Open MPI State of the Union Community Meeting SC '13

November 19, 2013

Jeff Squyres

George Bosilca

Brice Goglin

Mike Dubman

Connect. Accelerate. Outperform.™

Open_MPI_Init()

```
shell$ svn log https://svn.open-mpi.org/svn/ompi -r 1
------------------------------------------------------------------------
r1 | jsquyres | 2003-11-22 11:36:58 -0500 (Sat, 22 Nov 2003) | 2 lines

First commit
------------------------------------------------------------------------
shell$
```

Open_MPI_Current_status()

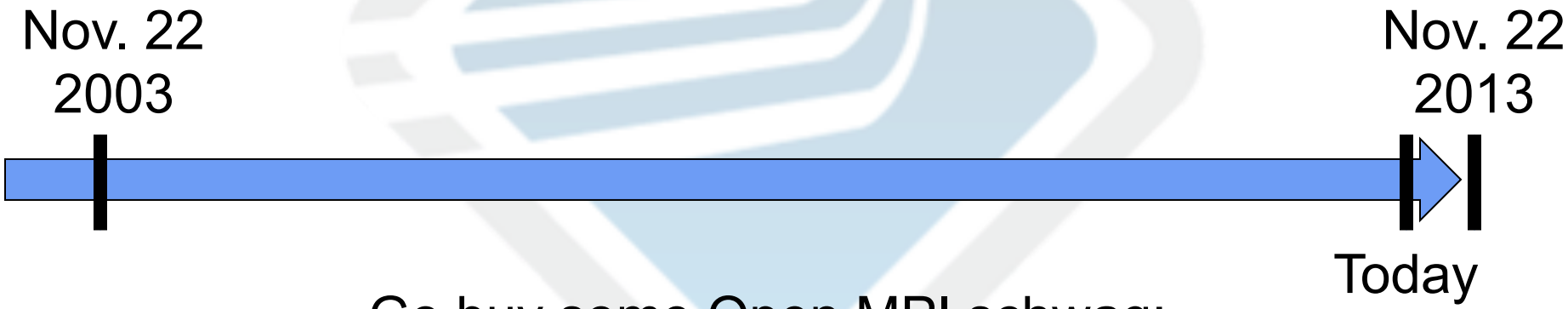
```
shell$ svn log https://svn.open-mpi.org/svn/ompi -r HEAD
------------------------------------------------------------------------
r29720 | vasily | 2013-11-19 00:00:21 -0700 (Tue, 19 Nov 2013) | 2 lines

"lf" statement wrapping with #if MEMORY_LINUX_UMMUNOTIFY in order to prevent ptmalloc2 hooks disabling in case if OMPI was not configured with ummunotify support.
------------------------------------------------------------------------
shell$
```

10 years of Open MPI!

