Open MPI on the RWTH Systems

Hristo Iliev

aiXcelerate 2017 // Aachen // 5-7.12.2017

IT Center | RWTH AACHEN UNIVERSITY

What is Open MPI?

- An open-source implementation of MPI-3.1
  https://www.open-mpi.org
  https://github.com/open-mpi/ompi

- Multiple versions
  - 1.10.x and 2.0.x – discontinued
  - 2.1.x – maintenance mode
  - 3.0.x – current version

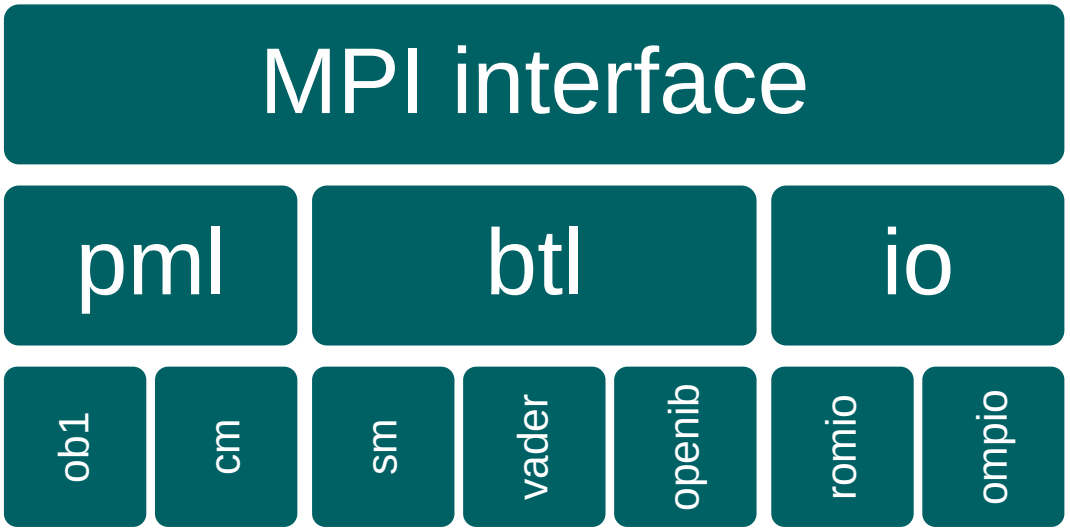
- Base of some other (commercial, vendor-specific) MPI implementations
  - Sun HPC Cluster Tools (R.I.P.)
  - Bullx MPI
  - IBM Spectrum MPI

- The default MPI library on the RWTH compute systems
  - We still use 1.10.x and consider 2.x+ as too broken for production use
  - Newer versions are available in the BETA group of modules

Internal Design

Modular Component Architecture (MCA)

- Built on a modular framework called MCA
  - A set of high-level frameworks with specific functionality specifications (think: interfaces)
  - Multiple components providing each framework's specifications (think: classes)
  - Runtime instances of one or more components – modules (think: class instances)
  - MCA components selected automatically at runtime based on
    - priority
    - heuristics
    - user overrides
- MCA parameters exposed to the user allow run-time fine tuning of the inner workings of Open MPI



Internal Design

Open Runtime Environment (Open RTE)

- The underlying process management and out-of-band communication is provided by the Open Runtime Environment (Open RTE / ORTE)
  - Starting and stopping MPI processes
  - Unix signal propagation
  - IO redirection
  - End-point registry
  - Name registry services
- ORTE is also built upon MCA
- The process launcher is called **`orterun`**
  - **`mpirun`** and **`mpiexec`** are symlinks to **`orterun`**
- Neither Open RTE nor MCA are MPI-specific and can be used to build other kinds of distributed computing projects

