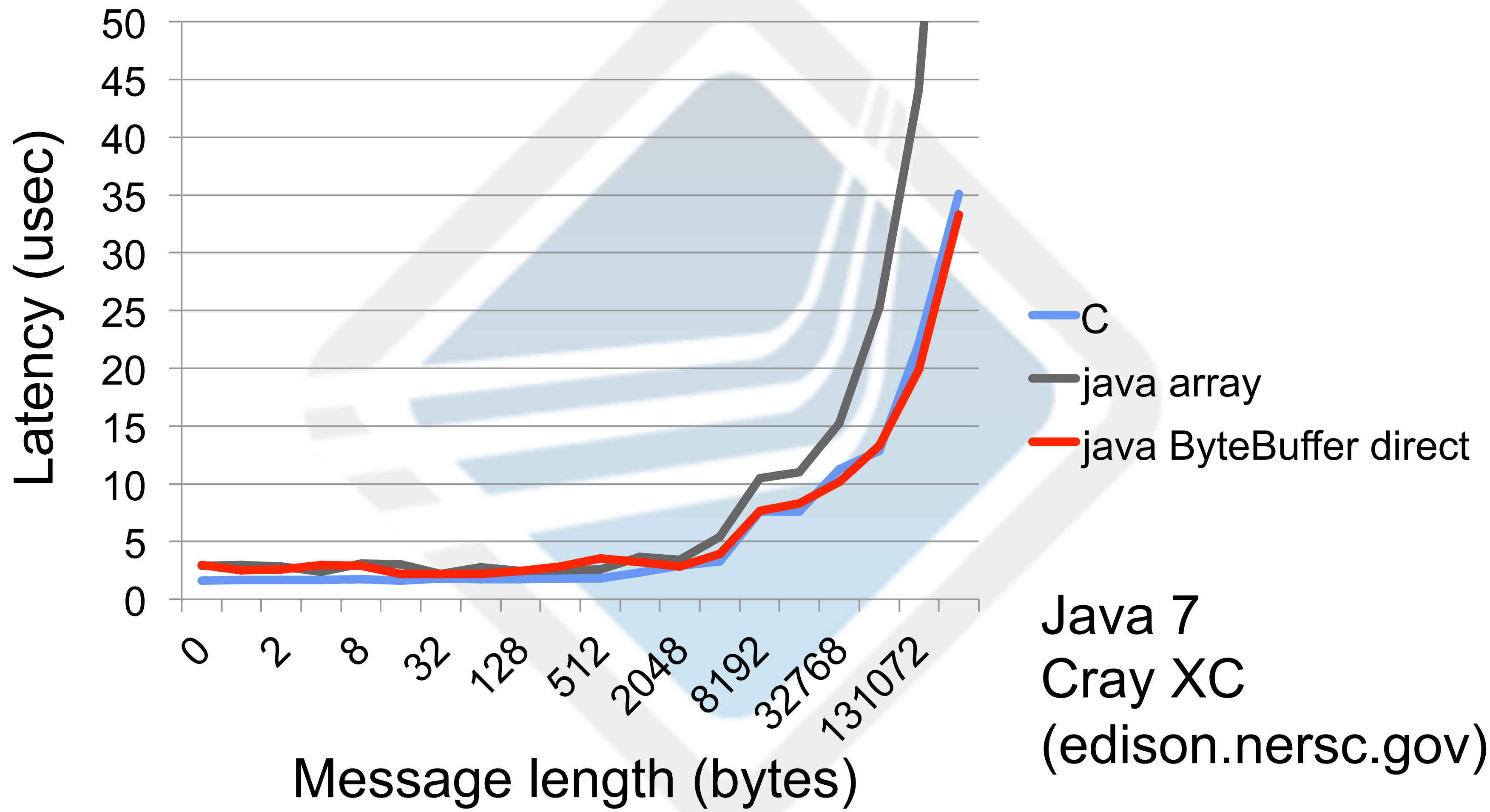
- Open MPI has some cryptic plugin names
  - OB1, R2, Vader, Yoda, Ikrit, …
- New libraries provide multiple ways to get to the same underlying network
  - Libfabric
  - UCX

  → Need a simple, clear interface for users
- This is an ongoing discussion in the development community…

Java bindings updates

- Primarily contributed by Oscar Vega-Gispert
  - Community (LANL) updating the bindings/fixing bugs
- Now includes (almost) all MPI 3.1 functionality
  - *MPI_WIN_shared_query*, etc. not implemented because they don't make sense in Java (no explicit virtual address concept)
  - *MPI_Neighbor_Alltoall(w/v)* will be coming after v2.0.0
- Various bug fixes, including:
  - Better interaction with JVM GC
  - Fixes for PSM/JVM signal handler conflicts (PSM fix)
- Java MPI tests / examples now available
  - https://github.com/open-mpi/ompi-java-test

Java performance – OSU latency



Java 7
Cray XC
(edison.nersc.gov)

2.x goodies

George Bosilca

Features dropped from v2.x

- Myrinet / MX support
- Cray XT support
- VampirTrace support
- Checkpoint / restart support
- Legacy collective module: ml

User question

How do I performance profile my code?

There’s a lot of good PMPI-based tools available.

VampirTrace has been removed from Open MPI, but ScoreP is it’s direct replacement (not yet bundled in Open MPI).

https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/ScoreP

Com

It’s complicated!