I declare November 22, 2013 to be

Open MPI Day



Go buy some Open MPI schwag:
cafepress.com/openmpi

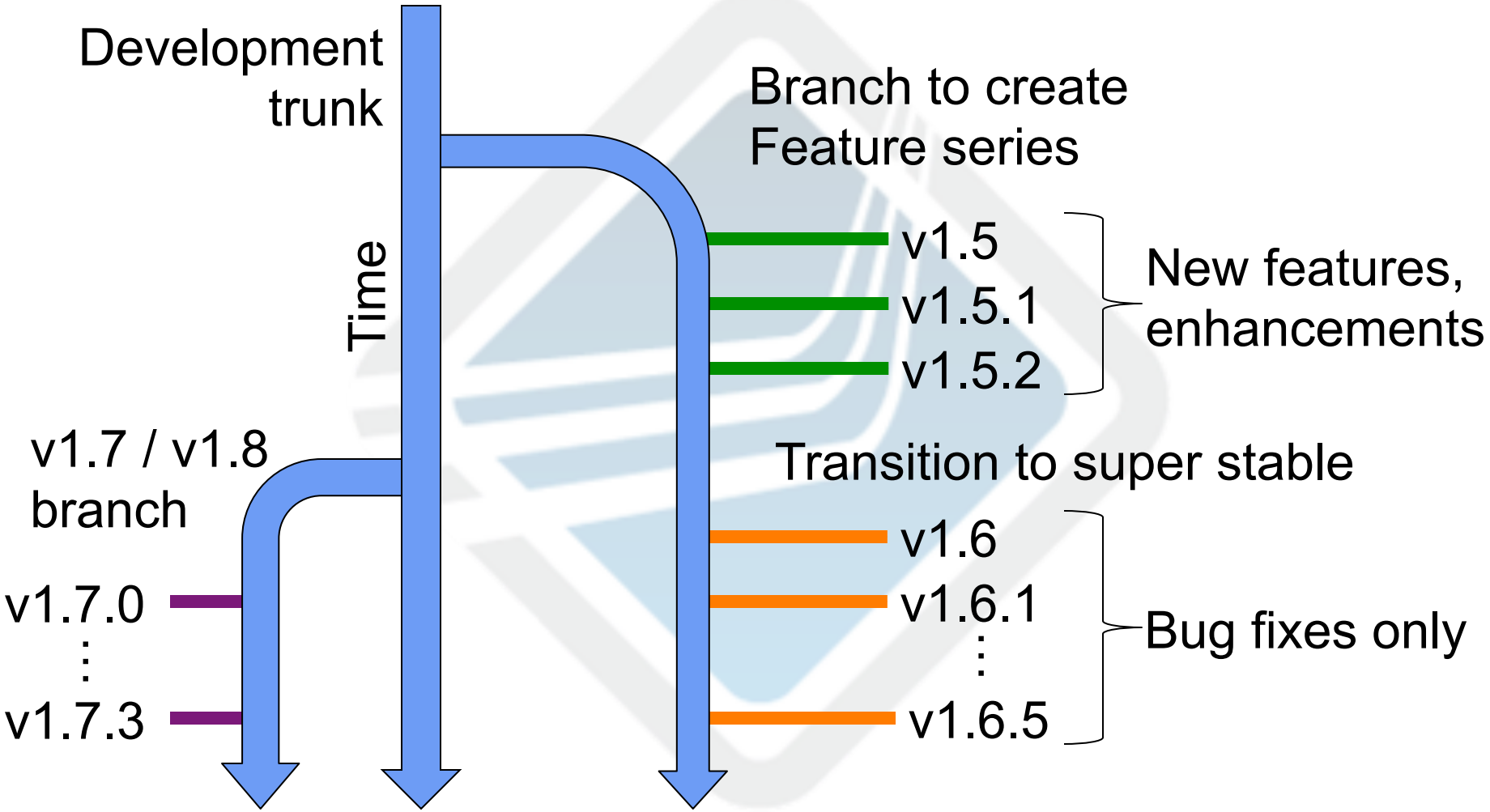
Open MPI 2014 membership

13 members, 15 contributors, 2 partners

Versioning scheme

- Open MPI has 2 concurrent release series
  - “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



v1.6 roadmap

- NULL

(unless someone finds a catastrophic bug, there will be no further v1.6 releases)

v1.7 Series

1.7 goals

- MPI-3[.1] full compliance
- Better resource exhaustion resilience
- Better collectives
- Improved scalability at all layers
  - Runtime, startup, memory, resources
- More transports, more offloading
- MPI_T tools interface
  - And revamp of MCA params

MPI 2.2 compliance (*)

- As of v1.7.3: done!
- Finally finished last 2.2 features
  - IN_PLACE support for ALLTOALL
  - COMM_CREATE for intercommunicators
  - MPI_Dist_graph support
  - Ordered attribute destruction on COMM_SELF
- Why the delay?
  - (Very) Few users cared about these features

MPI 3[.1] compliance

(*)

Non-blocking collectives	Done
Neighborhood collectives	v1.7.4 (already in nightly snapshots)
RMA	In progress
MPI shared memory	In progress
MPI_T tools interface	Done
Non-collective comm. create	v1.7.4 (already in nightly snapshots)
F08 bindings (beyond MPI 3.0)	Done
New datatypes	Done
Large counts	Done
Matched probe	Done

