Open MPI State of the Union Community Meeting SC '14

November 19, 2014

Jeff Squyres — Cisco

George Bosilca — ICL UT

Nathan Hjelm — Los Alamos National Laboratory, EST. 1943

Laust Brock-Nannestad — DTU

Open_MPI_Init()

```
shell$ git log --reverse  | head -n 5
commit 350564b9f381dfbdbe119f26585f07da6f4b9e8a
Author: Jeff Squyres <jsquyres@cisco.com>
Date:   Sat Nov 22 16:36:58 2003 +0000

    First commit
```

Open_MPI_Current_status()

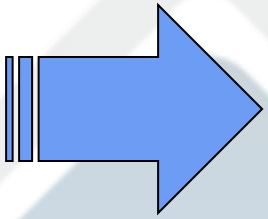
```
shell$ git log HEAD~1..HEAD
commit 34c156759ecde11c3bf6252050a14a9432c91405
Author: Howard Pritchard <hppritcha@gmail.com>
Date:   Tue Nov 18 11:32:37 2014 -0700

    fix some compiler warnings in ras/alps
```

Open MPI 2014 membership

12 members, 34 contributors, 2 partners

We migrated (!)



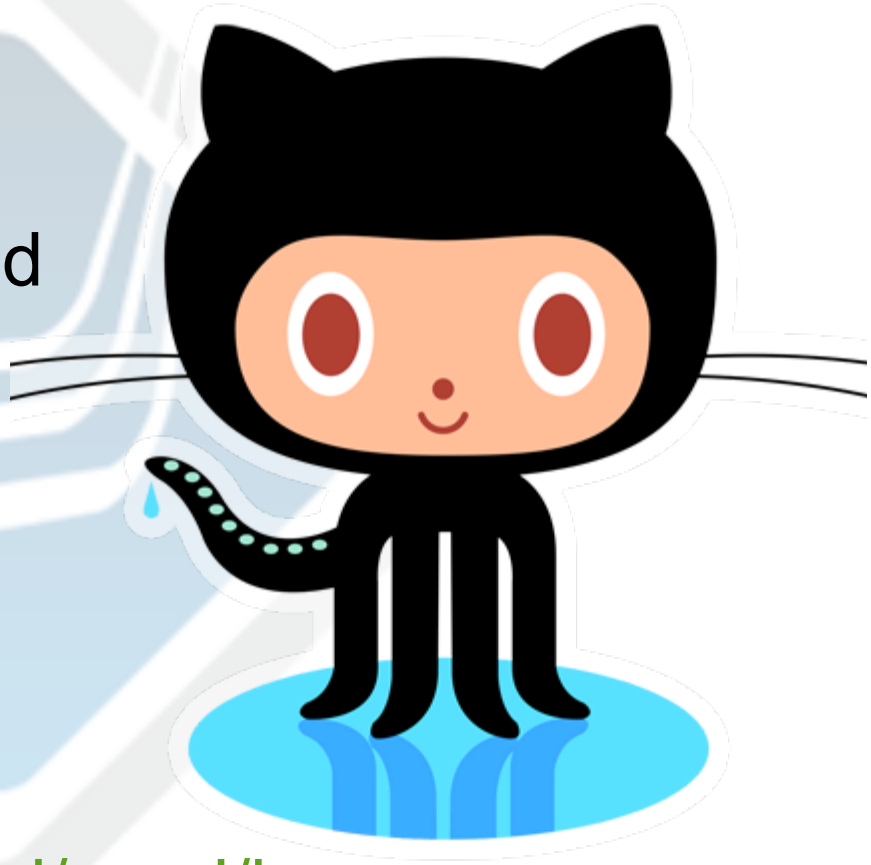
- The move was disruptive, but successful
  - We kept the entire history
  - All tickets
  - Kudos to Jeff and Dave
- But hopefully worth it

Moved hosting to GitHub

- Git
  - Encourage forks
- Github
  - Social coding
  - Better collaboration tools (discussion / code comment)
  - Easier to work with individual patches / contributions

Moved hosting to GitHub

- SVN and Trac now in read-only mode
- Please file new bugs and pull requests on GitHub

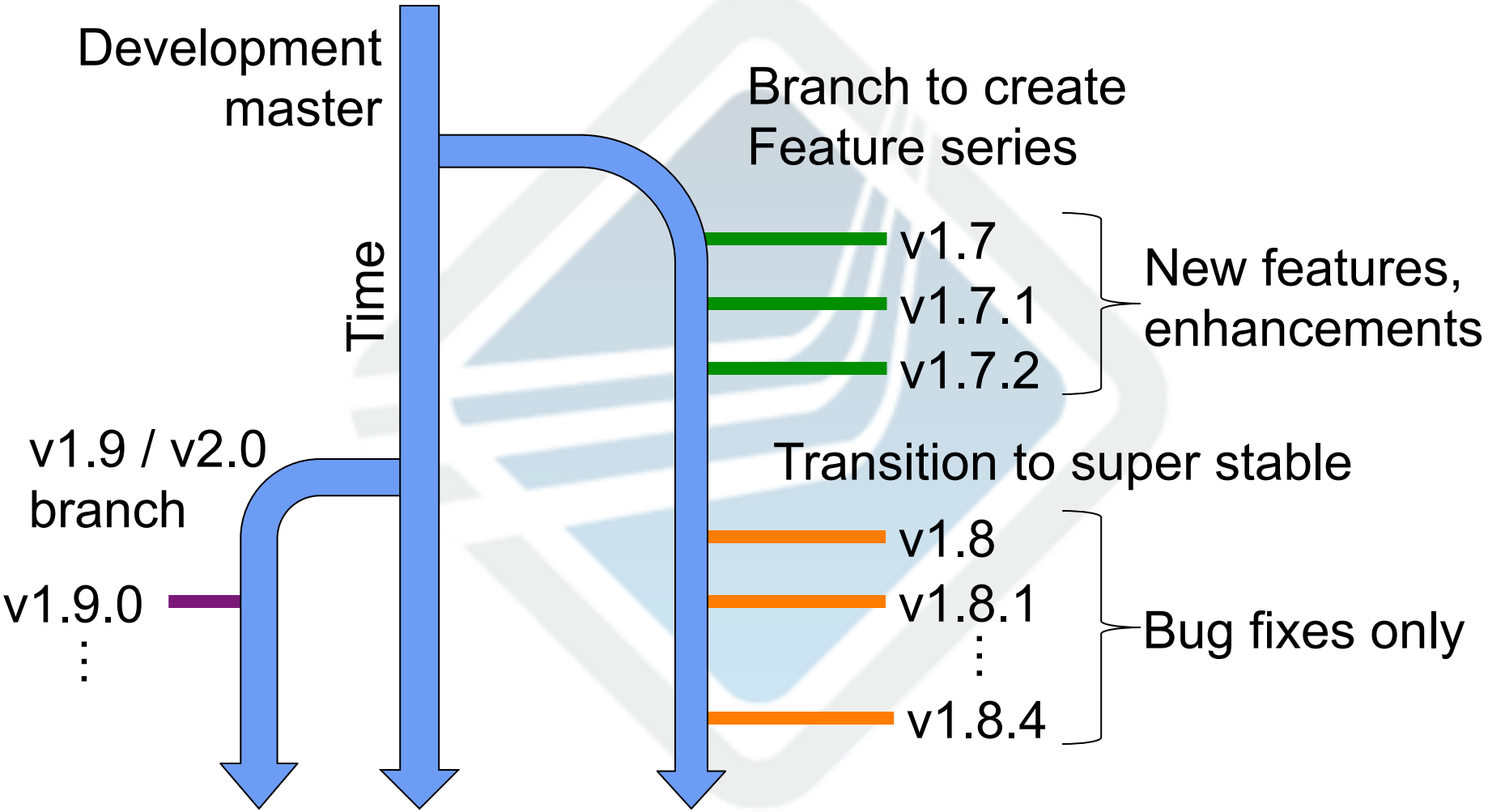


https://github.com/open-mpi/ompi/issues

Versioning scheme

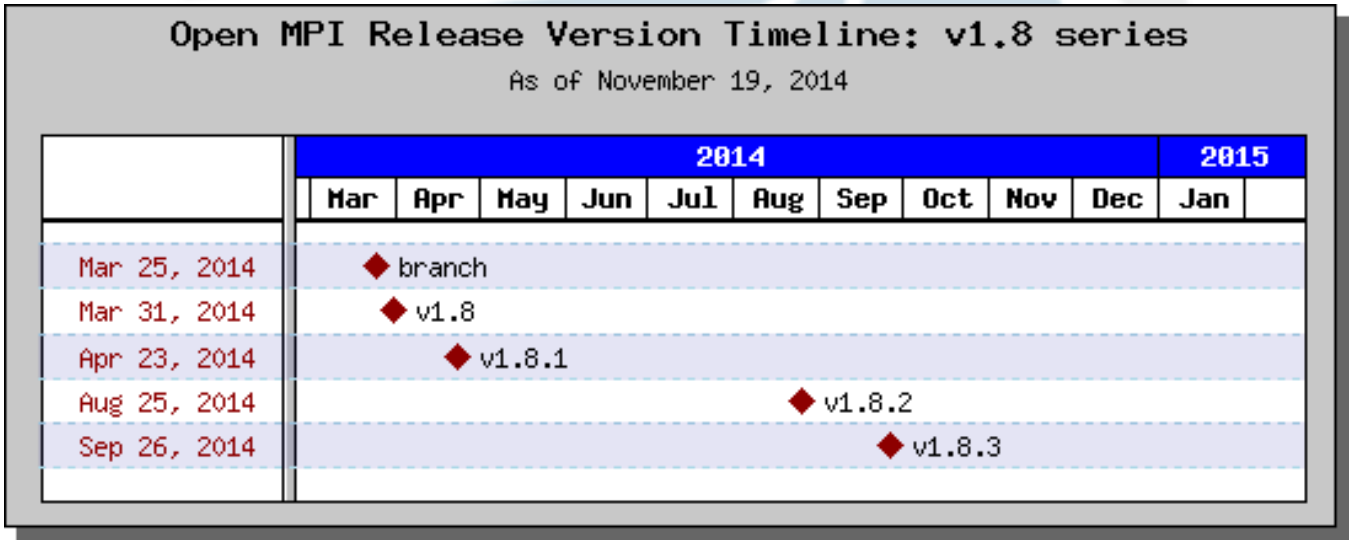
- Open MPI has 2 concurrent release series
  - "Tick / tock" versioning scheme
  - "Feature series" → v1.<odd>
  - "Super stable series" → v1.<even>
- Both are tested and QA'ed
  - Main difference between the two is time

Feature / stable series



v1.8 roadmap

- 1.8.4
  - …to be released very soon (hopefully) December 2014
- v1.8.5
  - Likely to be another 1.8.x release containing minor bug fixes and cleanup



What's new in 1.8

- OpenSHMEM is now part of Open MPI
- Improve support for the MPI Fortran bindings
- Improved CUDA support (non-blocking and async)
- Performance improvement for small messages over blocking communications
- Full MPI coverage in Java
- Valgrind-friendly (!)
- Added the new MPI 3.1 tool interface
- Better startup and shutdown (PMIx)

v1.8 MPI conformance

- Rock solid

MPI 3.0	1.8
NB collectives	✔
Neighborhood collectives	✔
RMA	✔
Shared memory	✔
Tools Interface	✔
Non-collective comm. create	✔
F08 Bindings	✔
New Datatypes	✔
Large Counts	✔
Matched Probe	✔

1.6 → 1.8 gotchas

- Mapping / binding / ranking
  - --map-by
  - --bind-to
  - --rank-by
  - New options: L1 cache, etc.
  - mpirun.1 man page is (finally) updated in 1.8.4
- Hostfile: if you don't say "slots=N", Open MPI autodetects

1.6 → 1.8 gotchas

- Binding by default (!)
- Launch on all available nodes at first
  - Except if you're in an allocation and you –host a,b,c, then you'll only VM launch on a,b,c
- Hetero topology: --hetero-nodes
  - To include alloc'ing different cores on different servers
- Be easy on MPI_THREAD_MULTIPLE support

1.8.x notable bug: THREAD_MULTIPLE

- MPI_THREAD_MULTIPLE was accidentally enabled
  - Performance degraded
  - Particularly in shared memory latency
- To be fixed in v1.8.4

v1.9 / v2.0 Series

v1.9 / v2.0 series

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

v1.9 / v2.0 series

- Version number changes
  - “v1.9.0” (vs. “v1.9”)
  - v2.0 – reflect the scope of changes across the v1.9 series

v1.9.0 [tentative] timeline

January, 2015

- Branch for v1.9 / v2.0 series

April, 2015

- v1.9.0 feature complete

July, 2015

- Release v1.9.0

https://github.com/open-mpi/ompi/wiki/Releasev19

v1.9 removed features

- Trim supported systems list in README
  - …maybe delete Solaris?
- Cray XT legacy items
- Outdated / orphaned plugins (i.e., deleted)
  - MX
  - “hierarch” collective setup

v1.9 / v2.0 MPI conformance

- MPI-3.1 planned conformance for v1.9 series (not yet published)
  - Various errata, non-blocking I/O
  - Will be included in v1.9 series
- MPI-4.0 …? (at least 2 years away)
  - Content far from certain
  - Too far off to make predictions
  - Will likely include portions of MPI-4 over time

Threads

- MPI_THREAD_MULTIPLE
  - For real. Really.
  - Transport-specific
    - …we're flogging transport authors to make their transport thread-safe
- Asynchronous progress
  - …same flogging above applies

Open MPI I/O (OMPIO)

- Research work from the University of Houston
- Highly modular architecture for parallel I/O
  - Adaptability through MCA parameters
- Selected OMPIO Highlights
  - Multiple Collective I/O algorithms supported
  - File view based automatic selection of collective I/O module
  - Automatic adjustments of number of aggregators
  - Enhanced support for shared file pointer operations

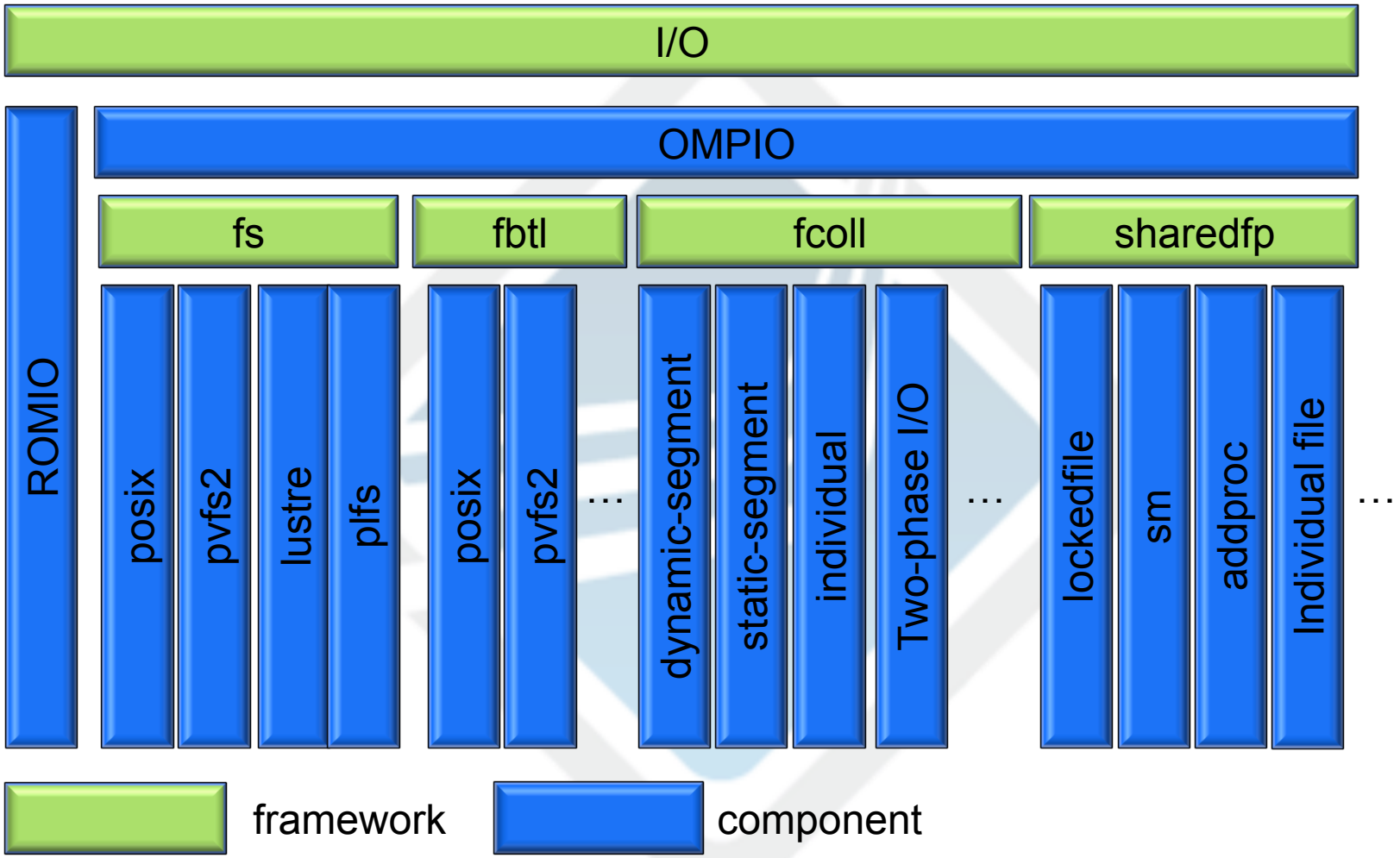
This project is funded in part by NSF through grant SI2-SSE 1339763.