(only listing those that are in release branches)

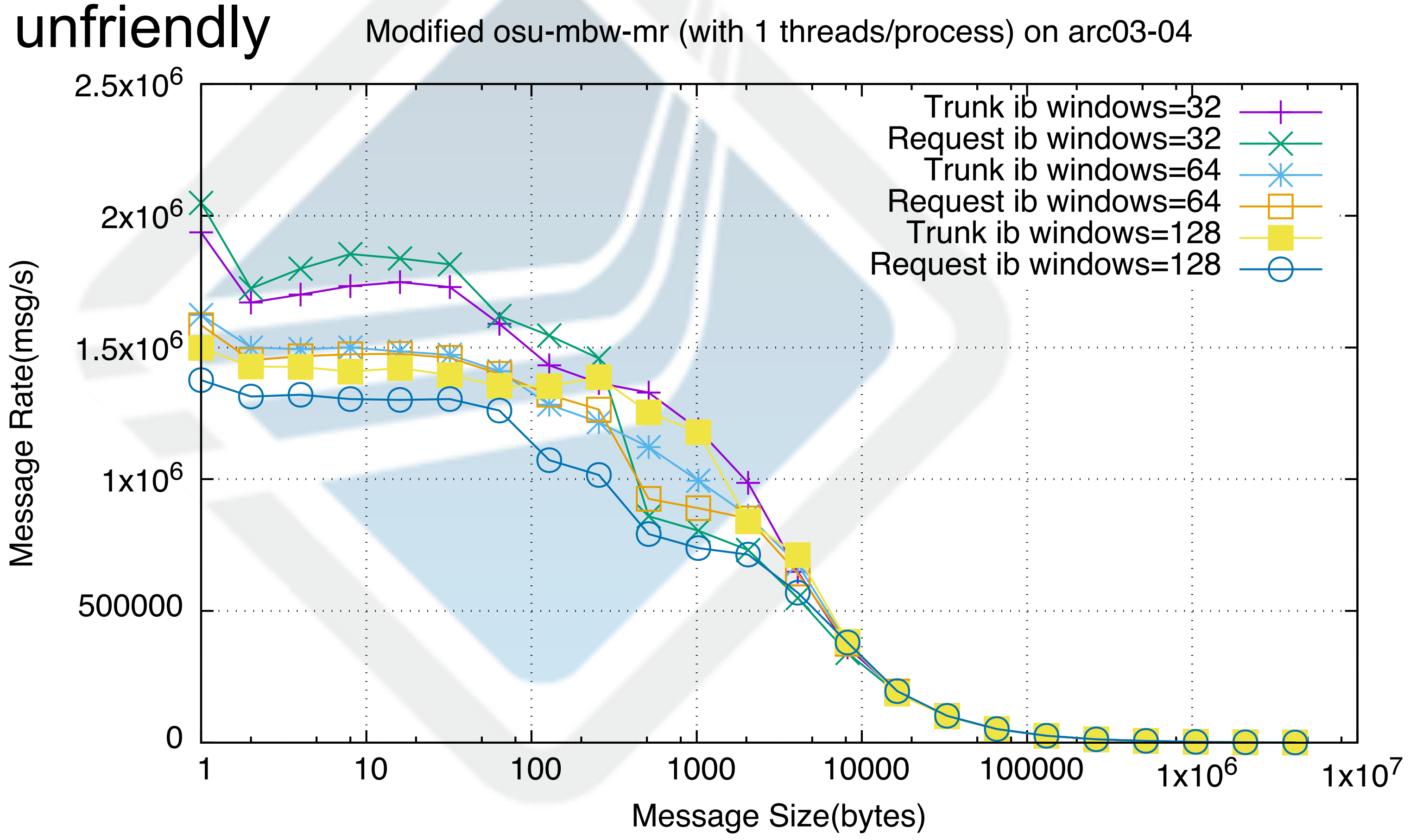
	Name	Owner	Status	Th.
BTL	OpenIB	Chelsio	maintenance	🟩
	Portals4	SNL	active?	🟥
	Scif	LANL	maintenance	▨
	self	UTK	active	🟩
	sm	UTK	active	🟩
	smcuda	NVIDIA	active	🟩
	tcp	UTK	active	🟩
	uGNI	LANL	active	🟩
	usnic	CISCO	active	🟩
	vader	LANL	active	🟩
PML	Yalla	Mellanox	active	▨
	UCX	Mellanox/UTK	active	▨
MTL	MXM	Mellanox	active	▨
	OFI	Intel	active	▨
	Portals4	SNL	active	▨
	PSM	Intel	active	▨
	PSM2	Intel	active	▨

MPI_Init / MPI_Finalize extension(s)

- Also called the #~~1007~~ fiasco
- Thanks to the MPI standard (3.1) we know that some functions must be thread safe even for libraries that do not support thread safety (Section 12.4):
  - MPI_INITIALIZED, MPI_FINALIZED, MPI_QUERY_THREAD, MPI_IS_THREAD_MAIN, MPI_GET_VERSION and MPI_GET_LIBRARY_VERSION
- Clearly there is a need for a better (more precise) definition. Meanwhile:
- If MPI_INITIALIZED is invoked and MPI is only partially initialized, wait until MPI is fully initialized before returning.
- If MPI_FINALIZED is invoked and MPI is only partially finalized, wait until MPI is fully finalized before returning.