Runtime support

(*)

- OMPI layer now independent of the runtime
  - A well-defined interface between the two layers
  - Support for ORTE and PMI2 is available
  - …other runtimes are in the works…
- Startup data is now stored in internal DB
  - Transferred when needed, exposing different levels of information (local, node, global)
- ORTE asynchronous progress

(*)

MPI bindings

- C++ bindings deprecated by the MPI Forum
  - To be disabled by default in v1.9 (but still included)
- Next generation Fortran bindings
  - Can combine mpif.h, “use mpi”, and “use mpi_f08”
  - “use mpi_f08” is the best way
- C bindings updated with the const keyword
- Java bindings (next generation)
  - Full support of all MPI capabilities
  - Support for Java Direct buffers

Better accelerators support

- CUDA
  - GPU direct transfer over InfiniBand using asynchronous pipelined copies
  - Support for CUDA 6.0 (new pointer attribute)
  - Better small message latency
- Intel Xeon Phi
  - Native support for SCIF interface

Transport changes

(*)

- Support maintained only for current and future hardware
- Older hardware should maintain older Open MPI versions

Transport	1.6	1.7 BTL	1.7 MTL
Elan			
MX			
OFUD			
SCIF			
SCTP			
UDAPL			
Portals			v4
Windows Verbs			
SMCUDA			
UGNI			
usNIC			
VADER			

Transports still supported

(*)

Transport	1.6	1.7 BTL	1.7 MTL
OpenIB (OpenFabrics)			
TCP			
Shared memory			
MX			
MXM			
PSM			

Better processor / memory affinity

- Evolving hardware architectures
  - Evolving application affinity needs
- Smallest unit of affinity is hyperthread
  - “mpirun –bind-to-core” binds to all hyperthreads in a core
  - “mpirun –report-bindings” much more readable
- Probe nodes for topology at run-time
- Location Aware Mapping Algorithm
  - New / additional affinity options
  - Available starting with v1.7.1
- A NUMA-aware process mapper: mindist

(Hierarchical) Collectives communication

- Support for FCA 3.0
- Support for Mellanox HCOL
- Support for Portals 4 collectives
- Support for MPI-3 neighborhood collectives
- New general collective algorithms
  - ORNL / LANL
  - Scheduled for v1.7.4

PMI for exascale (PMIx)

- Extend PMI to support emerging exascale requirements
  - Scale to 100k+ nodes, 10M+ processes
- Fully support current PMI-1/2 interfaces
  - More scalable algorithms for distributing key-values
  - Plug-in architecture for algorithm development
- Extend APIs
  - Add support for binary payloads
  - Pack/unpack routines
  - User datatype definitions
  - Heterogeneous support
- Reduces number of required keys
  - Add non-blocking interfaces
  - Callback notification when requested data becomes available

MPI_T tools interface (*)

- Control variables
  - All MCA parameters available programmatically
  - Read, write (before MPI_INIT)
- Performance variables
  - Only a few exposed to far
  - Cisco usNIC BTL -- network statistics
  - Users: ask for what you want

MPI_THREAD_MULTIPLE (*)

- More users are asking about it
- Continues to be an elusive goal
  - …but we're working on it
  - It is *the* topic on the December Open MPI developer's meeting in Chicago
- To be blunt: we will not promise a timeline
  - (Extremely) unlikely to be before v1.8

Moar featurez (*)

- C99 enabled
- MCA parameters overhaul
  - Supports all POSIX types
  - “Levels” of MCA params (reflecting MPI_T)
- Better support for MPI dynamic processing
  - MPI_COMM_SPAWN and MPI_COMM_MERGE
  - Particularly: shared memory support on Cray
- mpirun CLI <TAB> completion
  - CLI options
  - MCA parameters (!!!)