Open MPI I/O (OMPIO)

- Deeply integrated with Open MPI
  - Derived data type optimizations
  - Main progress engine used for non-blocking I/O operations
- Already available in v1.7 / v1.8 series
  - But not the default
- Significantly enhanced and stabilized version in upcoming v1.9 series

This project is funded in part by NSF through grant SI2-SSE 1339763.

OMPIO frameworks overview



This project is funded in part by NSF through grant SI2-SSE 1339763.

Extended Process Management Interface (PMIx)

- Collaboration between Intel and Mellanox
- MPI job launch time is a hot topic!
  - Extreme-scale system requirements
  - 30 second job launch time for $O(10^6)$ MPI processes
- PMI and PMI2 have measureable limitations at scale
  - Intent is to address these limitations in PMIx

PMIx: How does it work?

- Client-server model – same as PMI / PMI2
  - Revamp API to minimize data exchanges
- Support for:
  - Blocking and non-blocking collective operations
  - Binary blobs – eliminates the need to slice/ encode/decode/reassemble “meta-keys”
  - Bulk “get” operations and prefetch through shared memory

PMIx: How does it work?

- Support for bulk collectives
  → Well suited to applications with dense connectivity
- Support for point-to-point operations
  → Ideal for applications with sparse connectivity
- Hints intended to decrease the volume of data exchanged globally
  - Global
  - Local
  - Remote

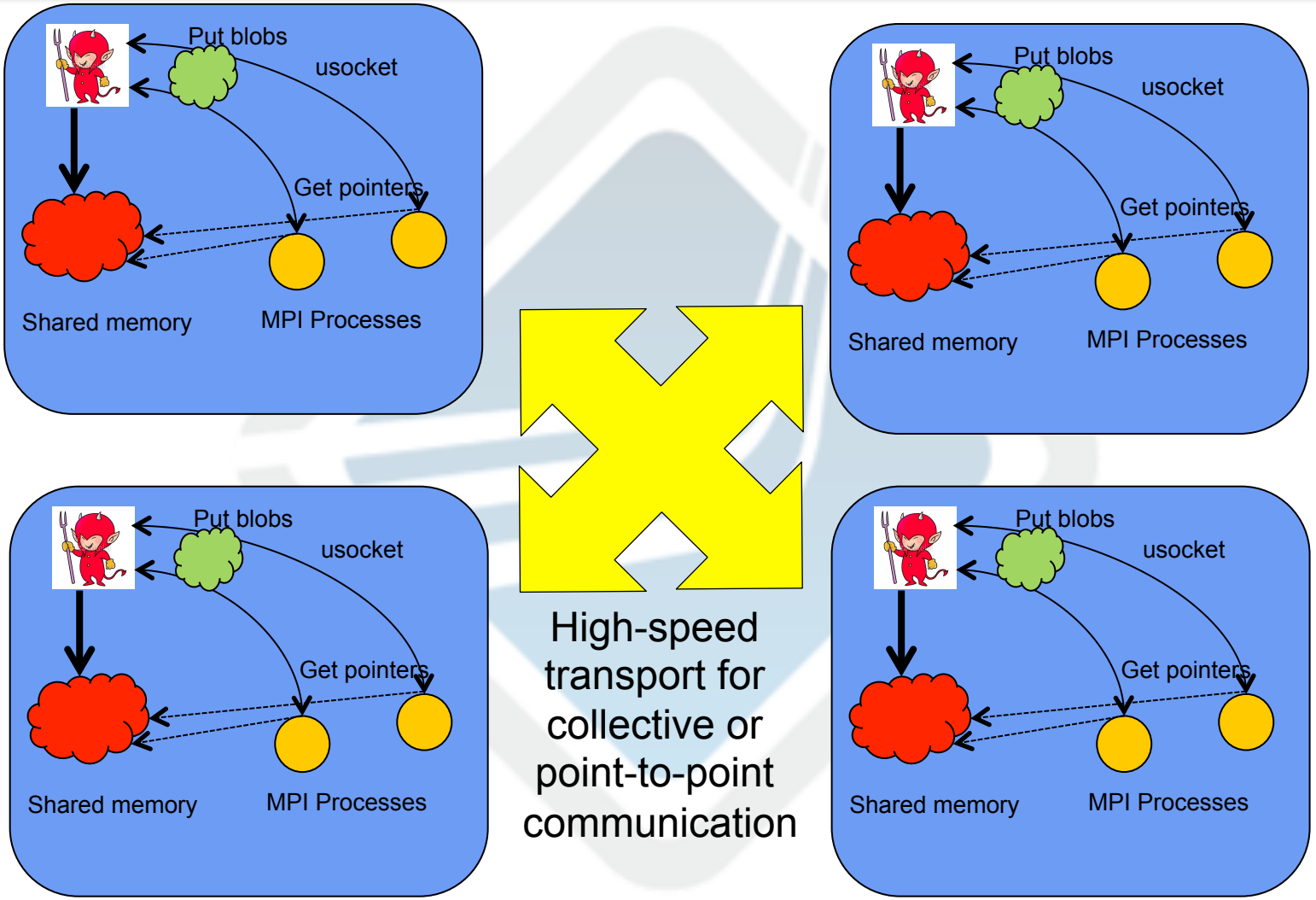
Status

- Reference implementation in Open MPI 1.9 server side over ORTE
  - Can leverage high-speed interconnects via BTLs for PMIx daemon operations
- Work in progress:
  - Extract client-side into standalone library

    → MPI implementation agnostic
  - Server implementations for SLURM and ORCM

    → Used in direct launch scenarios – e.g. srun

High-level overview



Contribute or follow along!

- https://github.com/open-mpi/pmix/wiki
- Interested in learning more?
  - Mail Ralph Castain (Intel) or Josh Ladd (Mellanox)
  - rhc@open-mpi.org
  - joshual@mellanox.com

Yalla PML

- MXM (Mellanox Messaging Library) specific PML
  - Reduces software overheads, minimizes time-to-wire
- Significantly outperforms OpenIB BTL in terms of message rates
  - 1.6x increase in message rate over OpenIB BTL (!)
- Lowers latency as well

Performance

Ivy Bridge 2.7 Ghz, Connect IB HCA

PML / MTL or BTL / Provider	Latency (microsec.) osu_latency	Millions of messages /sec osu_mbw_mr
CM / MXM / Mellanox	1.14	4.9
Yalla / None / Mellanox (Plugs directly into MXM)	1.08	7.3
OB1 / OpenIB / OMPI	1.11	4.5

Large-scale job start performance improvements

- Collaboration between Mellanox and Hewlett-Packard
- Better internal hash table implementation
  - Significantly improves data retrieval time
  - Grows and shrinks dynamically
  - Reduces OMPI job start time at scale by ~20%

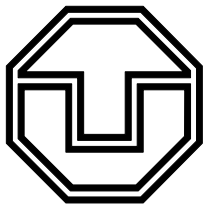
OpenSHMEM

- Work done by Mellanox
- Test kit released into open source:
  - https://github.com/openshmem-org/tests-mellanox
- Added support for hardware atomics in ikrit SPML
  - For RC, DC
- OSHMEM startup improvement
  - Added scalable algorithms for bootstrapping
- OSHMEM collectives can use collectives from OMPI COLL framework, e.g. FCA, HCOLL
  - Added a new SCOLL component “MPI”
  - **`--mca scoll_mpi_enable 1`**

Open MPI and ZIH

Past, Present, and Future

Bert Wesarg
ZIH, TU Dresden



TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services & High Performance Computing

The Past

- VampirTrace has been part of Open MPI since Version 1.3
  - Committed to trunk January 2008
- Supports MPI, multiple threading paradigms, and CUDA
- Only one major performance analysis tool

The other past

- In 2009 the ZIH started participating to build a new performance measurement infrastructure, now named Score-P
  - http://score-p.org
  - Community driven
  - Governed by a consortium

- Writes profiles and traces in common data formats (without recompiling)
- Supported by multiple tools

The Present

- New features in Open MPI:
  - MPI-3
  - OpenSHMEM
- ..but VampirTrace is in maintenance mode
  - Does not support some of these new features
- ZIH is major contributor to Score-P, in particular the support for OpenSHMEM

The Future

- Score-P is a matured product
  - v1.0 in 2012
  - v1.3 in 2014
- Healthy and broad community
- Rapidly adopting new features

- Discussing integration with Open MPI

Advertisement

- Visit ZIH at booth #2323
- Score-P / Vampir talks
  - Today 1:30 PM
  - Tomorrow 10:30 AM and 1:30 PM

Open MPI: RMA support

Nathan Hjelm
LANL

RMA: current status

- Fully support MPI-3 RMA as of v1.7.4
- Full support for MPI datatypes
- Uses point-to-point communication components (PML) for off-node communication
- No asynchronous progress
  - Target must call MPI functions to progress RMA communication

LA-UR-14-28952

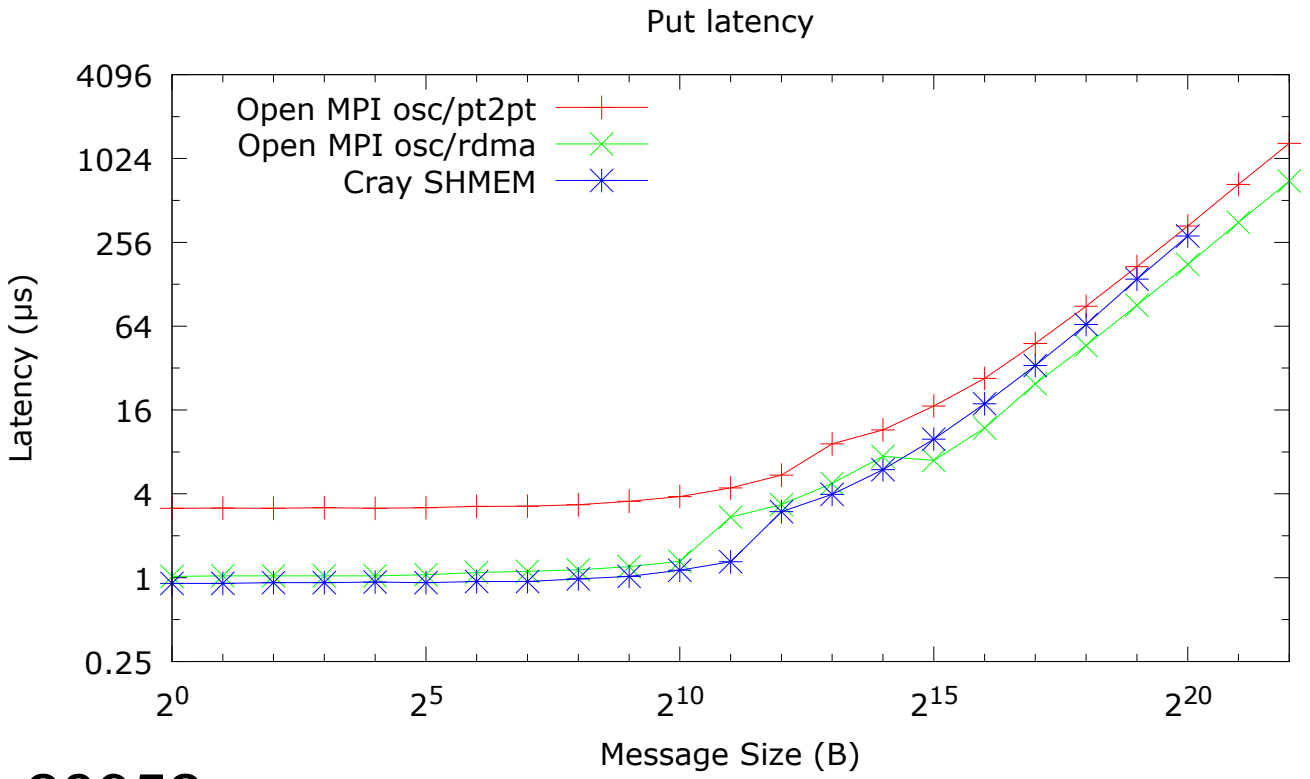
RMA: what's next?

- Use network RDMA and atomic operation support
  - Lower overhead, asynchronous progress, etc.
- This requires changes to the Byte Transport Layer (BTL) in Open MPI
  - Adding support for network atomics (compare-and-swap, fetch-and-add, etc)
  - Updating interface to better support RMA operations

LA-UR-14-28952

RMA: performance

- Early performance on Cray Gemini network



LA-UR-14-28952

RMA: wrap up

- BTL changes in master this week
- RMA optimizations will come later
- Will be available as part of the Open MPI v1.9 / v2.0 release series

LA-UR-14-28952

Exposing MPI Objects for Debugging

Laust Brock-Nannestad

Technical University of Denmark

Joint Work

- Developed in collaboration with:
  - John DelSignore, Rogue Wave Software
  - Jeffrey M. Squyres, Cisco Systems, Inc.
  - Sven Karlsson, Technical University of Denmark
  - Kathryn Mohror, Lawrence Livermore National Laboratory

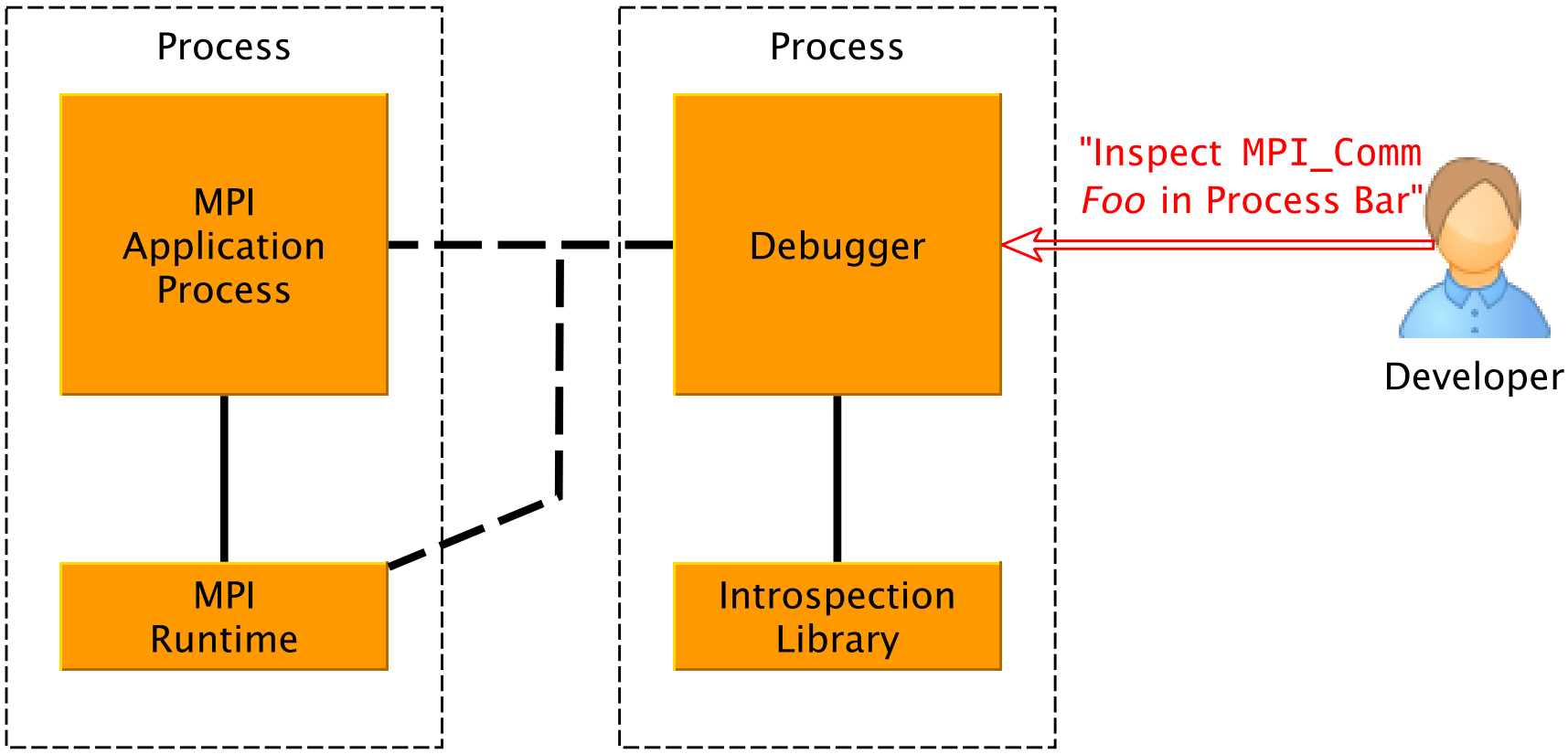
Motivation

- Debuggers have limited insight into the MPI runtime
- “What’s going on inside this communicator?”
- “Why did my program stall?”
- Runtime experts *can* debug these problems, but:
  - Application developers are not system developers
  - Time consuming and MPI implementation dependent
  - Some existing debug support – *Message Queue Dumping*