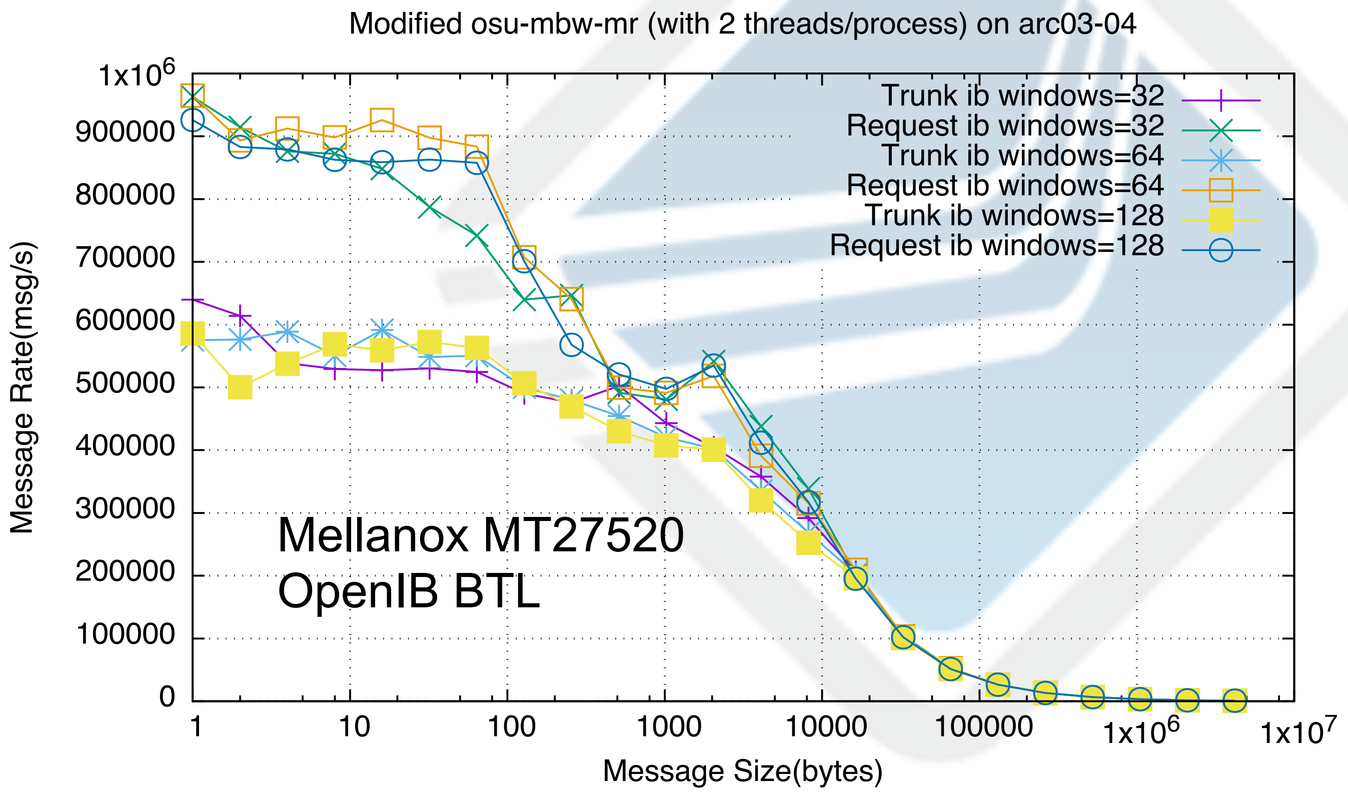
MPI_[Wait|Test]* (#176)

- We had supra-linear complexity
  - Due to error case scenarios
  - Thread unfriendly



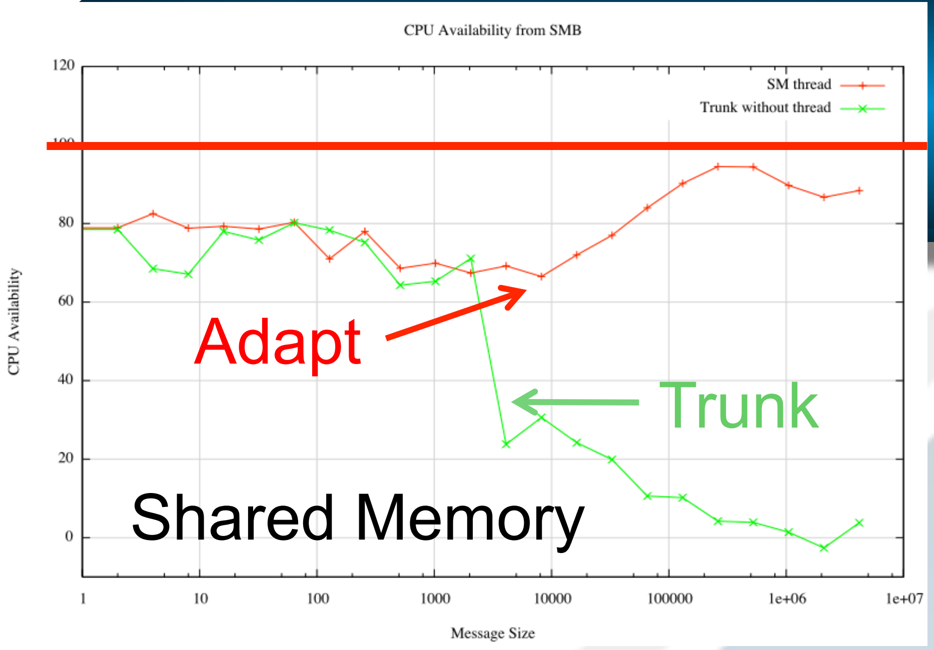
Threading

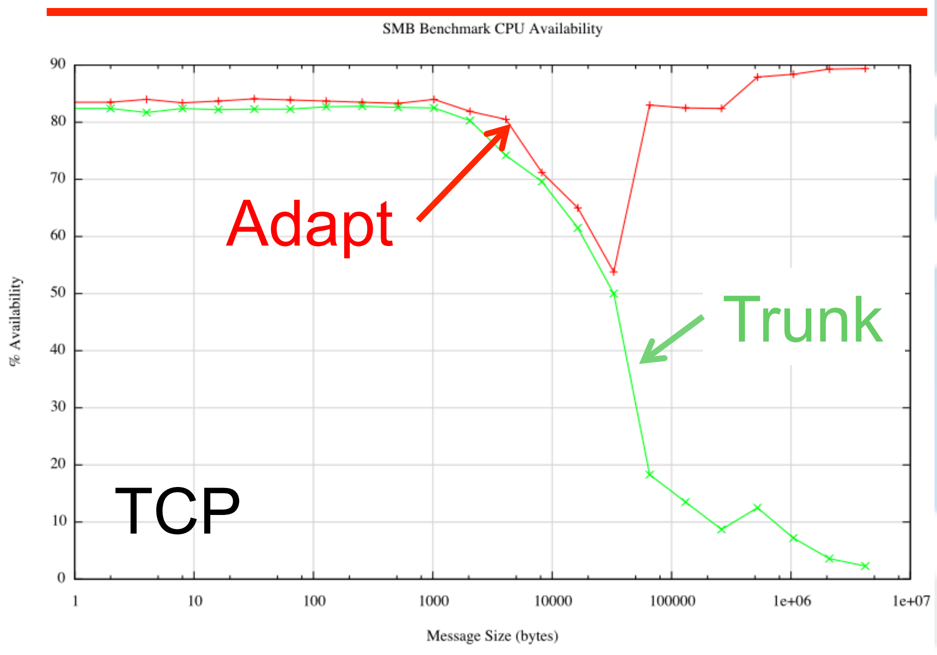
- Audit of the communication support
  - Fine grain locks for request management
  - Atomic operations for everything else