Removed features

- Windows support
  - Lack of developer support
  - Native Cygwin builds available from Cygwin
- Fault tolerance
  - Hopefully to be put back before 1.8
- PERUSE

MPI Forum Fault Tolerance Working Group

Define a minimal set of semantics and interfaces to enable fault tolerant applications and libraries to be constructed portably

- User Level Failure Mitigation
  - **MPI Forum Fault Tolerance Working Group:** https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/FaultToleranceWikiPage
- Prototype in Open MPI is guiding proposal development
  - http://fault-tolerance.org/

Netloc (Network Locality)

Brice Goglin
Inria – Bordeaux – France
Brice.Goglin@inria.fr

Locality matters

- Inside the servers
  - You got hwloc in Open MPI about 3 years ago
  - NUMA, shared caches, I/O affinities, etc.
  - Mostly used for distributing and binding processes inside nodes

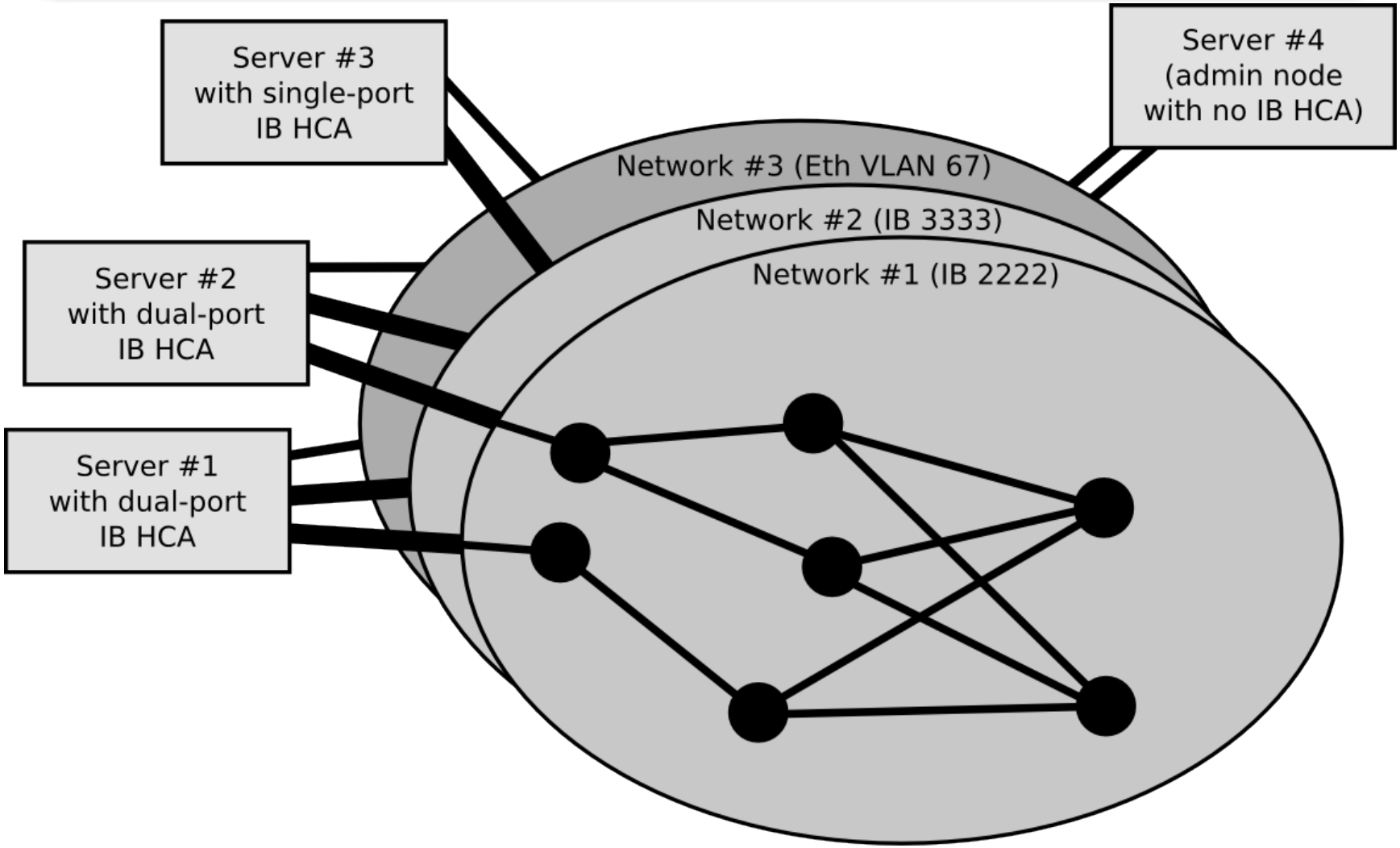
What about the network topology ?