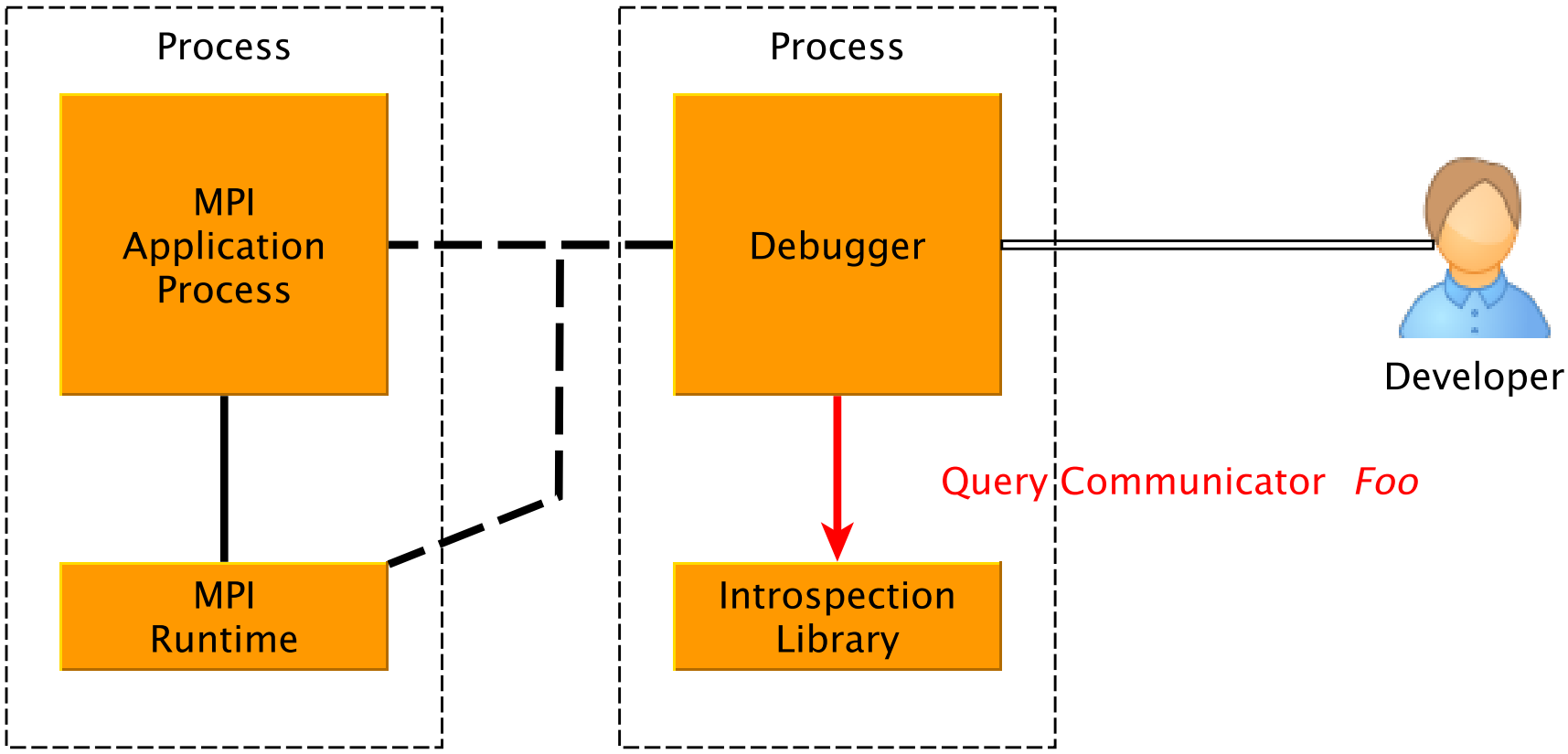
Contributions

- Proposal from the MPI Tools Working Group:
  - Debugger ⇔ MPI library interface for inspecting MPI handles
  - *MPI Handle Introspection*
- We implement the interface in *Open MPI* and a development version of the *TotalView* debugger
- We use the implementation to view MPI communicator state from the debugger