From now until the release the focus is on performance

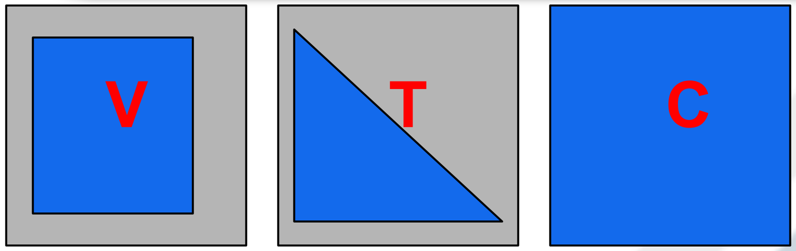
Async progress

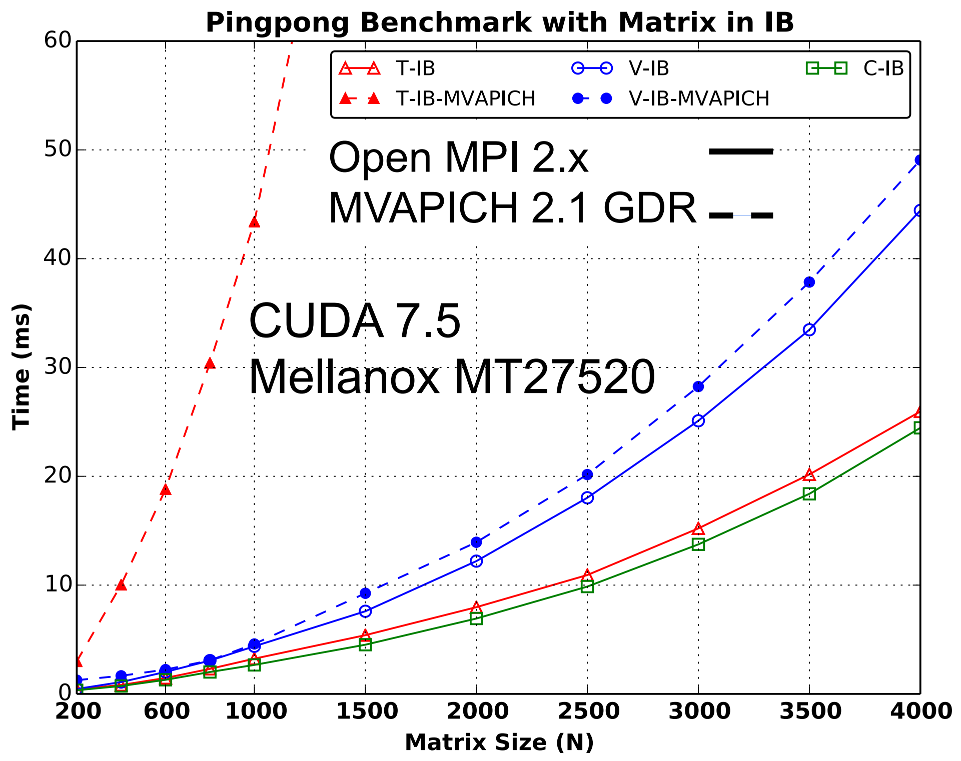




- Helper thread(s) to make progress
  - Support for more communication infrastructures to come