- Papers about placing processes according to network topology
  - Reduce the distance between related processes
- Papers about adapting collective implementations to the network topology
- Distance, shared links, contention matter
  - More than inside servers ?
    - Depends on the size of the network

Introducing netloc (Network Locality)

- hwloc companion
- Takes care of network topology
- and joins hwloc and network information
  - Global « map » of your cluster
    - Connects hwloc objects to network edges
- Public API made of
  - Network queries (nodes, edges, etc.)
  - Global map queries
  - hwloc API when looking inside servers

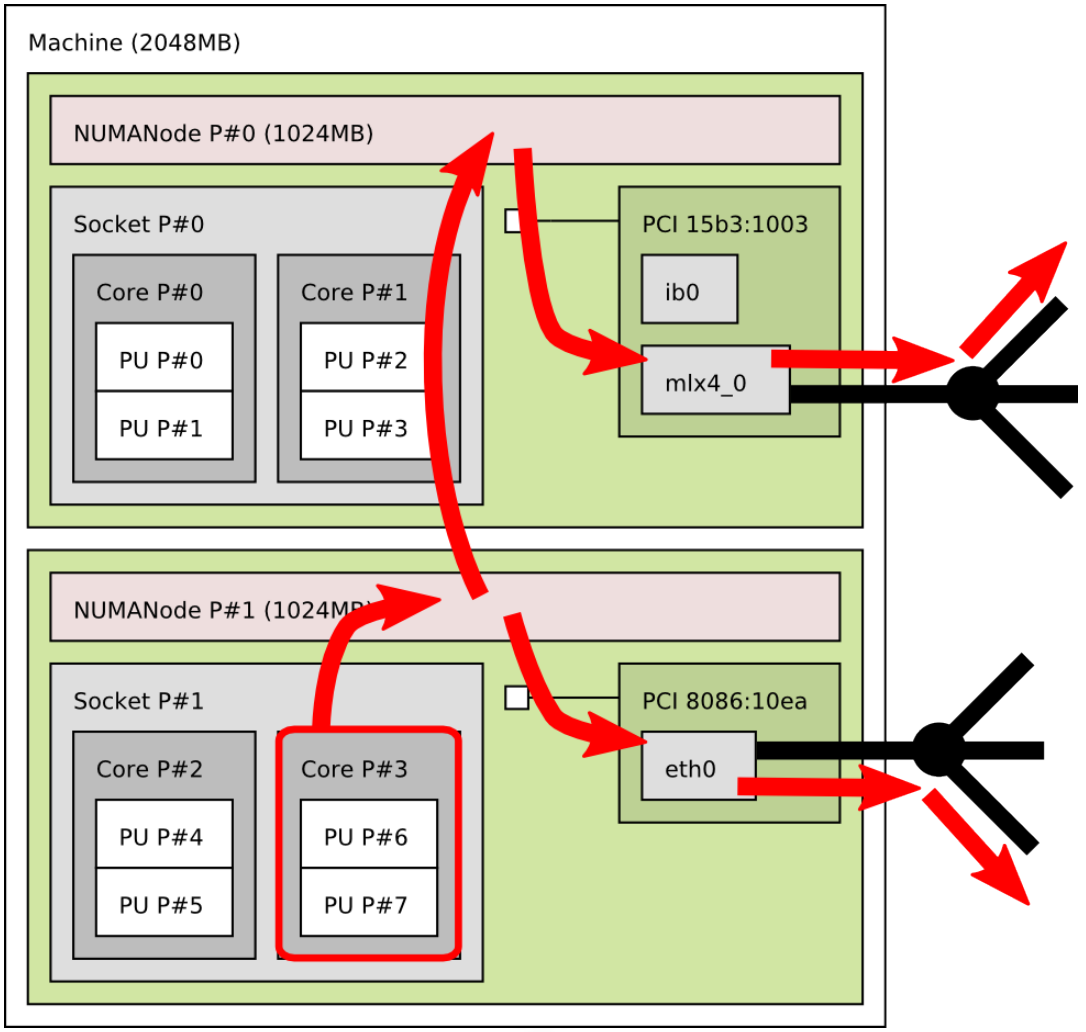
Netloc global « Map »