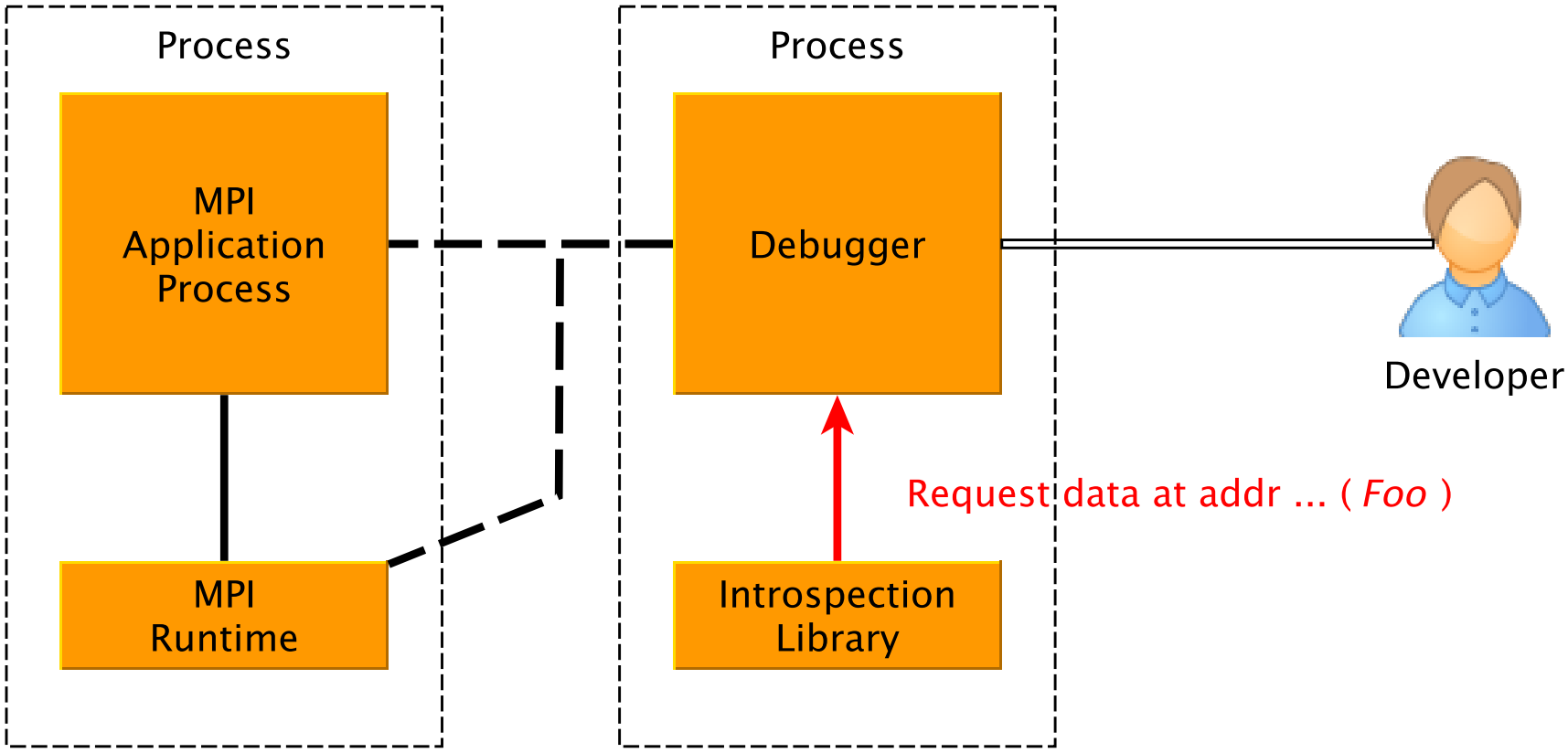
Operation 1/8



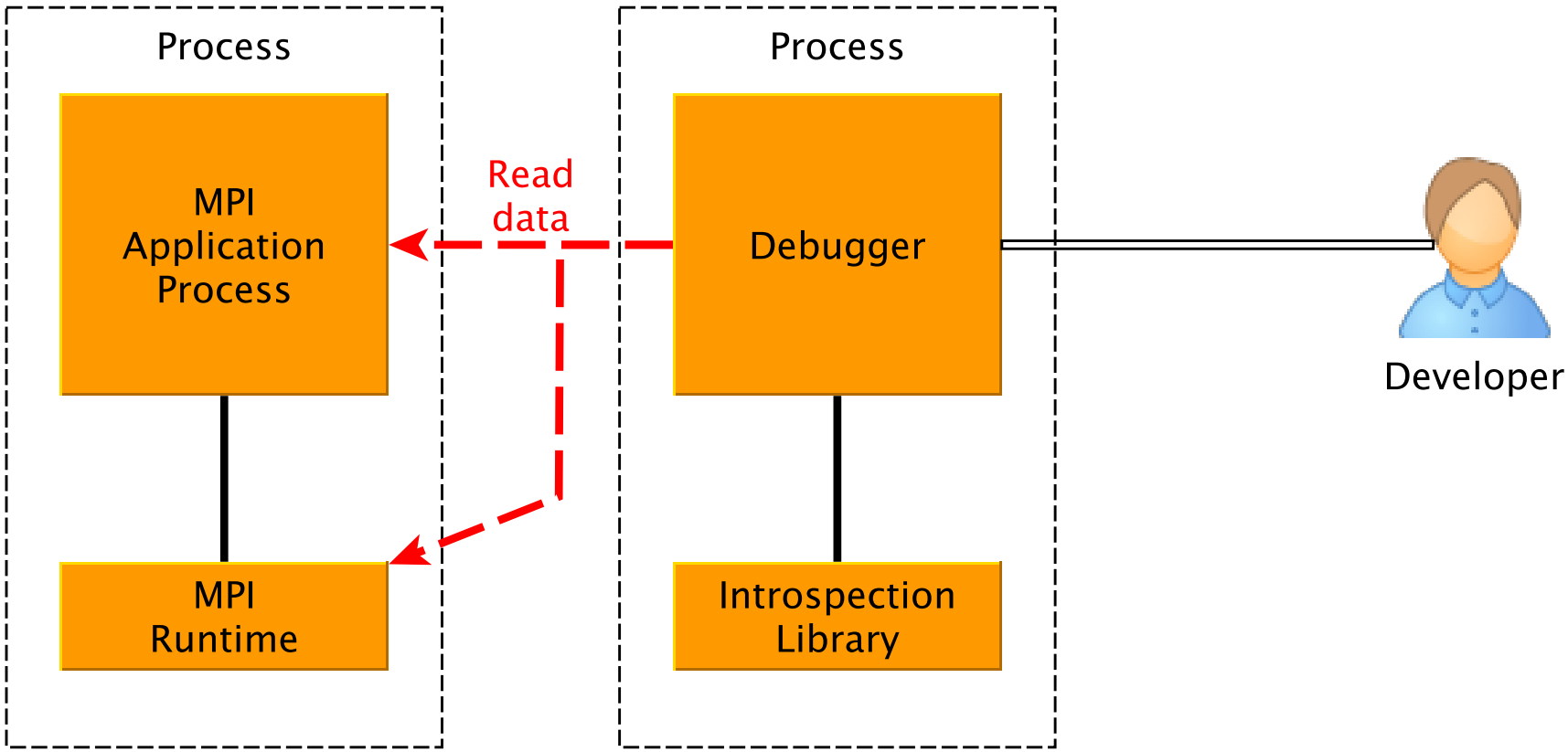
Operation 2/8



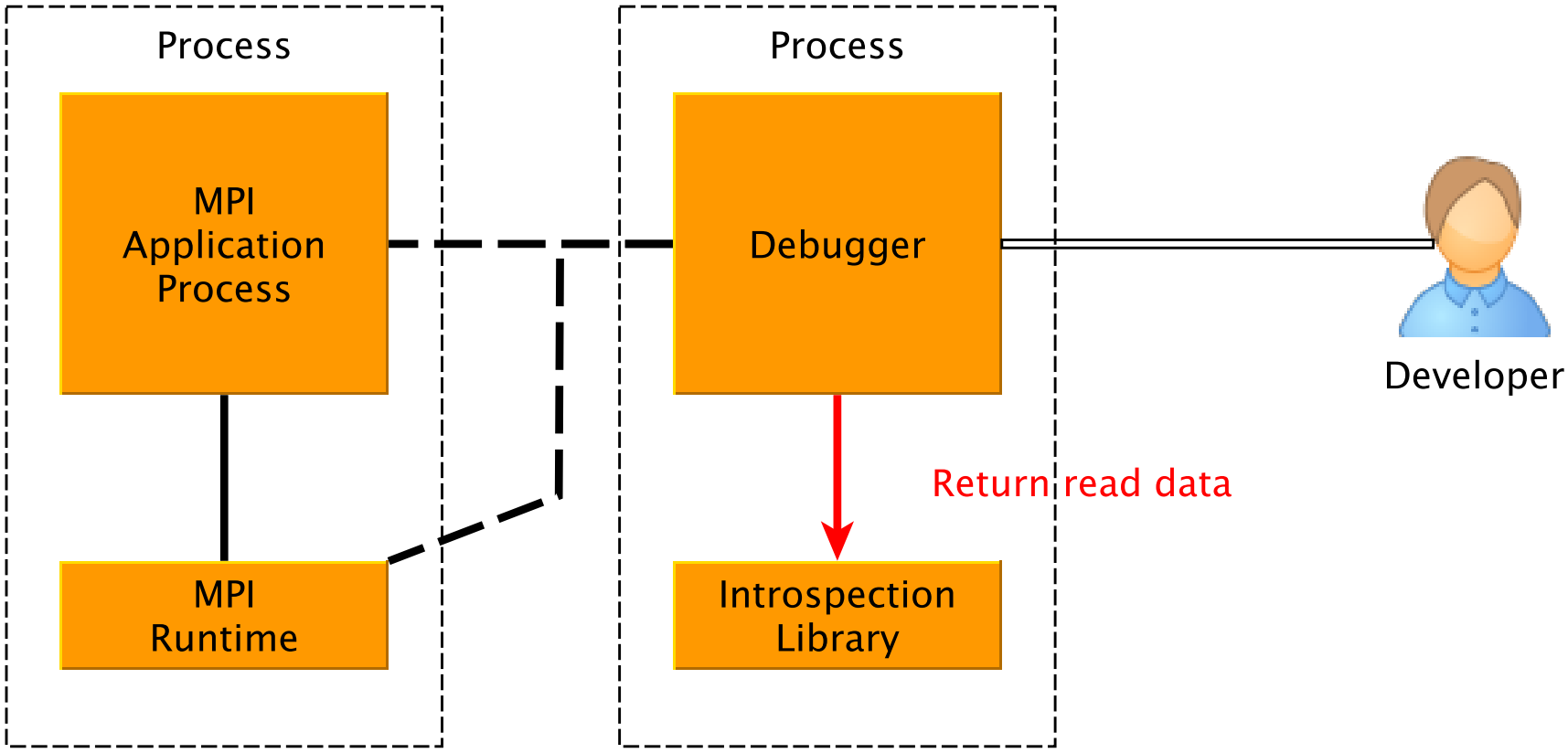
Operation 3/8



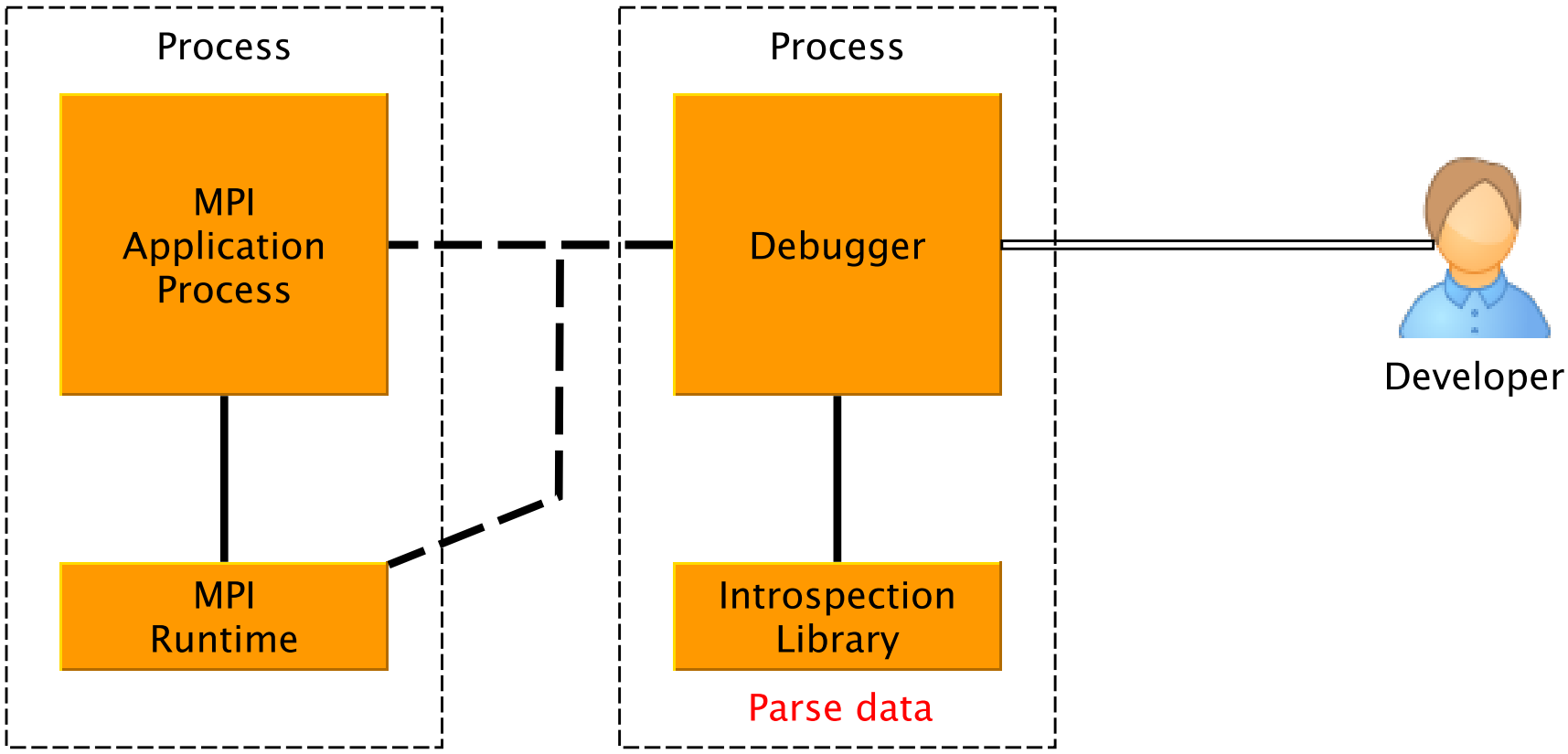
Operation 4/8



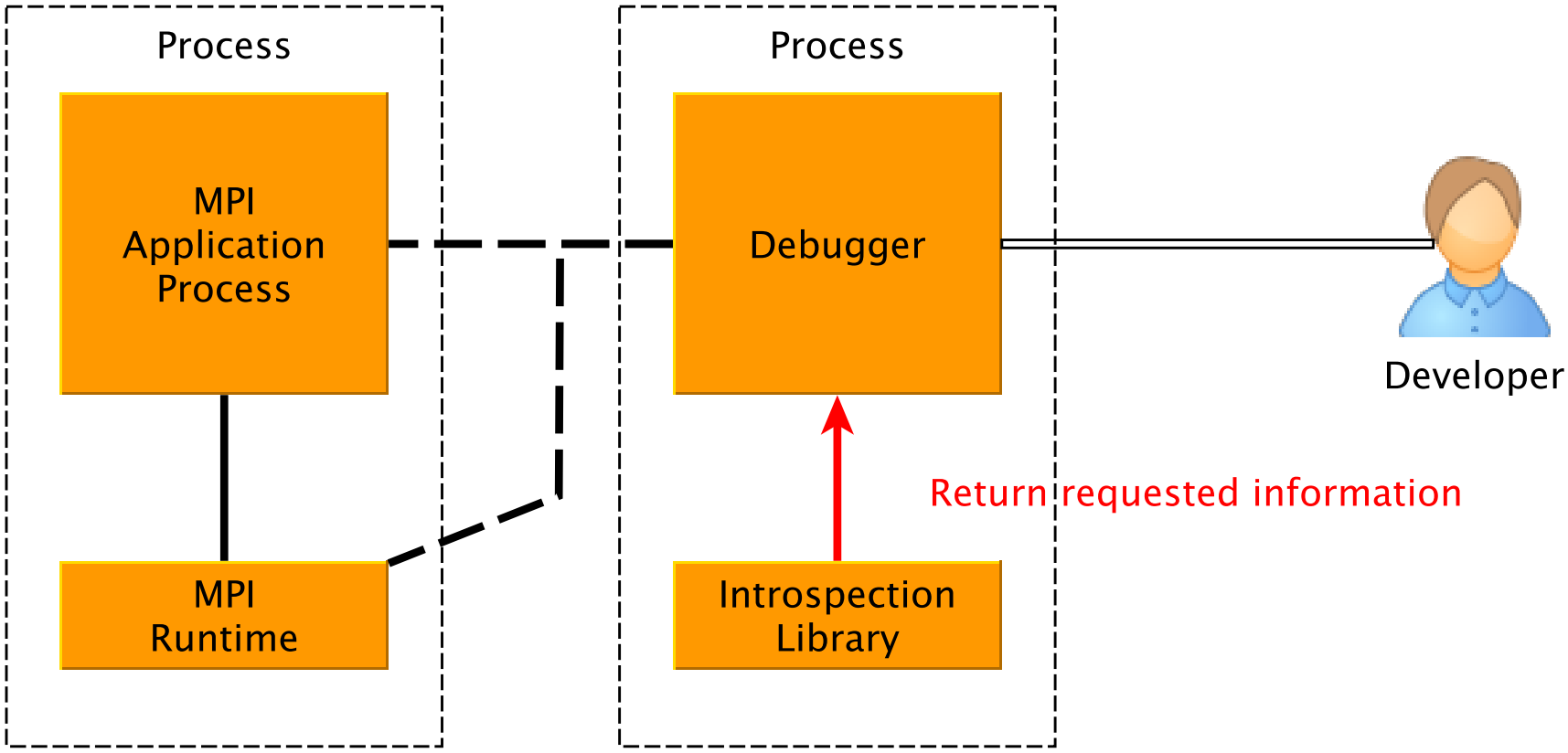
Operation 5/8



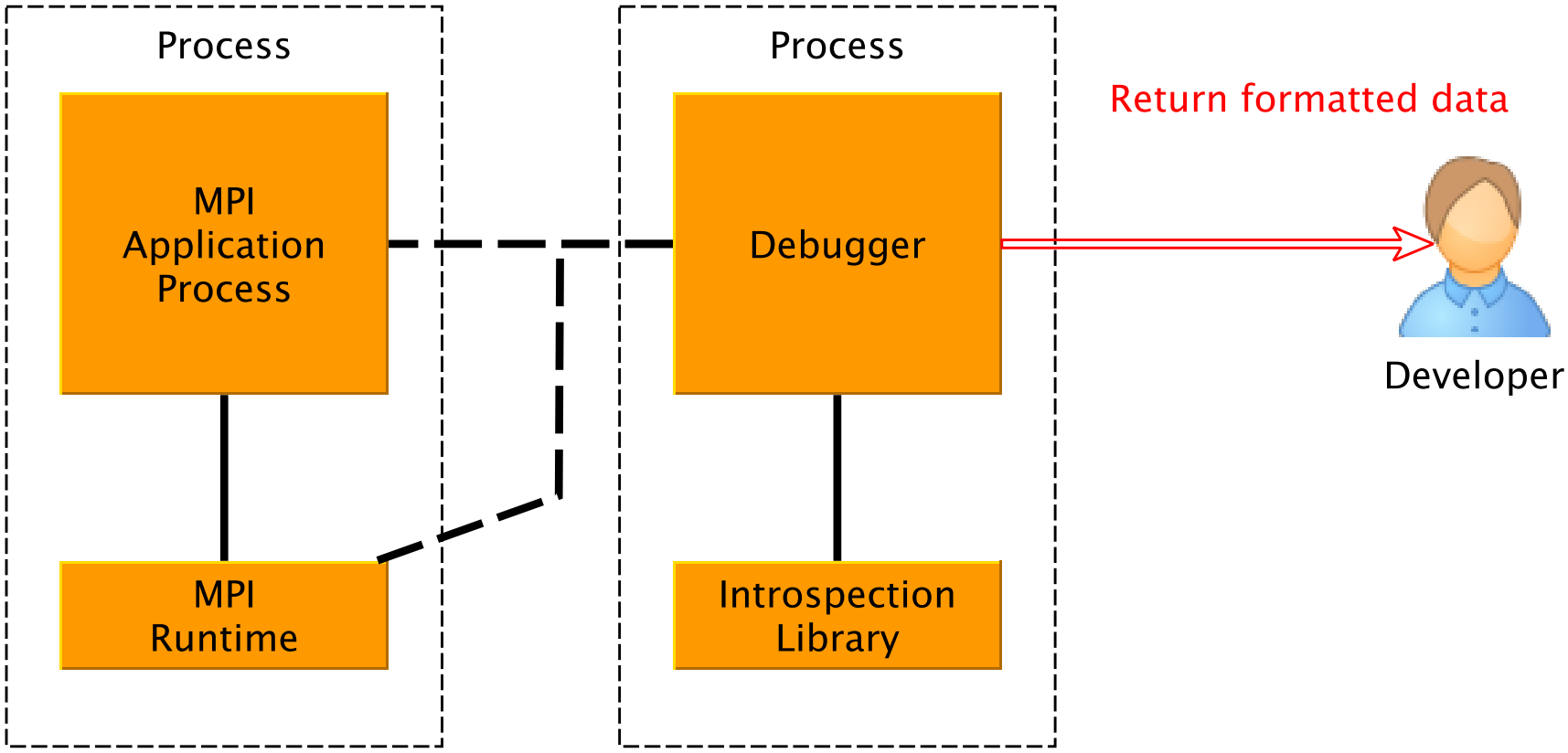
Operation 6/8



Operation 7/8



Operation 8/8



Debugger ⇔ MPI interface

- Debugger and MPI implementation agnostic
- MPI vendor provides a library

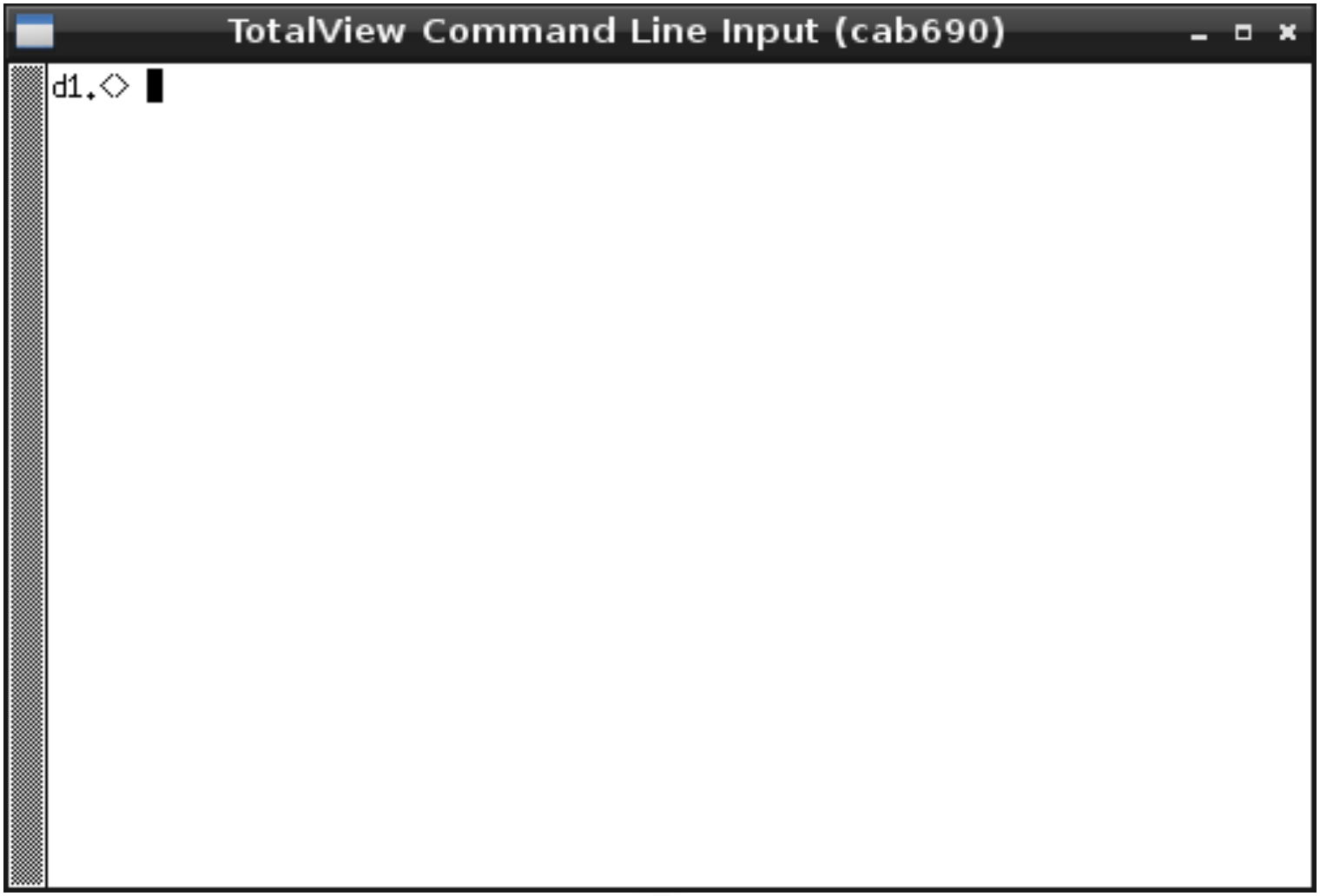
- Debugger services library
  - Provides read access to application's state
- MPI library services debugger
  - Give debugger insight into runtime – *introspection*