Obtaining library information

- To list all frameworks and components implementing them – `ompi_info`

```
$ ompi_info
...
          MCA installdirs: env (MCA v2.0.0, API v2.0.0, Component v1.10.4)
          MCA installdirs: config (MCA v2.0.0, API v2.0.0, Component v1.10.4)
               MCA memory: linux (MCA v2.0.0, API v2.0.0, Component v1.10.4)
                MCA pstat: linux (MCA v2.0.0, API v2.0.0, Component v1.10.4)
                  MCA sec: basic (MCA v2.0.0, API v1.0.0, Component v1.10.4)
                MCA shmem: mmap (MCA v2.0.0, API v2.0.0, Component v1.10.4)
                MCA shmem: posix (MCA v2.0.0, API v2.0.0, Component v1.10.4)
                MCA shmem: sysv (MCA v2.0.0, API v2.0.0, Component v1.10.4)
                MCA timer: linux (MCA v2.0.0, API v2.0.0, Component v1.10.4)
                  MCA dfs: test (MCA v2.0.0, API v1.0.0, Component v1.10.4)
                  MCA dfs: orted (MCA v2.0.0, API v1.0.0, Component v1.10.4)
                  MCA dfs: app (MCA v2.0.0, API v1.0.0, Component v1.10.4)
...
```

Important components

- Low-level point-to-point message transfer
  - `btl/sm` – intranode communication via shared memory
  - `btl/vader` – ditto
  - `btl/openib` – internode communication over InfiniBand and RoCE
  - `mtl/psm2` – intra- and internode communication via Intel's PSM2
- Collective communication
  - `coll/tuned` – tuned blocking collective algorithms
  - `coll/libnbc` – non-blocking collective algorithms
- One-sided communication
  - `osc/sm` – intranode shared-memory windows
  - `osc/pt2pt` – message-passing backend for internode RMA
- MPI-IO
  - `io/ompio` – Open MPI's own implementation (experimental in 1.x, incomplete)
  - `io/romio` – MPICH's high-performance implementation with special support for Lustre and other cluster file systems

Using Open MPI

It Just Works™

- The good news
  - Open MPI is pretty good at selecting the optimal set of components
  - Decent defaults for the MCA parameters
  - In theory it is enough to just execute: `mpiexec a.out [program arguments]`
    - Obtains the host list and number of processes from the DRM
    - Launches the executables
    - Determines the fastest communication paths

- The bad news
  - Not always
  - Meddling with MCA parameters necessary sometimes

- More good news
  - We've already done most of the mandatory meddling
    - Don't mess up your shell environment!

The Force Wielder vs the Highlander

- Open MPI has two major Point-to-point Messaging Layer (PML) components

- `ob1` controls various Byte Transfer Layer (BTL) components to move bytes around
  - `sm` or `vader` for intranode messaging
  - `openib` or `tcp` for internode messaging
  - multiple BTLs can be active at the same time for each physical path between any two given MPI processes

- `cm` supports various Matching Transport Layer (MTL) components for scalable communication
  - MTLs offload envelope matching to the hardware / device driver
  - only one MTL can be active
  - `psm2` for intra- and internode communication via Intel's PSM2
    - intranode via shared memory
    - internode over Omni-Path Architecture fabric

Using Open MPI

The Force Wielder vs the Highlander

- The Bull cluster has InfiniBand fabric
  - Open MPI automatically choses `ob1` + `sm` + `openib`, which is already optimal
  - OOB communication goes over TCP/IPoIB

- InfiniBand (very) fine tuning
  - `btl_openib_mtu` – InfiniBand MTU
  - `btl_openib_eager_limit` – max size for “small” messages
  - `btl_openib_rndv_eager_limit` – max size for “phase 1” fragments of large messages
  - `btl_openib_max_send_size` – same for “phase 2” fragments
  - `btl_openib_rdma_pipeline_frag_size` – same for “phase 3” fragments
  - `btl_openib_receive_queues` – specifies the list of IB receive queues

- Full list of MCA parameters obtainable from `ompi_info`:
  - `ompi_info --param btl openib --level 9`