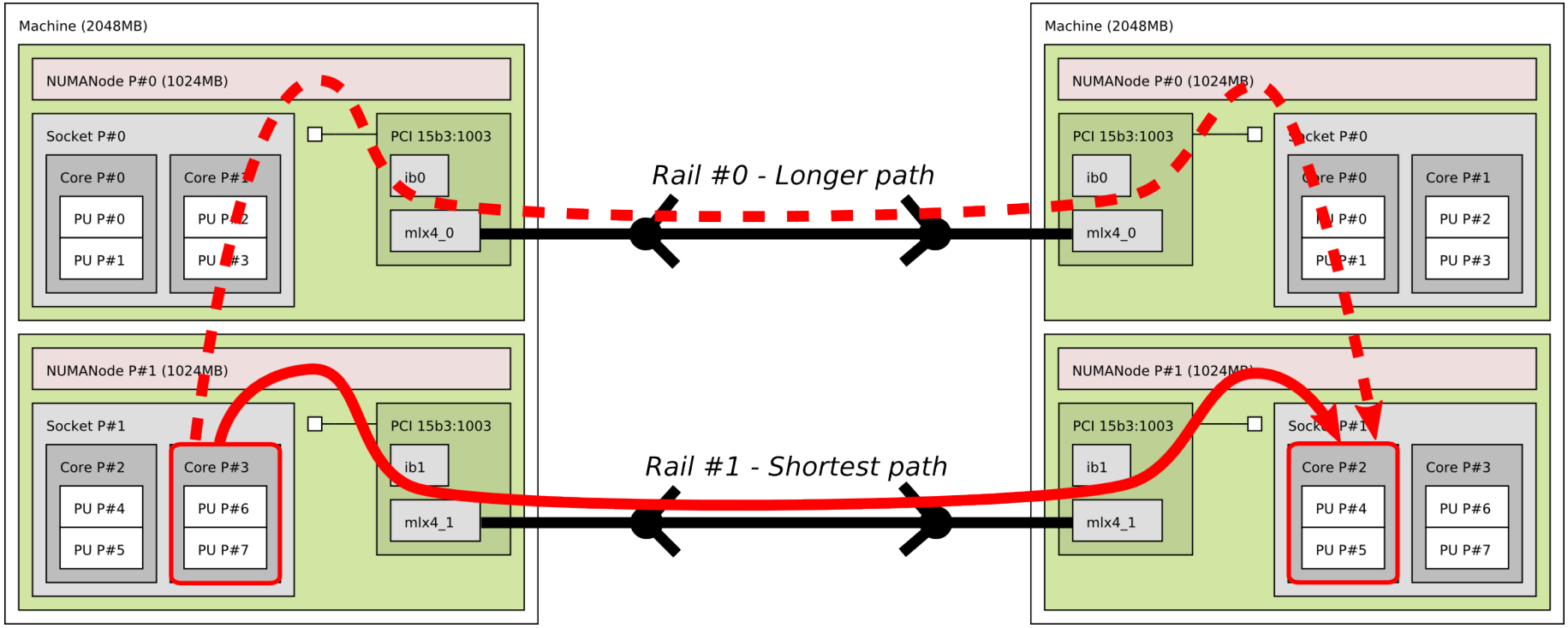
(Network) Portability

- Trying to be as generic as possible
  - More than just IB fat-trees
  - No need to run proprietary scripts anymore
- Existing backends
  - InfiniBand
  - Ethernet
    - Through OpenFlow for now
    - Maybe SNMP/LLDP for small clusters one day?
- Upcoming Cray Gemini and Aries support?