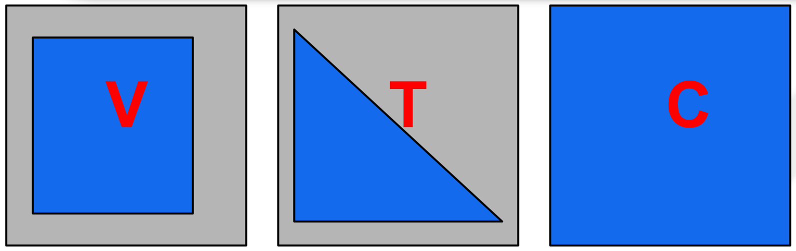
CUDA Support

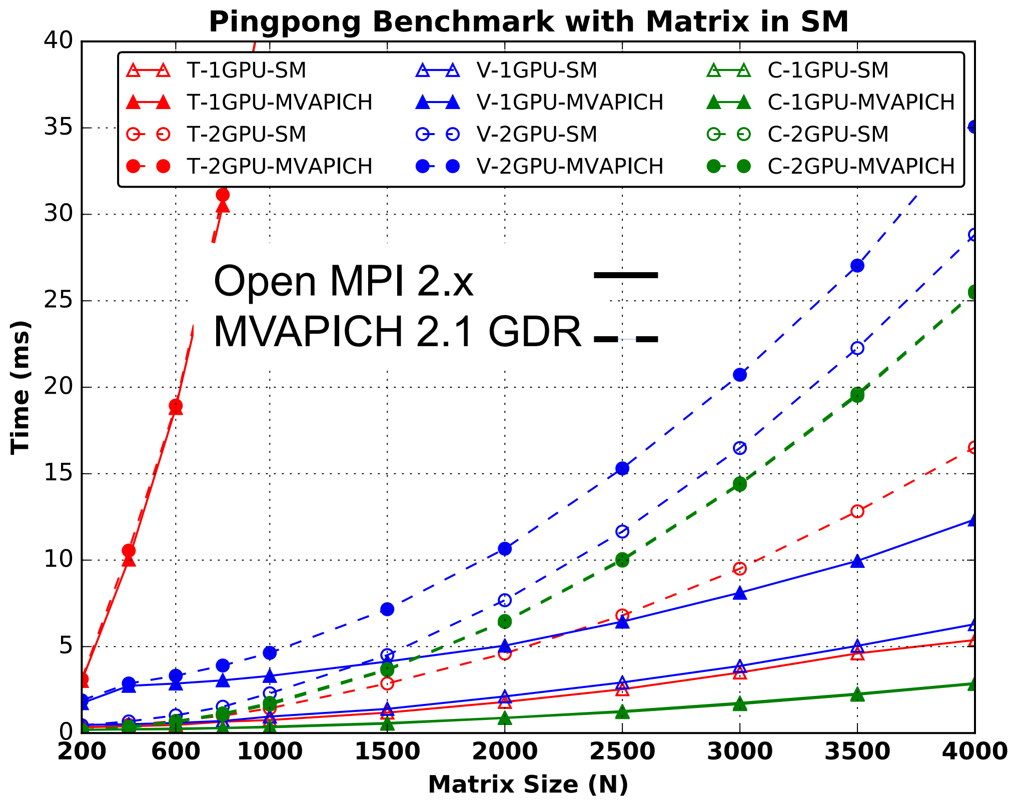




- Multi-level coordination protocol based on the location of the source and destination memory
- Delocalize part of the datatype engine into the GPU
  - Major part still driven by the CPU
  - Provide a different datatype representation (avoid branching in the code)

CUDA Support





- Support for GPUDirect
- Right now integrated with OpenIB and shared memory
  - BTL independent support in the design stage

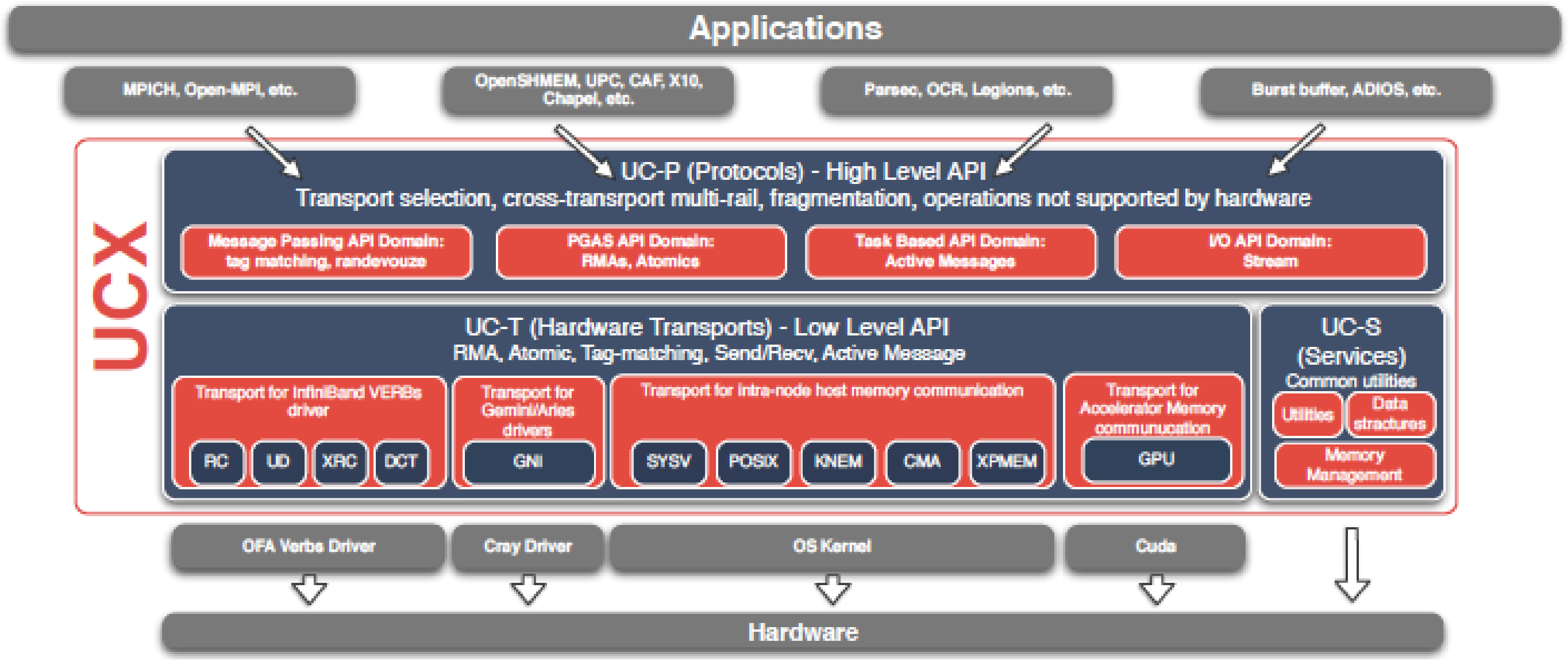
Monitoring PML (#724)

- Intercept the component interface
  - through careful manipulations of the init/enable/fini interfaces
  - Expose more [low-level] details about communication patterns without PMPI
    - Bytes for point-to-point, groups for collectives, control messages, collective communications patterns, no one-sided
  - Controlled by MPI_T to generate an extended communication map
    - Topology components (TreeMatch) reorder the processes