Implementation

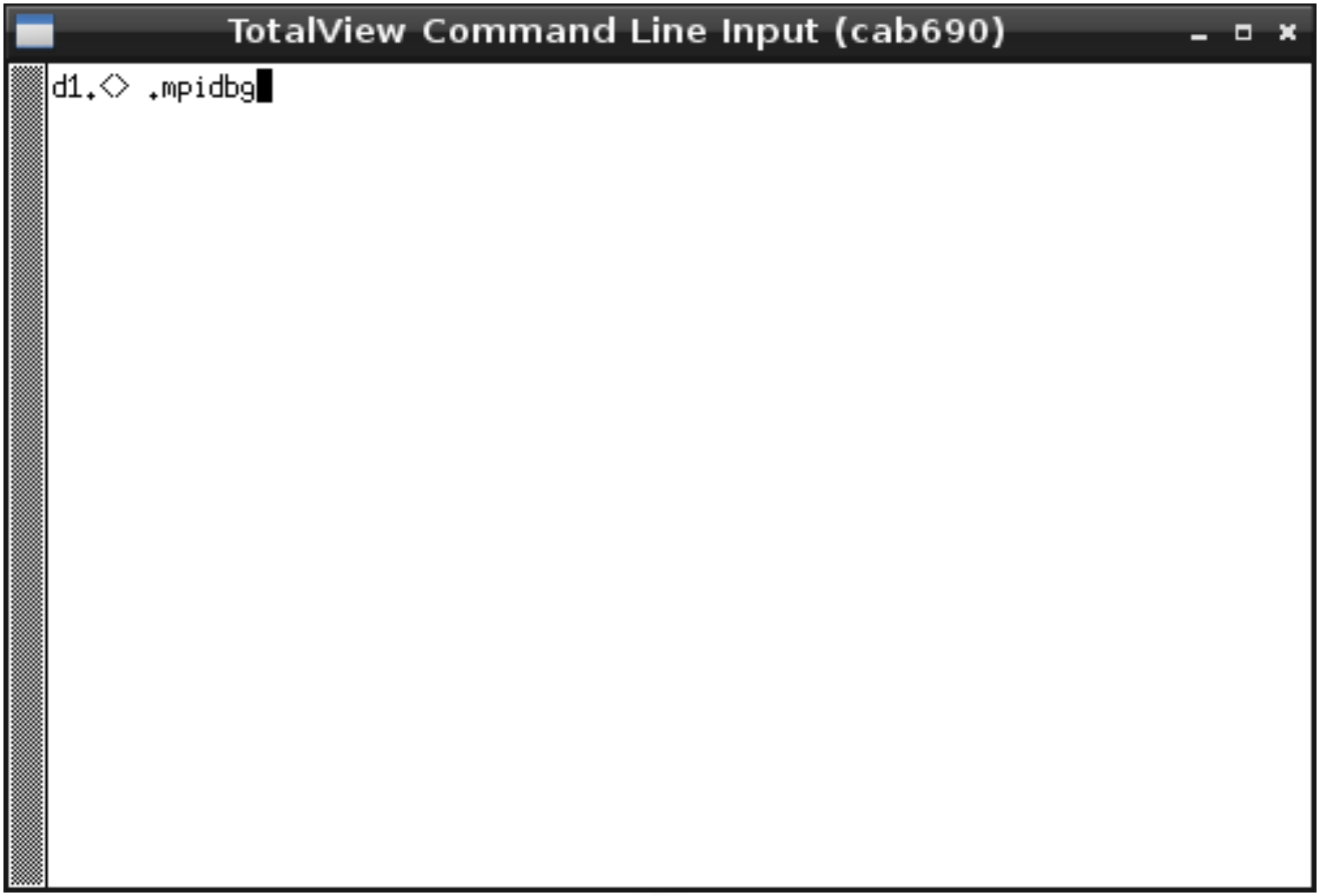
- Command line interface to TotalView
- Development version
- Queries can be performed on MPI Communicator handles