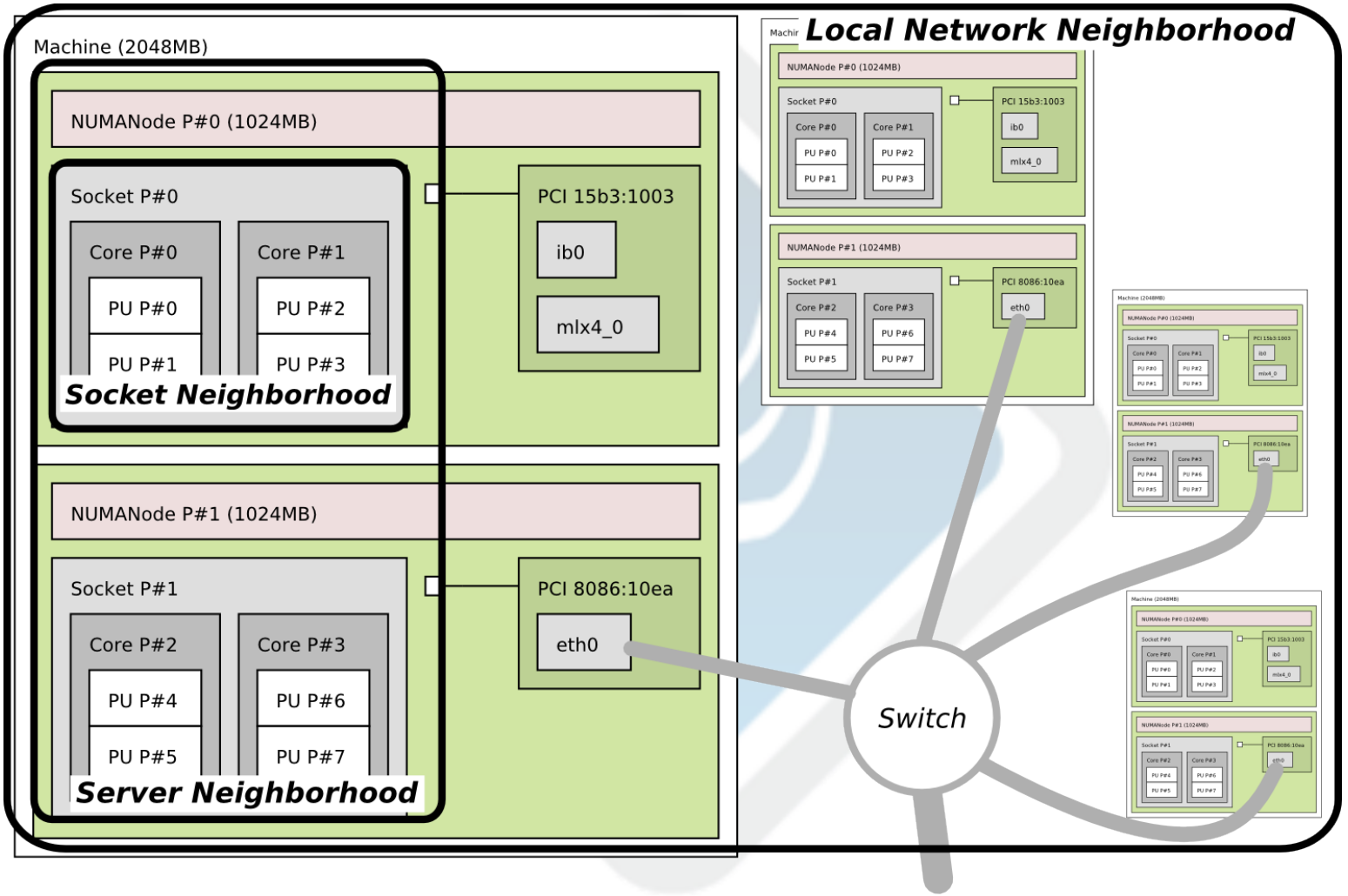
Global path across your cluster



Multirail / Multipath Locality



Hierarchy of Neighbors



Current Status

- Under discussion since SC12
- Netloc 0.5 released for SC13
- Public API not finalized yet
  - Needs users' feedback
- Written in C99
- Requires hwloc (bonus features if ≥ 1.8)
- Source code publicly available on github

Get involved !