I/O

- Supports 2 modules:
  - ROMIO 3.1.4 (~~#518~~), 3.2b (~~#504~~)
  - OMPIO (UH) – default in 2.x
- Highly modular architecture for parallel I/O
- Supports Multiple Collective I/O algorithms
  - File view based automatic selection of collective I/O component
- Automatic adjustments of no. of aggregators depending on
  - Application characteristics (e.g. data volume, access pattern)
  - File system characteristics (e.g. stripe size, stripe depth)
- Asynchronous individual and collective I/O operations
  - Integrated with the progress engine of Open MPI

UCX integration

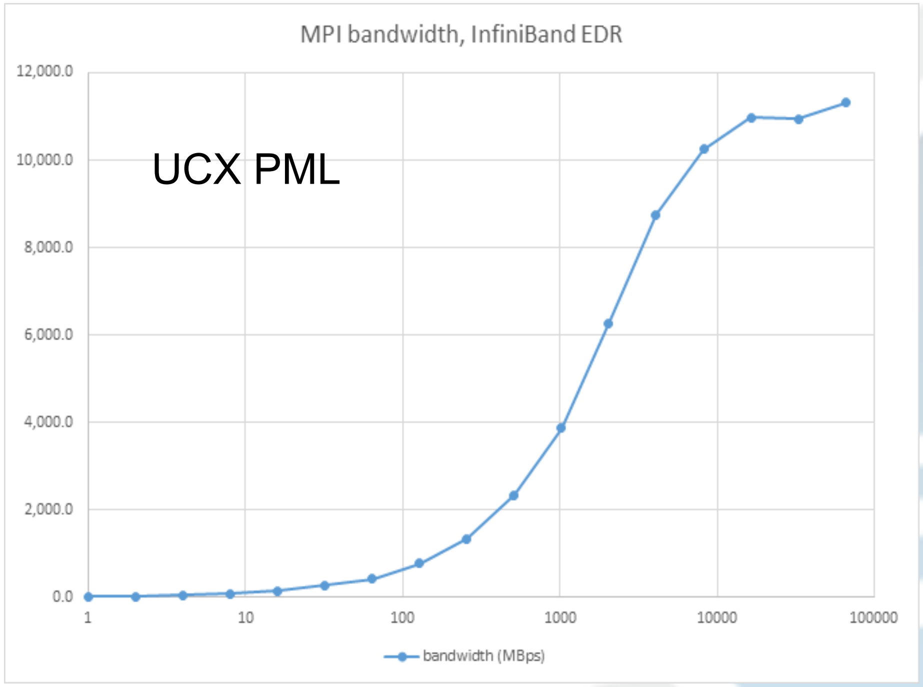

Applications
MPICH, Open-MPI, etc.
OpenSHMEM, UPC, CAF, X10, Chapel, etc.
Parsec, OCR, Legions, etc.
Burst buffer, ADIOS, etc.
UCX
UC-P (Protocols) - High Level API
Transport selection, cross-transrport multi-rail, fragmentation, operations not supported by hardware
Message Passing API Domain: tag matching, randevouze
PGAS API Domain: RMAs, Atomics
Task Based API Domain: Active Messages
I/O API Domain: Stream
UC-T (Hardware Transports) - Low Level API
RMA, Atomic, Tag-matching, Send/Recv, Active Message
Transport for InfiniBand VERBs driver
RC
UD
XRC
DCT
Transport for Gemini/Aries drivers
GNI
Transport for intra-node host memory communication
SYSV
POSIX
KNEM
CMA
XPMEM
Transport for Accelerator Memory communucation
GPU
UC-S (Services)
Common utilities
Utilities
Data structures
Memory Management
OFA Verbs Driver
Cray Driver
OS Kernel
Cuda
Hardware

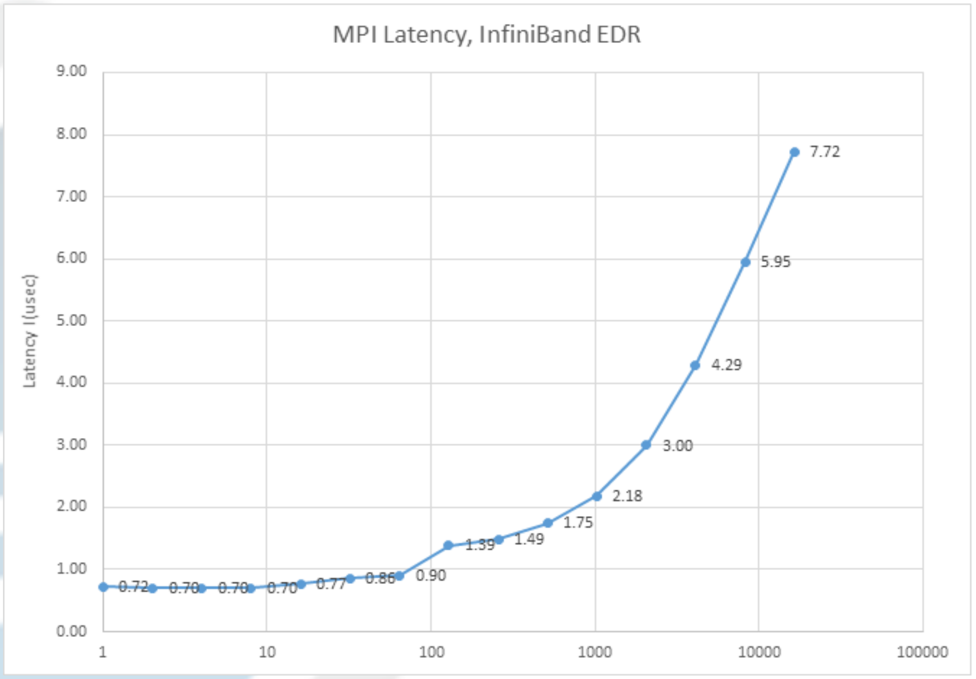
- UCX is the next generation open-source production-grade communication middleware.
- Collaboration between industry, laboratories, and academia.
- Exposes near-bare-metal performance by reducing CPU overhead.
- Provides both a thin low-level API, and a general purpose high-level API.