- Demonstration follows

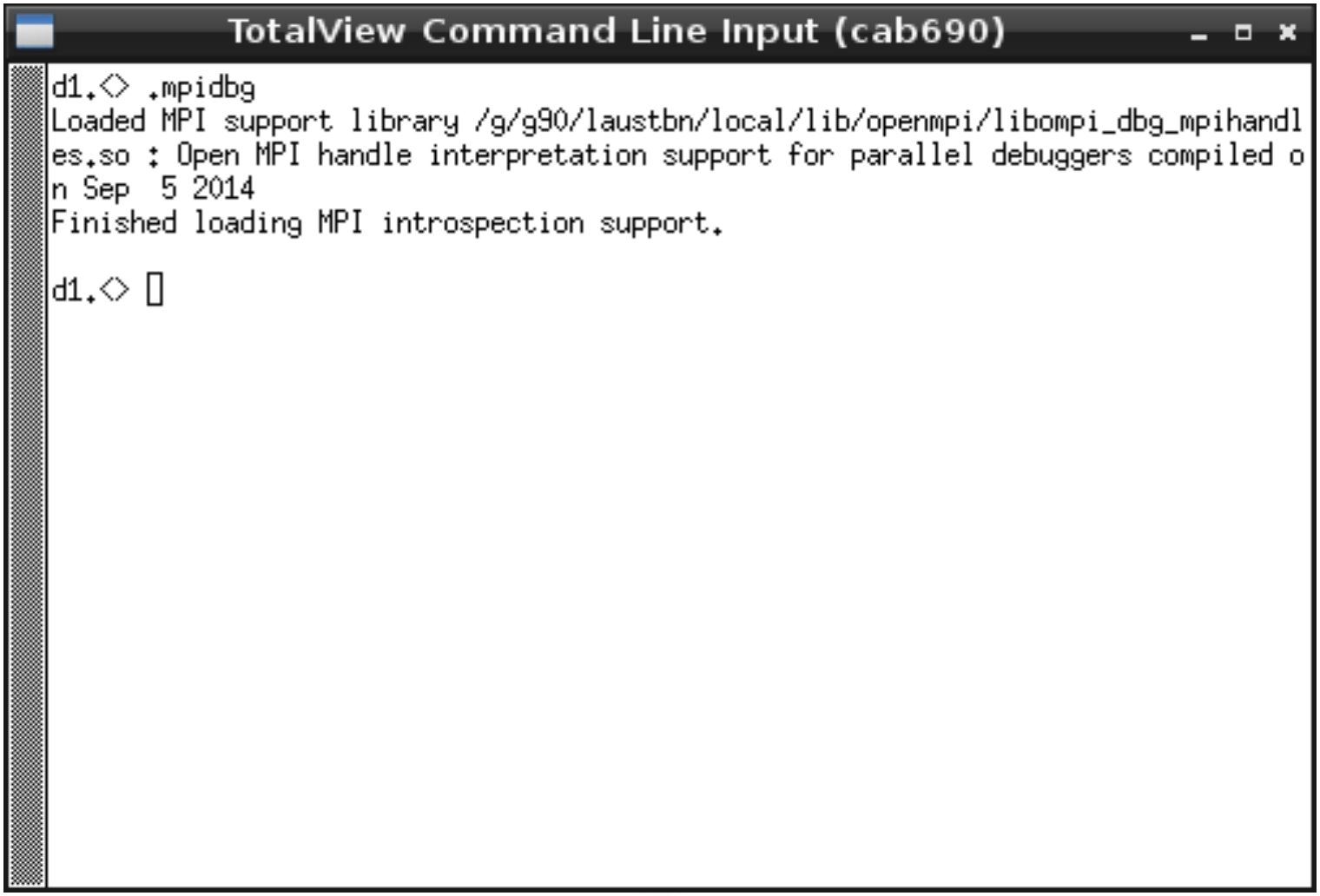
Demonstration



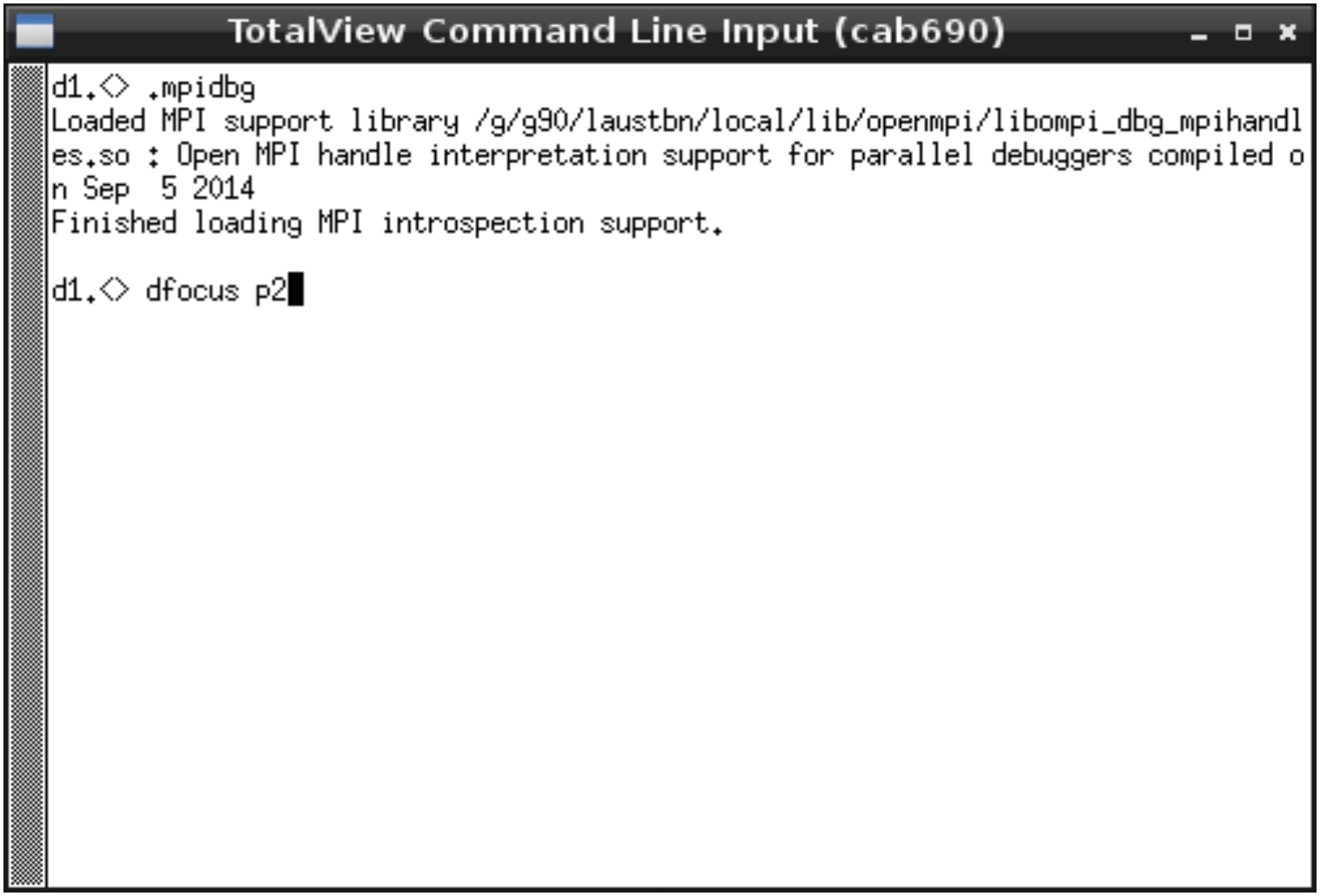
Demonstration



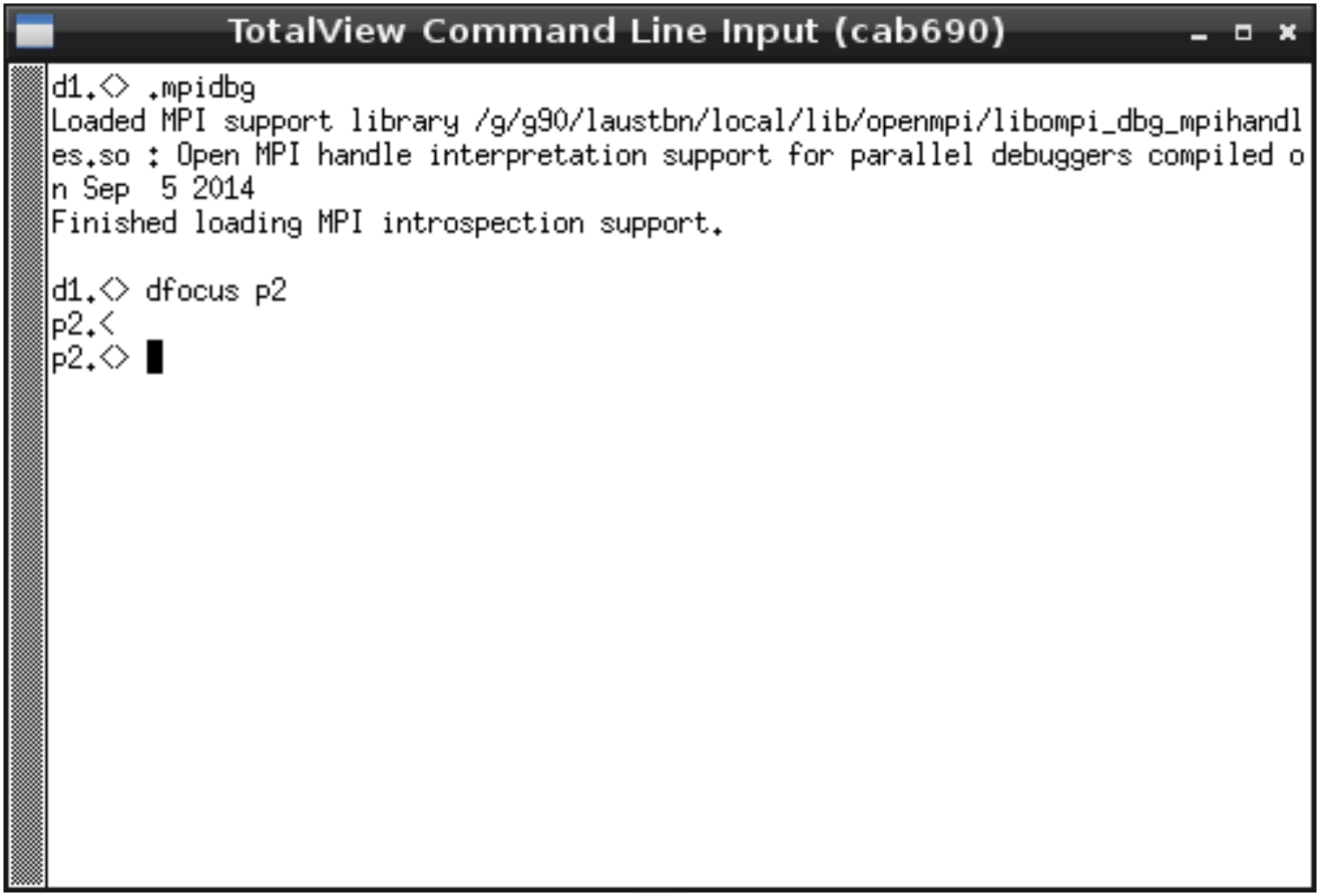
Demonstration



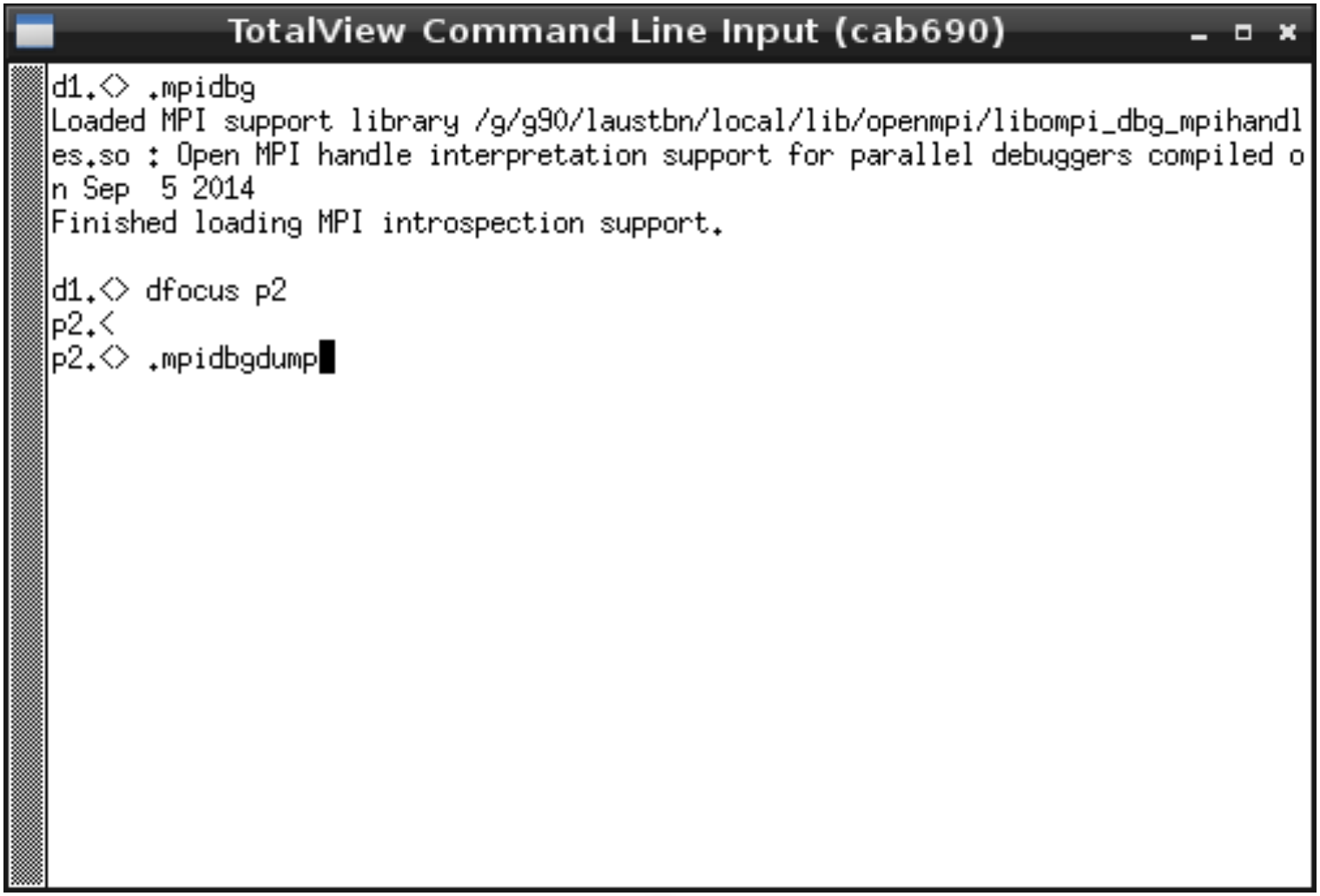
Demonstration



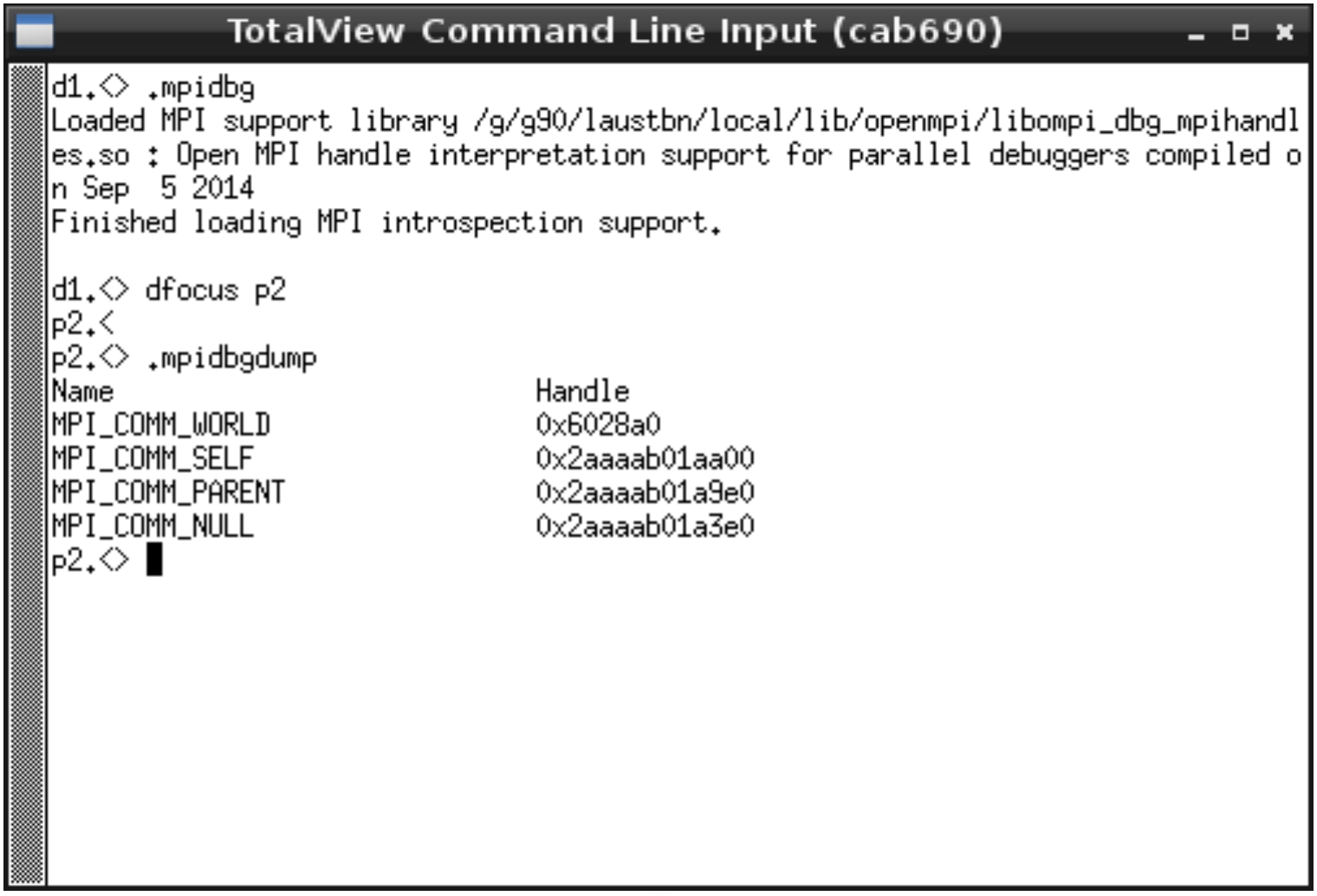
Demonstration



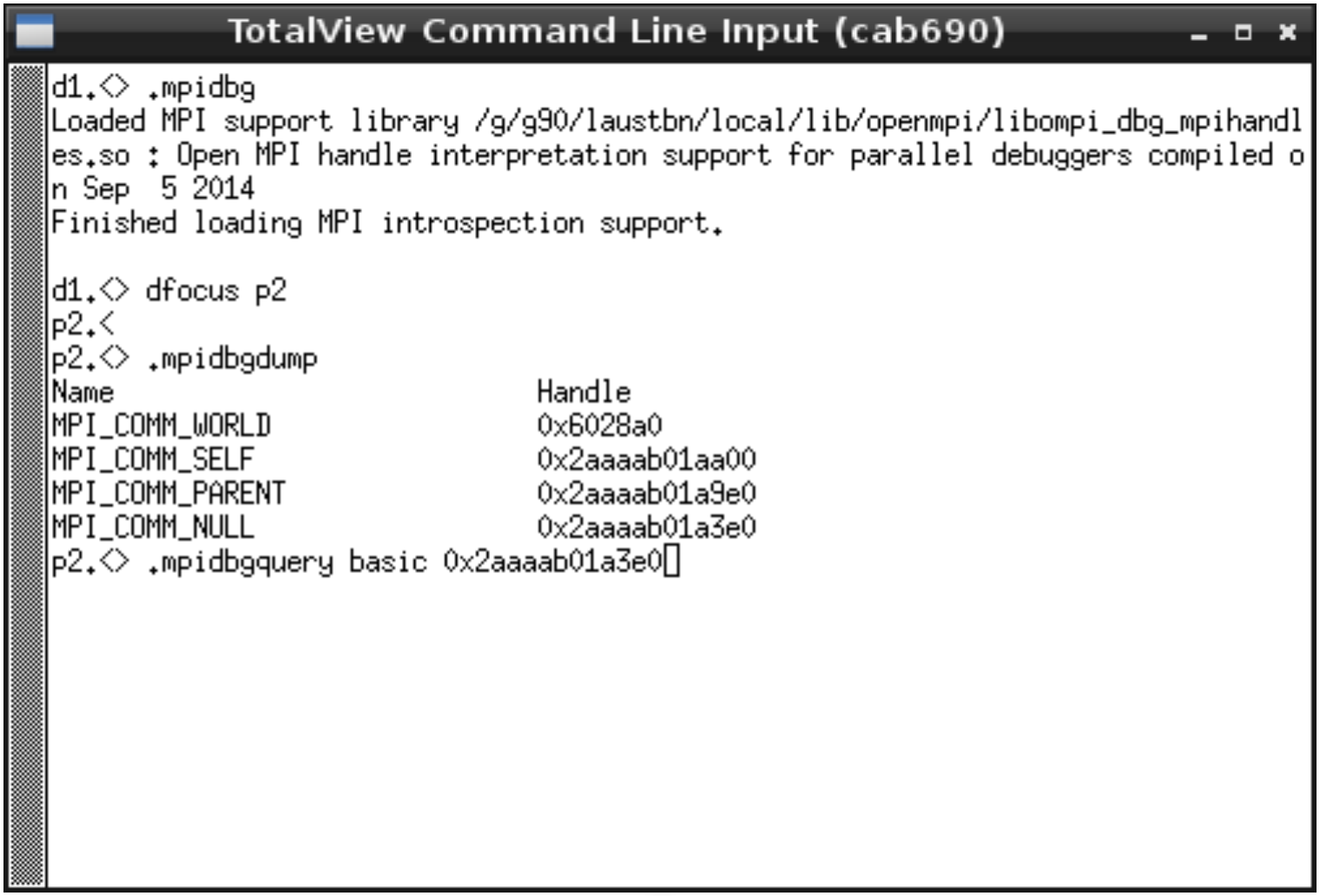
Demonstration



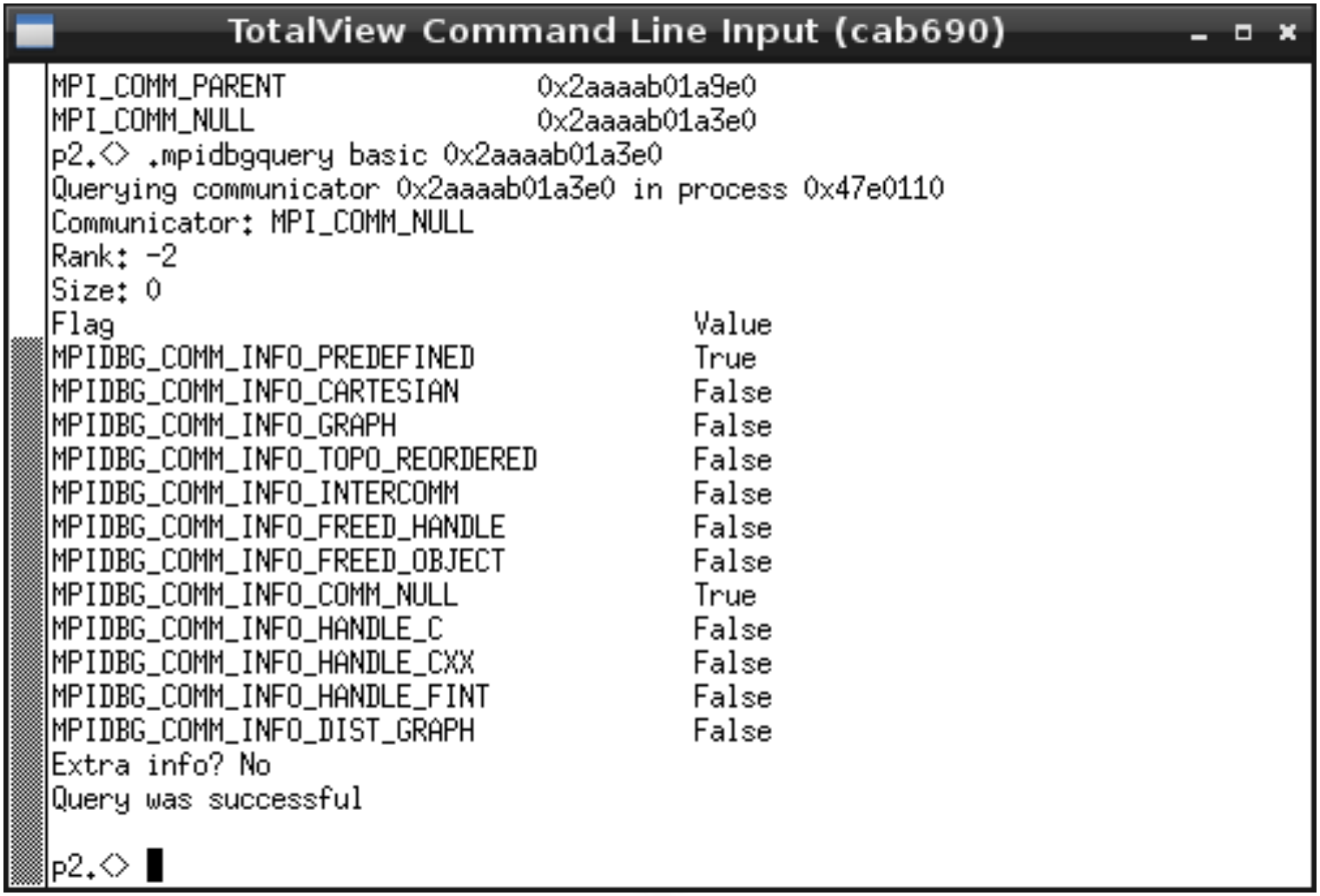
Demonstration



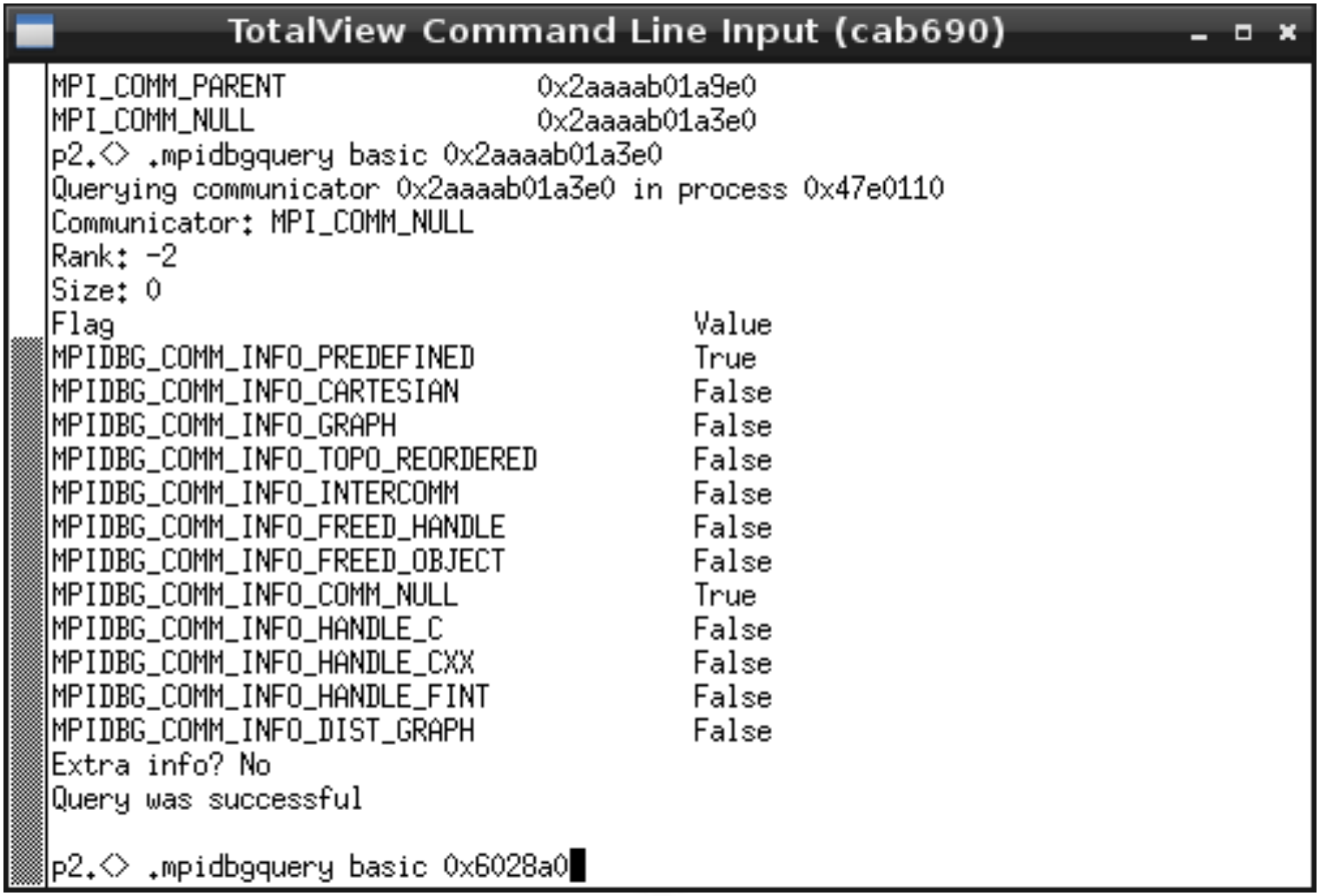
Demonstration



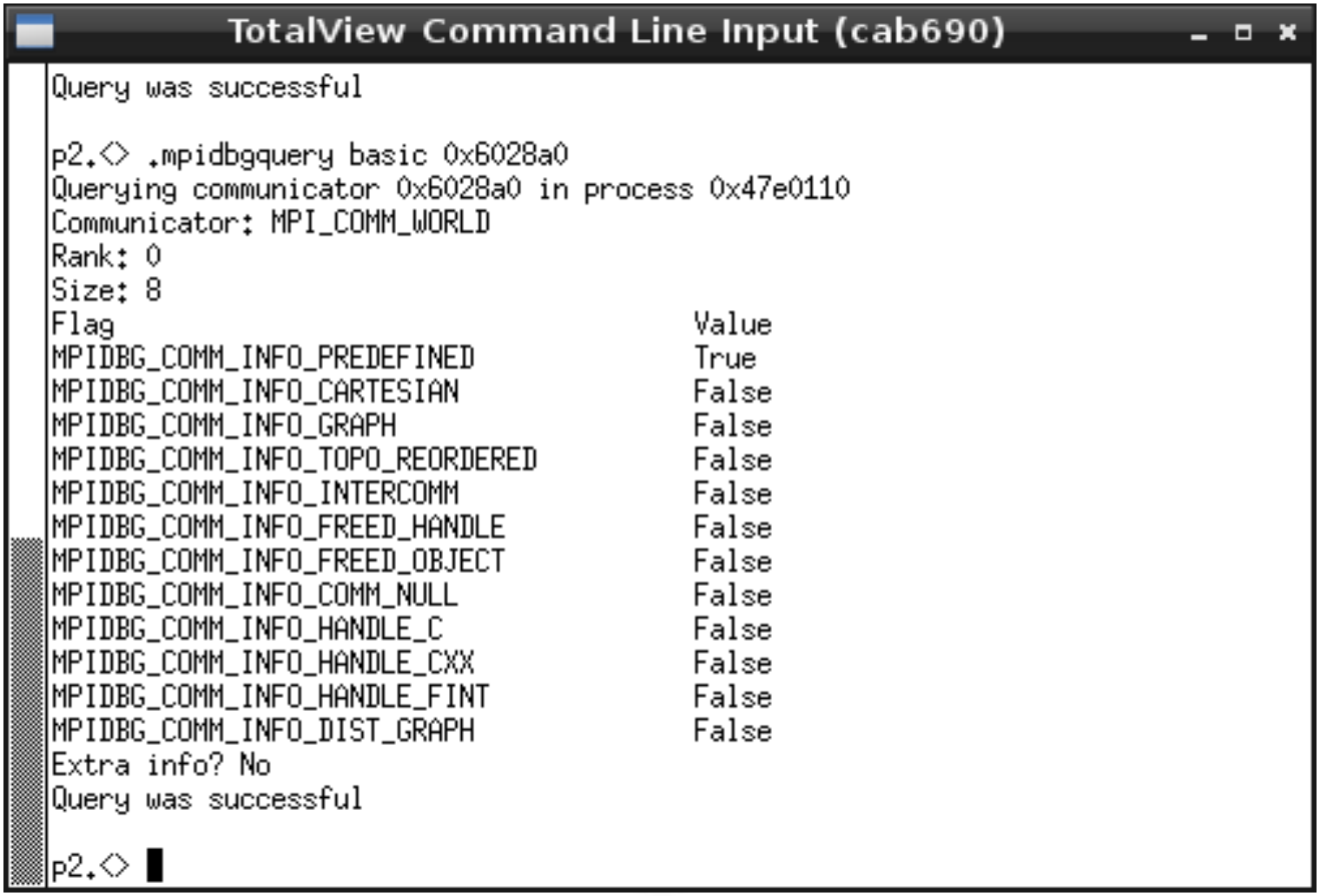
Demonstration



Demonstration



Demonstration



Conclusions

Conclusions

- MPI handle introspection simplifies debugging of MPI related problems
- Developer gains insight into MPI runtime
- Cross MPI runtime and debugger support

Future Work

- Support for more aspects of MPI objects
  - Communicator topologies, Error Handlers, etc.
- Support TotalView's graphical interface
- Allow flexible breakpoint/watch conditions?

Thread Multiple and Asynchronous progress – take 2

University of Tennessee

LANL

Support for THREAD_MULTIPLE

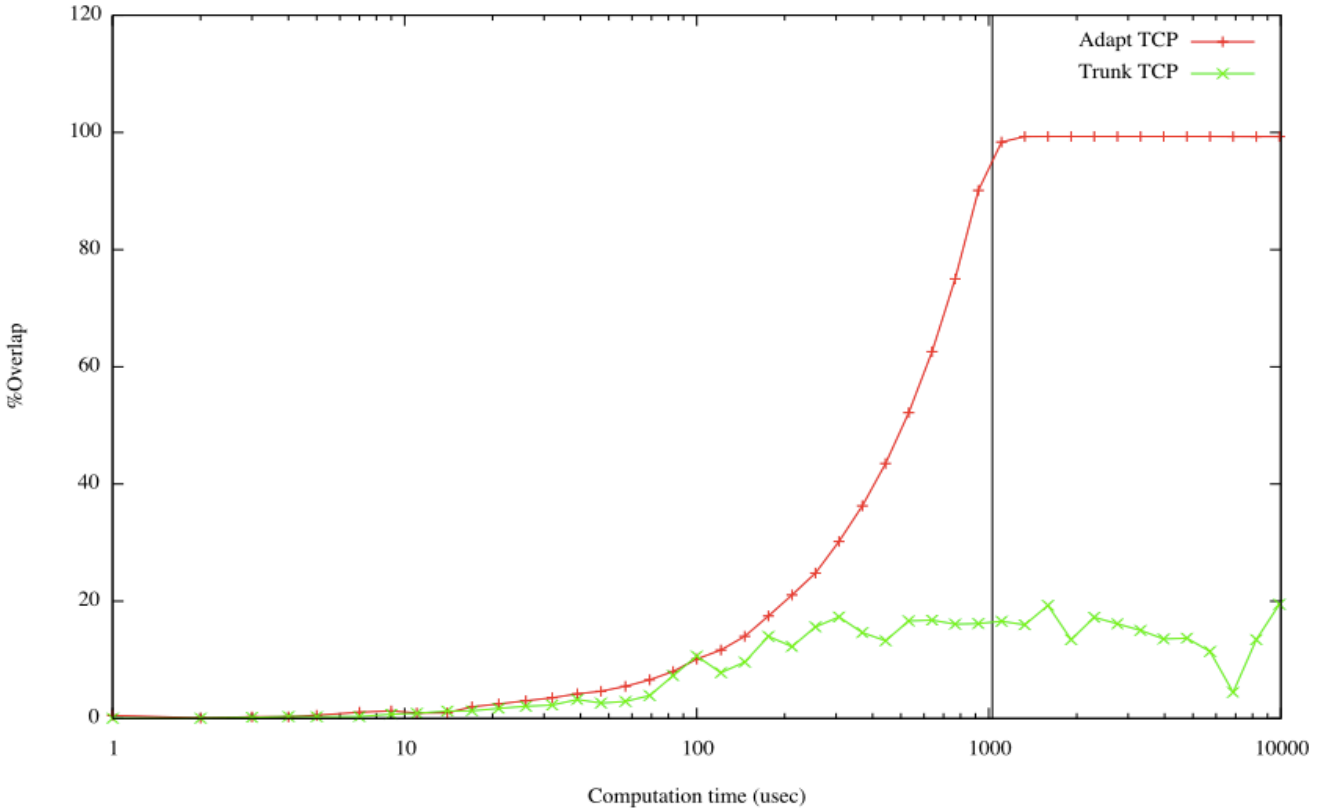
- Do we have it / Do we not have it (?)
  - We did not do it right
- Reassess the costs/benefits
  - Cost of internal objectification
  - Minimize the matching logic protection overhead
  - Define requirements for the BTL (and other components)
  - How to allow threads to collaborate while inside the library?
  - Redo the wait/test support for multiple requests

One step further

- Allow asynchronous progress
  - Major obstacle the PML
- Ongoing experimentation of the design proposal
  - UTK: BTL (tcp, sm)
  - LANL: BTL (ugni, vader)
  - ?: BTL(openib)
  - MTL work in progress

The TCP BTL (easy)

Overlap on TCP with a 128Kb message

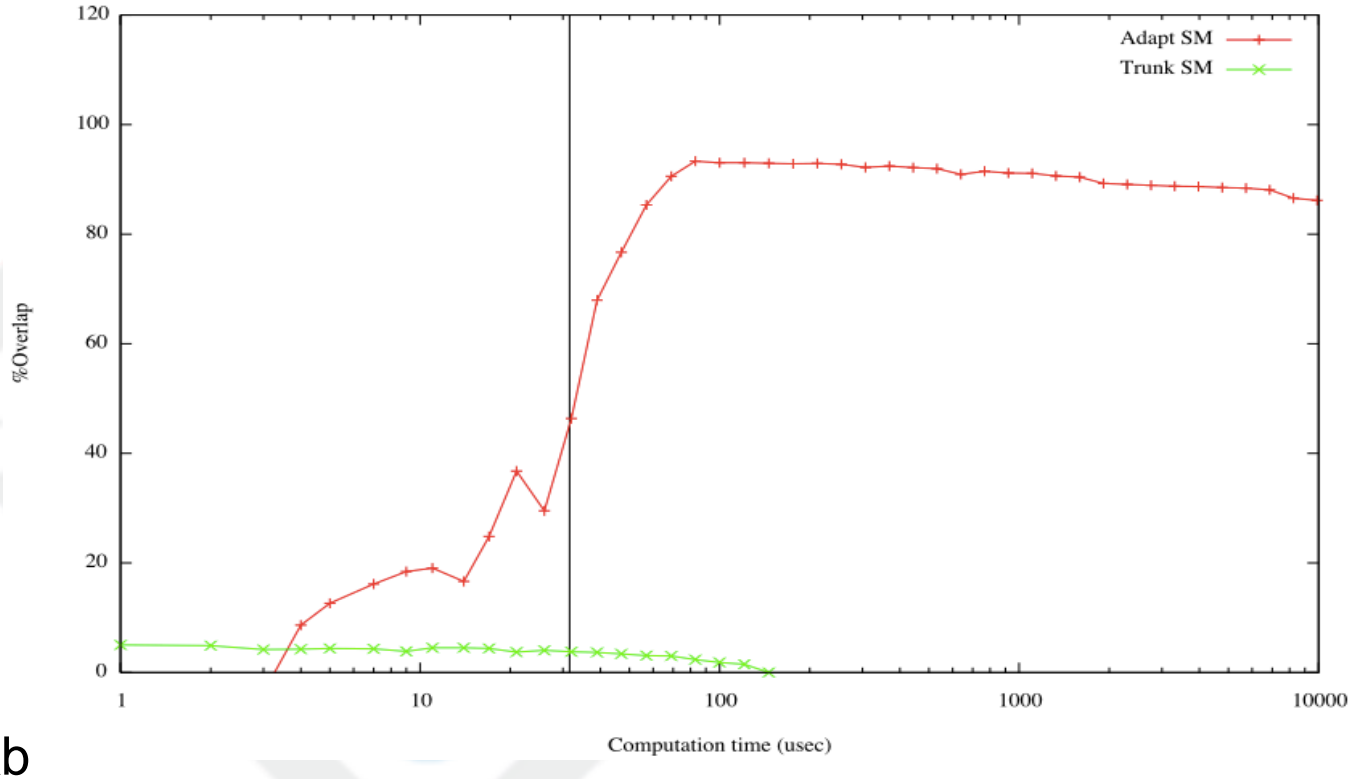


Vertical line = the time to send a 128Kb message

Sandia SMB – Host availability