- Currently developed by
  - University of Wisconsin-LaCrosse (J. Hursey)
  - Inria (B. Goglin)
  - Cisco (J. Squyres)
  - Under the umbrella of the Open MPI consortium
- There's a lot to do !

Need more ?

- Visit the Cisco booth: #2535
- J. Squyres gives a talk about netloc on Inria booth #2116 (Wednesday 2pm)
- See Open MPI website for links, mailing lists, etc.

Thank you!

CISCO

informatics mathematics

Inria

University of Wisconsin
LA CROSSE

OpenSHMEM in Open MPI

Mike Dubman
miked@mellanox.com

Connect. Accelerate. Outperform.™

PGAS/SHMEM

- Model to allow processes to globally share variables
- Each process to see the same variable name, but each process keeps its own copy of the variable.
- Modification to another process address space is then accomplished using put/get (or write/read) semantics.

OpenSHMEM (1)

- Some similarity with MPI:
  - SPMD
  - Atomics, collectives operations
  - one-sided operations (put/get)
  - Jobstart and runtime support (mapping/ binding/…)

OpenSHMEM (2)

- Differences from MPI
  - No communicators (yet)
  - No user-defined datatypes
  - Limited set of collectives
  - Application can put/get data from pre-allocated heap or static variables

Why in OMPI

- OMPI has very flexible architecture, easy to reuse
- OMPI built with extensibility in mind
- Many OMPI layers are MPI semantics unaware and can be reused by other parallel paradigms

OMPI + OSHMEM

- Many OMPI frameworks reused (runtime, platform support, jobstart, btl, bml, mtl, profiling, autotools)
- OSHMEM specific frameworks added, keeping MCA plugin architecture (scoll, spml, atomics, synchronization and ordering enforcement)
- OSHMEM supports Mellanox p2p and collectives accelerators (mxm, fca) as long as OMPI provided transports (tcp, openib, portals, …)

OSHMEM cheat sheet

• mpicc	→ oshcc
• mpirun	→ oshrun
• ompi_info	→ oshmem_info
• rank	→ PE
• malloc()/free()	→ shmalloc()/shfree()
• MPI_Init()	→ start_pes()
• MPI_Finalize()	→ N/A
• MPI_Send(), MPI_Put() MPI_Recv(), MPI_Get()	→ shmem_put() → shmem_get()
• User defined datatypes, basic types	→ basic types only

Quick Start

- Build & Install OpenSHMEM

% wget http://www.open-mpi.org/nightly/trunk/openmpi-1.9a1r29419.tar.gz

% tar zxvf openmpi-1.9a1r29419.tar.gz

% cd openmpi-1.9a1r29419

% ./configure --with-oshmem --prefix=$PWD/install && make install

- Build example:

% $PWD/install/bin/oshcc –o oshmem_hello $PWD/example/oshmem-hello.c

- Run example:

% $PWD/install/bin/oshrun –np 4 –H node1,node2 $PWD/oshmem_hello

Thank You!

Connect. Accelerate. Outperform.™

Where do we need help? (*)

- Code
  - MPI 3 one-sided (this is complex)
  - Fault tolerance revival
  - …any bug or feature that bothers you
- Release engineering
- User documentation
- Usability
- Testing

Researchers: how can we help you? (*)

- Fork OMPI on Bitbucket or Github
  - Upstream is still SVN
- Ask questions on the devel list
- Come to Open MPI developer meetings
- Generally: be part of the open source community

Come Join Us!

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

CISCO

ICL UT

informatics mathematics Inria

Mellanox TECHNOLOGIES

Connect. Accelerate. Outperform.™