UCX Integration

ConnectX-4 EDR

Back-to-back, no switch





Other minor improvements

- Dynamic process management (~~#989~~)
  - Gracefully disabled if the RTE or the underlying communication does not supports this feature
- C99 component support (~~#541~~)
- Dismantle the tuned collective support
  - All algorithms are now available to all collective modules
  - Tuned became a tiny decision layer
  - Driver specific configuration become now possible
- Extend MPI_COMM_SPLIT_TYPE to include all HWLOC types (CU, HOST, BOARD, SOCKET, LxCACHE…) (~~#320~~)
- Did I mention Fortran yet?

Memory Scaling and RMA Update

Nathan Hjelm
Los Alamos National Laboratory

v1.10.x memory scaling

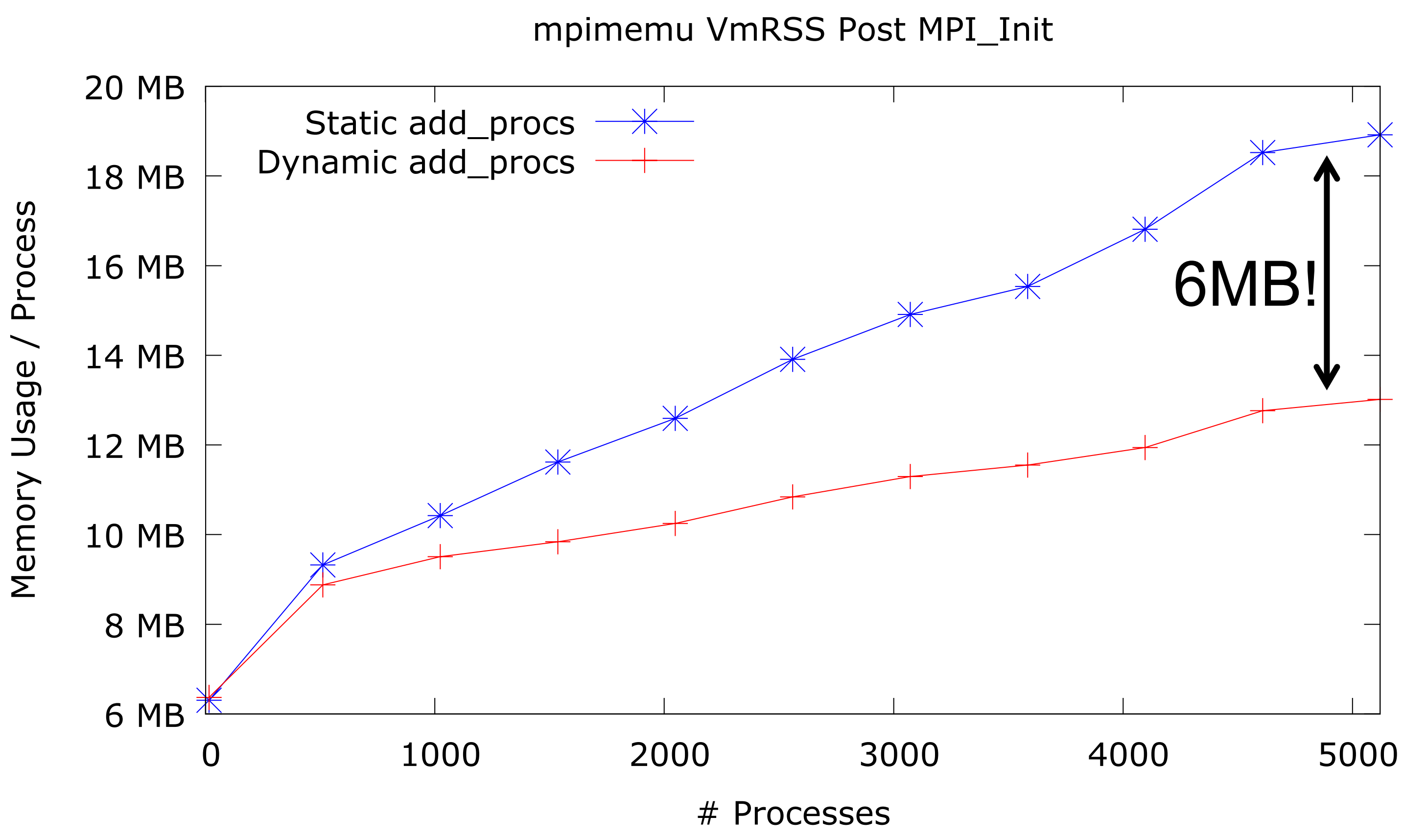
- Process structure and network endpoint allocated for all processes in MPI_COMM_WORLD
- Structures allocated even if when there is no communication with a peer

v2.x memory scaling

- Process structure and network endpoint allocated on first communication
- Small delay in 1st communication, but…
  - ***<u>Big</u>*** reduction in memory footprint
  - Configurable cutoff for new behavior
  - MCA variable: `mpi_add_procs_cutoff`
    - Default:1024 processes

Memory scaling

- mpimemu VmRSS on 320 nodes of a Cray XE-6. 16 ppn aprun
- Reduction of 6MB/process @ 5120 processes!