A difficult BTL (SM)

Overlap on SM with a 64Kb message



Vertical line = the time to send a 128Kb message

Sandia SMB – Host availability

Fault Tolerance

University of Tennessee

Fault Tolerance @ MPI level

- User Level Fault Mitigation
  - http://fault-tolerance.org
- Provide mechanism to MPI to gracefully survive failures
  - Allow both soft and hard failures
- Gained a lot of support from the user community
- Implementation details
  - Fork of 1.6.4
  - Soon to migrate to the master

ULFM: API extensions to "repair MPI"

User Level Failure Mitigation: a set of MPI interface extensions to enable MPI programs to restore MPI communication capabilities disabled by failures

- Flexible:
  - Must accommodate all application recovery patterns
  - No particular model favored
  - Application directs recovery, pays only the necessary cost
- Performance:
  - Protective actions outside of critical communication routines
  - Unmodified collective, rendez-vous, rma algorithms
  - Encourages a reactive programming style (diminish failure free overhead)
- Productivity:
  - Backward compatible with non-FT applications
  - A few simple concepts enable FT support
  - Key concepts to support abstract models, libraries, languages, runtimes, etc

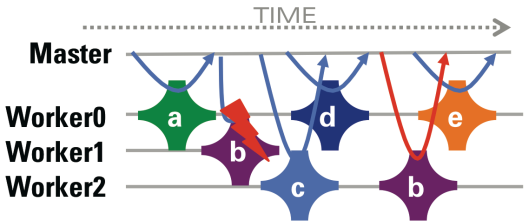
Application Recovery Patterns

Coordinated Checkpoint/Restart, Automatic, Compiler Assisted, User-driven Checkpointing, etc.

In-place restart (i.e., without disposing of non-failed processes) accelerates recovery, permits in-memory checkpoint

Naturally Fault Tolerant Applications, Master-Worker, Domain Decomposition, etc.

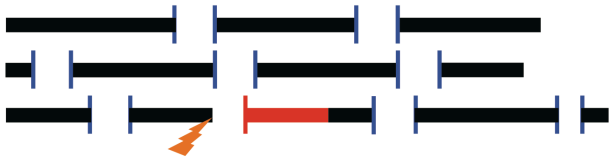
Application continues a simple communication pattern, ignoring failures



ULFM MPI Specification

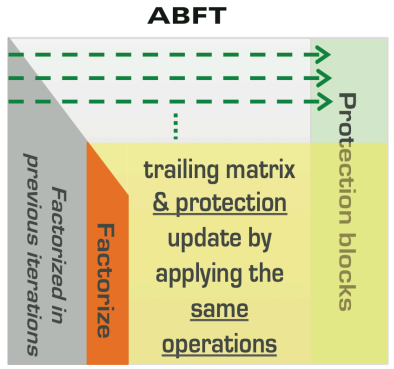
Uncoordinated Checkpoint/Restart, Transactional FT, Migration, Replication, etc.