v1.10 RMA

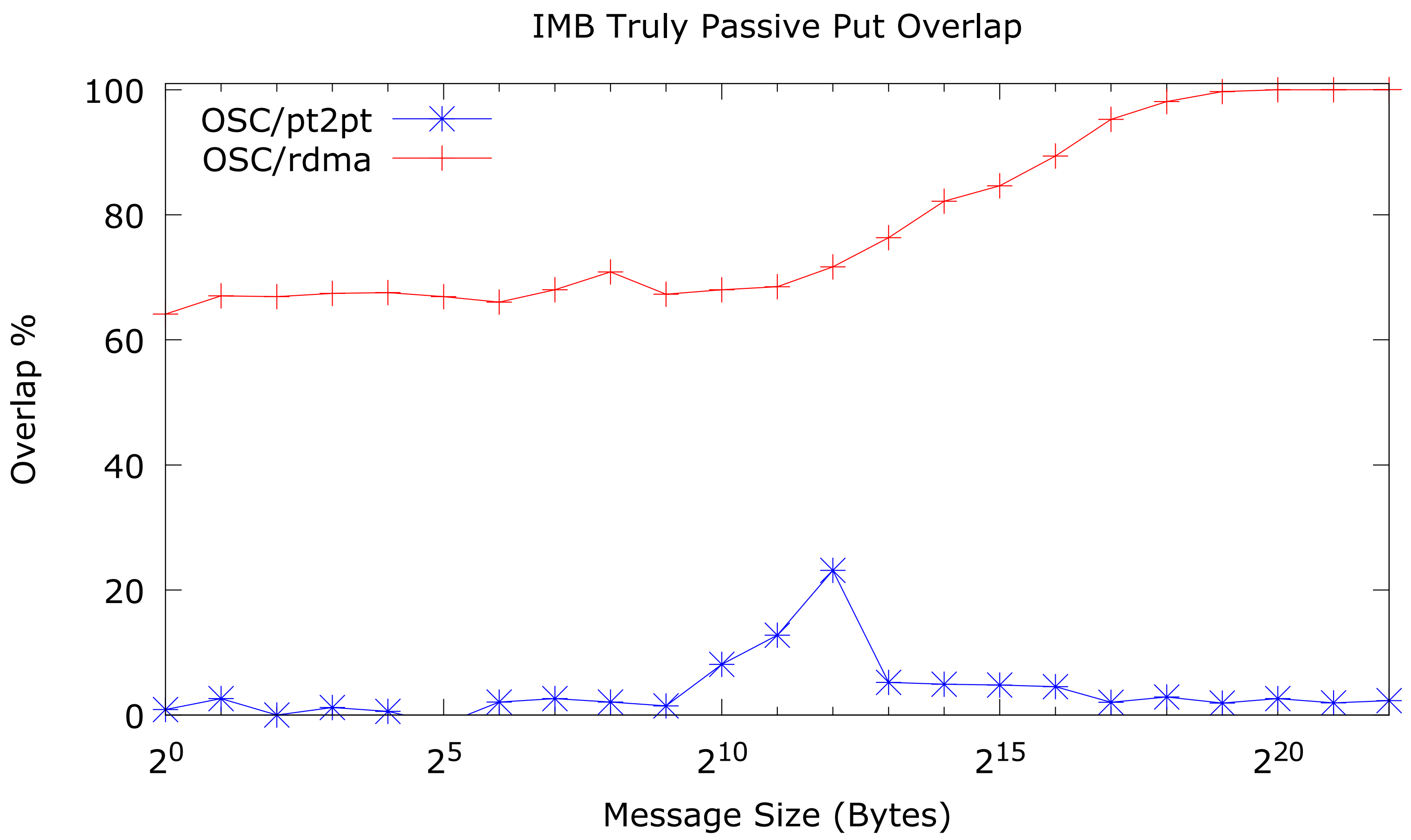
- Fully supports MPI-3 RMA
- Full support for MPI datatypes
  - Stress tested with ARMCI
- Emulates one-sided operation using point-to-point components (PML) for communication
- Caveat: no asynchronous progress support
  - Targets must enter MPI to progress any one-sided data movement (!)
  - Defeats the purpose of passive target RMA

v2.x RMA

- Uses network RMA and atomic operation support through Byte Transport Layer (BTL)
  - Fully supports passive target RMA operations
- Initially supports:
  - IB*, Infinipath/Omnipath**, Gemini/Aries
- More networks can be supported if they have
  - put, get, fetch-and-add, compare-and-swap
- Otherwise, fall back to emulated RMA

v2.x RMA performance

- IMB truly passive put on Cray XC-30



Beyond the v2.x series

FROM SC'14 BOF

What's the plan over time?

- *Plan* for a new release series once a year
  - v2.x: release in mid / late 2015
  - v3.x: release in mid / late 2016
  - v4.x: release in mid / late 2017
  - …etc.

NOTE: Scheduled releases is a new concept for the Open MPI developer community. We'll continue to evaluate this plan over time.

3.x series

- Branching for v3.x in June 2016 is… unlikely
  - As noted previously, v2.0.0 is taking longer than expected
- More likely:
  - The v2.x series will have a good life (~1 year?)
  - We'll postpone v3.x for a while
  - Stay tuned

MPI v4.0

- Reminder: go to the MPI Forum BOF
  - 3:30pm on Wednesday
- MPI v4.0 is at least 2 years away
  - Content far from certain
  - Too far off to make predictions
  - Will likely include portions of what will become MPI-4.0 over time

User question

What are the major implementation issues involved with the proposed MPI-4 endpoints? Any progress to report?

We (developers) have talked about implementing MPI-4 endpoints. But…

- We have identified a minimal set of changes in Open MPI for compliance with the current proposal
- The endpoints proposal is not final (yet)
- Not enough user demand (yet)

User question

What wire protocols are used? How do you tune them?
How does one tune FDR/EDR/OPA fabrics?

This is a bit more than we can cover in a 1-hour BOF.

Please come see us afterwards.

Where do we need help?

- Code
  - Any bug that bothers you
  - Any feature that you can add
- ***User documentation***
- Testing
- Usability
- Release engineering

Researchers: how can we help you?

- Fork OMPI on GitHub
- Ask questions on the devel list
- Come to Open MPI developer meetings
  - Next: February 22-25, 2016, Dallas, TX, USA
- Generally: be part of the open source community

Come Join Us!

http://www.open-mpi.org/