ULFM makes these approaches portable across MPI implementations



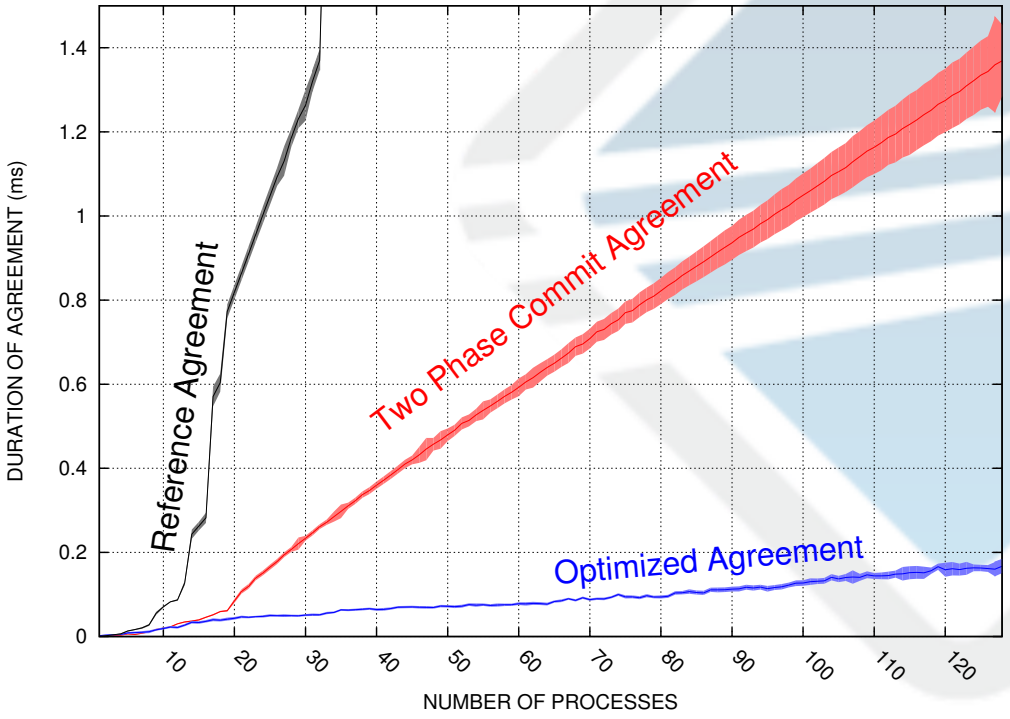
Algorithm Fault Tolerance

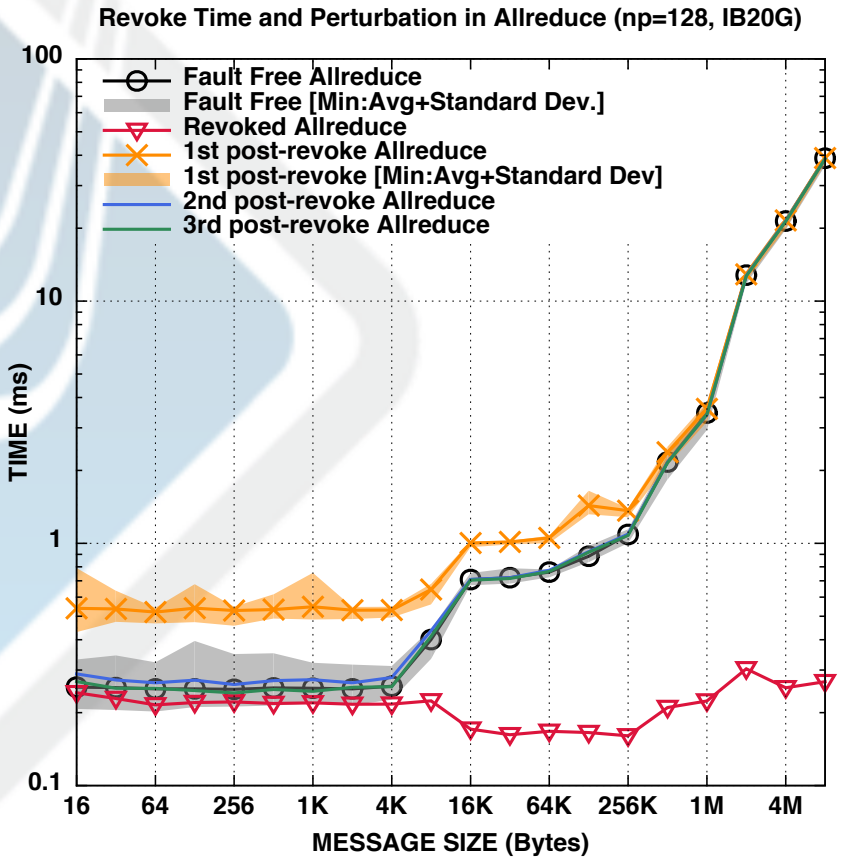
ULFM allows for the deployment of ultra-scalable, algorithm specific FT techniques.



Revoke & Agreement

- Cost in Log(n)





Cisco work

Jeff Squyres

Cisco Systems, Inc.

OFIWG / libfabric

- "Next generation verbs API"
  - Being developed by the OpenFabrics Interfaces Working Group (OFIWG)
  - Anyone can participate
- Charter:
  - *Develop an extensible, open source framework and interfaces aligned with upper-layer protocols and application needs for high-performance fabric services.*
- http://ofiwg.github.io/libfabric/

libfabric API

- Linux implementation of the OFIWG APIs
  - Design documents: man pages
  - Implementation for Linux
  - https://github.com/ofiwg/libfabric
- Similar structure to Linux Verbs API
  - Core + “provider” plugins for specific hardware
- Different focus than Linux Verbs API
  - Hardware independent
  - App-centric (e.g., target MPI)

Cisco libfabric participation

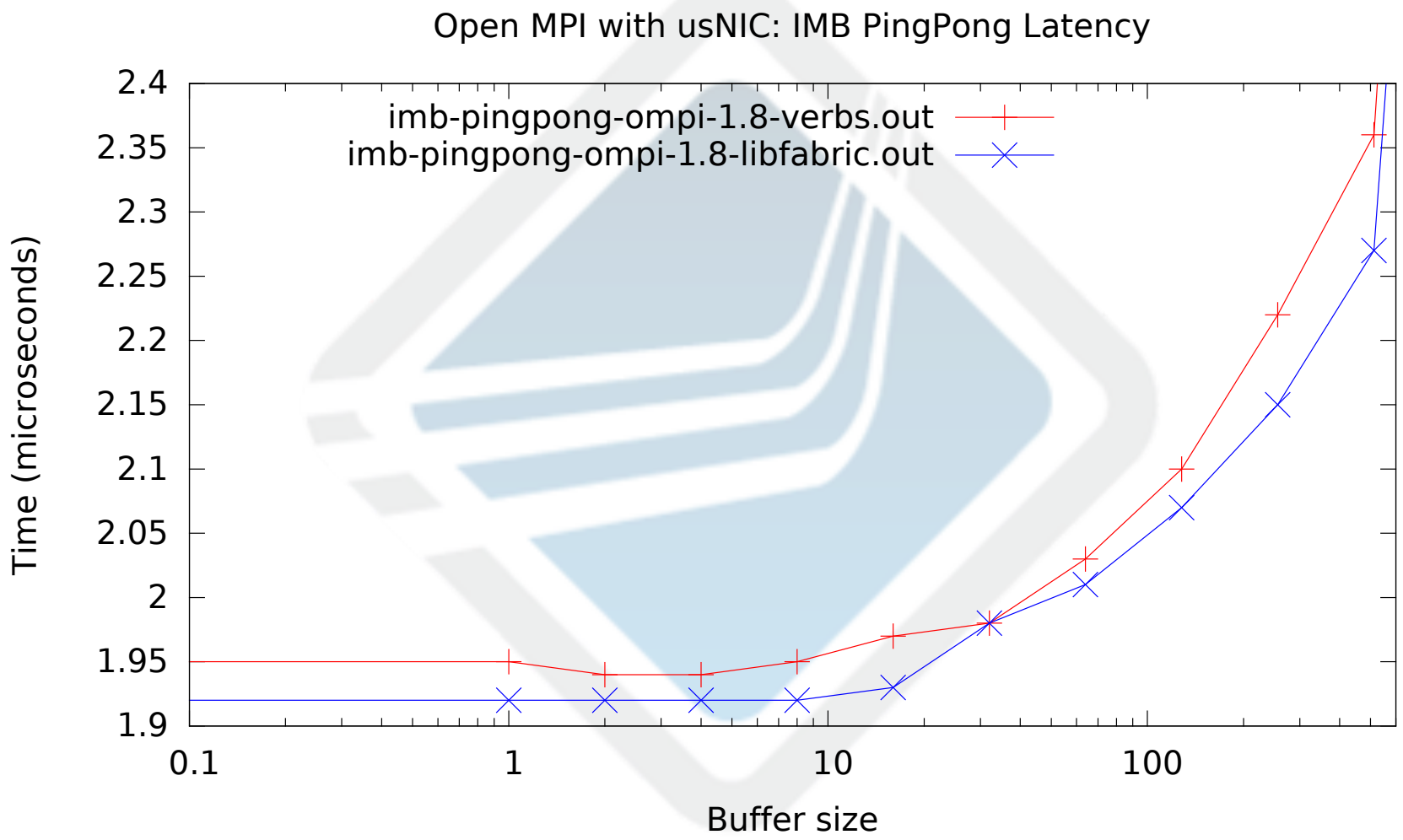
- Cisco ultra-low latency Ethernet
  - usNIC (userspace NIC)
  - Initially written to Linux Verbs
  - Now switching to libfabric

YAY!!

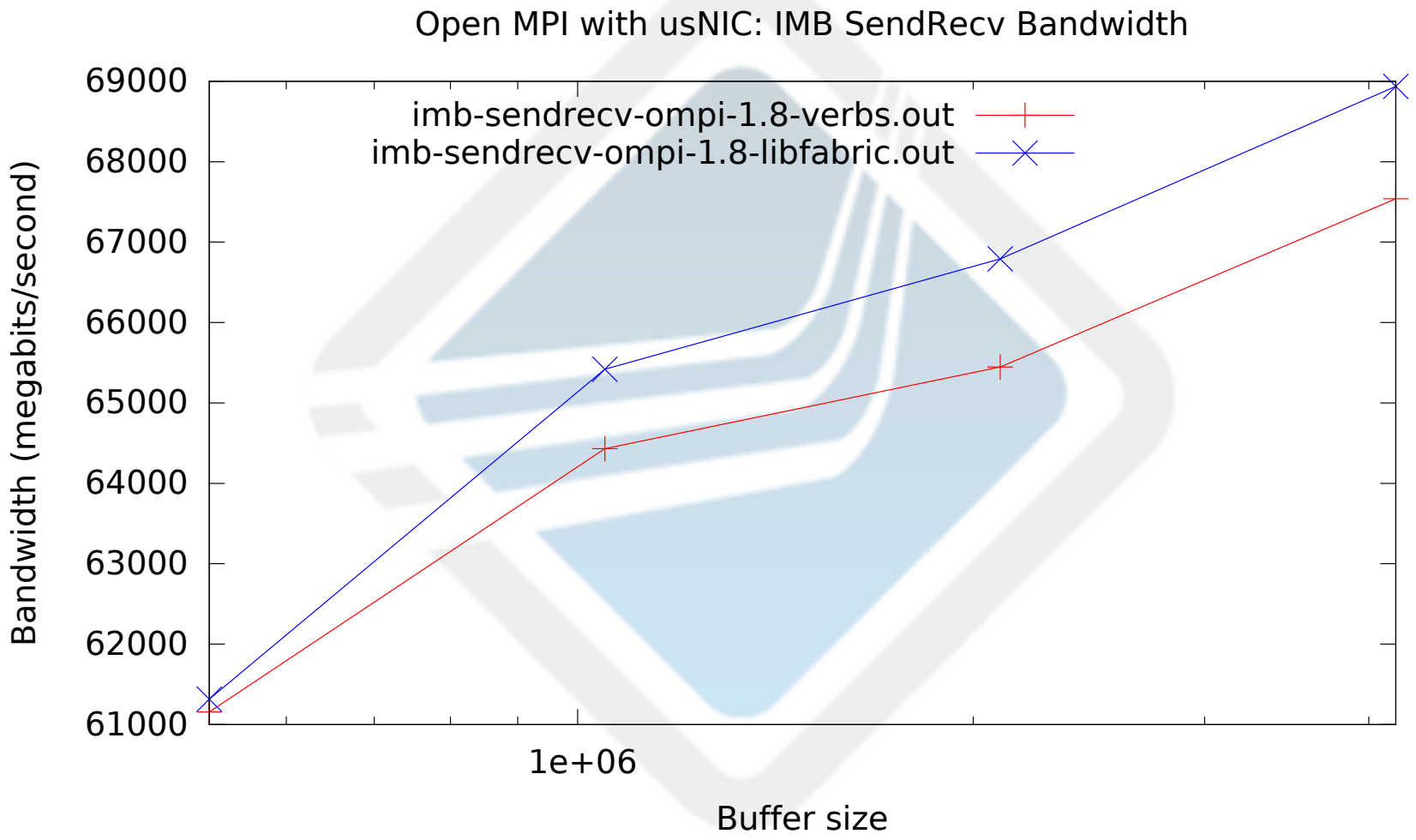
Cisco libfabric participation

- Contributed libfabric provider Oct 2014
- Published Open MPI usNIC libfabric BTL this past Monday
  - Branch on https://github.com/jsquyres/ompi
  - Still tweaking it a bit
  - Expected to go to master “soon”

usNIC performance
Verbs vs. Libfabric (latency)



usNIC performance
Verbs vs. Libfabric (bandwidth)



Picture says it all

Public service announcement

STOP USING mpif.h!

- All modern Fortran compilers have strong "**use mpi**" Open MPI support
  - Modern = Gfortran >= v4.9
  - Modern = any other Fortran compiler

Public service announcement

Change two lines of code

```
subroutine foo
    implicit none
    include 'mpif.h'

    integer :: a
    …
```

```
subroutine foo
    use mpi
    implicit none

    integer :: a
    …
```

Public service announcement

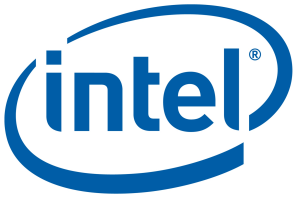
Stop the madness

Open Resilient Cluster Manager (ORCM)

Open MPI sub-project

Ralph Castain

Intel Corporation



Objectives

- Extend to exascale and beyond
  - Launch 1M procs on 50k nodes in ≤ 30s thru MPI_Init (current estimate: ~20s)
  - Support minimum of 100k nodes and 10M cores
- Full featured
  - IO subsystem support (preload, burst buffers)
  - Application and environment monitoring
    - Analytics support both post-collection and distributed for in-situ algorithms
  - Fabric management (QoS, topology info)
  - Checkpoint / restart (application and binary)

Objectives

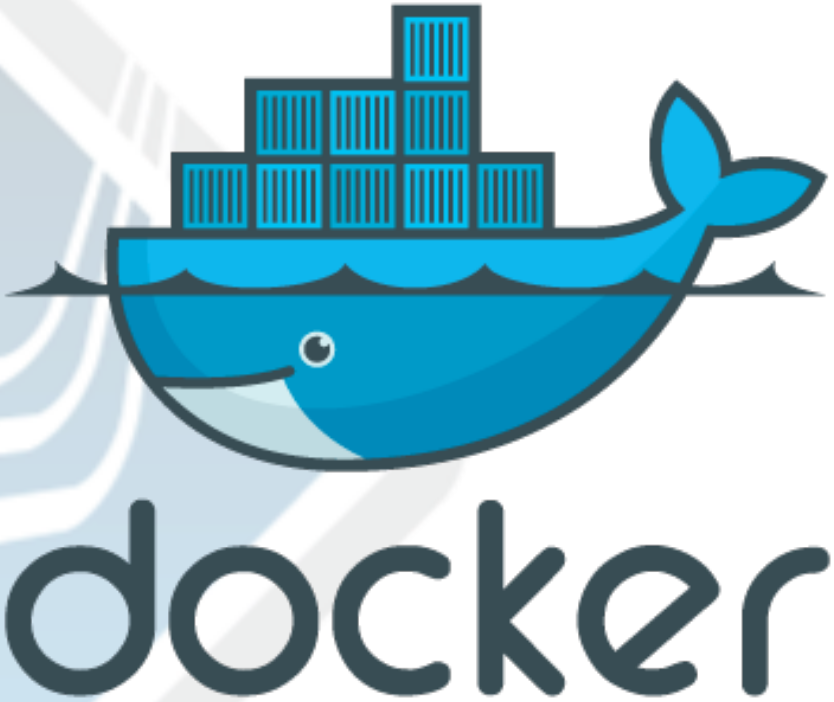
- Power management
  - Cluster power cap, job power specifications
  - Idle power, auto power-off configurations
- Resilient
  - Failover across fabrics, across routes
- Plug-in architecture
  - BSD licensed (Open MPI subproject)
  - Support proprietary plugins
  - On-the-fly updates

Contribute or follow along!

- https://github.com/open-mpi/orcm/wiki
- Interested in learning more, beta testing, or contributing?
- Mail Ralph Castain (Intel)
  - rhc@open-mpi.org

Fun fact

- ORCM developers use Docker to simulate giant clusters
  - 4 physical servers
  - 512 simulated servers
- Docker FTW!



Where do we need help?

- Code
  - Soon: MPI_THREAD_MULTIPLE testing
  - Soon: Asynchronous progress testing
  - …any bug or feature that bothers you
- Release engineering
- ***User documentation***
- Usability
- Testing

Researchers: how can we help you?

- Fork OMPI on GitHub
- Ask questions on the devel list
- Come to Open MPI developer meetings
  - Next: January 27-29, 2015, Dallas, TX, USA
- Generally: be part of the open source community

Questions?

Come Join Us!

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

CISCO

ICL UT

Los Alamos NATIONAL LABORATORY EST. 1943

DTU