The Force Wielder vs the Highlander

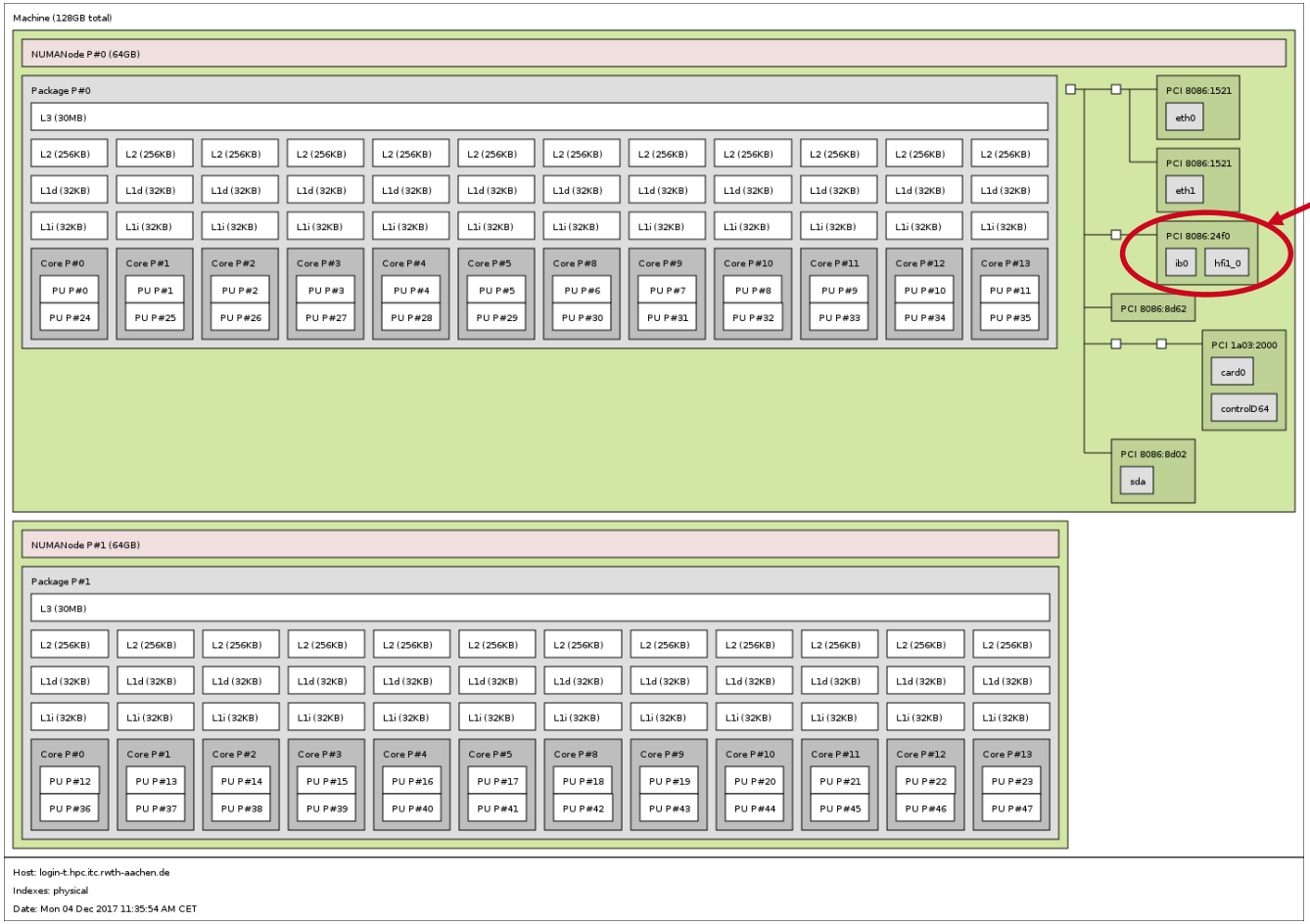
- CLAIX has Intel Omni-Path Architecture fabric
  - Open MPI automatically choses `cm` + `psm2`, which is already optimal
  - OOB communication goes over TCP/IPoIB
  - Beware the reduced tag space!

- OPA fine tuning
  - PSM2 is influenced by certain environment variables
  - `PSM2_MQ_RNDV_HFI_THRESH` – rendezvous protocol threshold for OPA (60 KB by default)
    - must not exceed the value of `PSM2_MQ_RNDV_HFI_WINDOW` (128 KiB by default)
    - larger values cause assertion failure in PSM2
  - `PSM2_MQ_RNDV_HFI_WINDOW` – rendezvous window size for OPA (undocumented!!)
  - `PSM2_MQ_RNDV_SHM_THRESH` – rendezvous protocol threshold for shared memory
  - `PSM2_MTU` – OPA HFI MTU; set it to 7 for MTU of 10 KiB

- Most PSM2 environment variables are listed in "Intel® Performance Scaled Messaging 2 (PSM2) – Programmer's Guide"

Know Your System

Process placement matters



Process placement matters

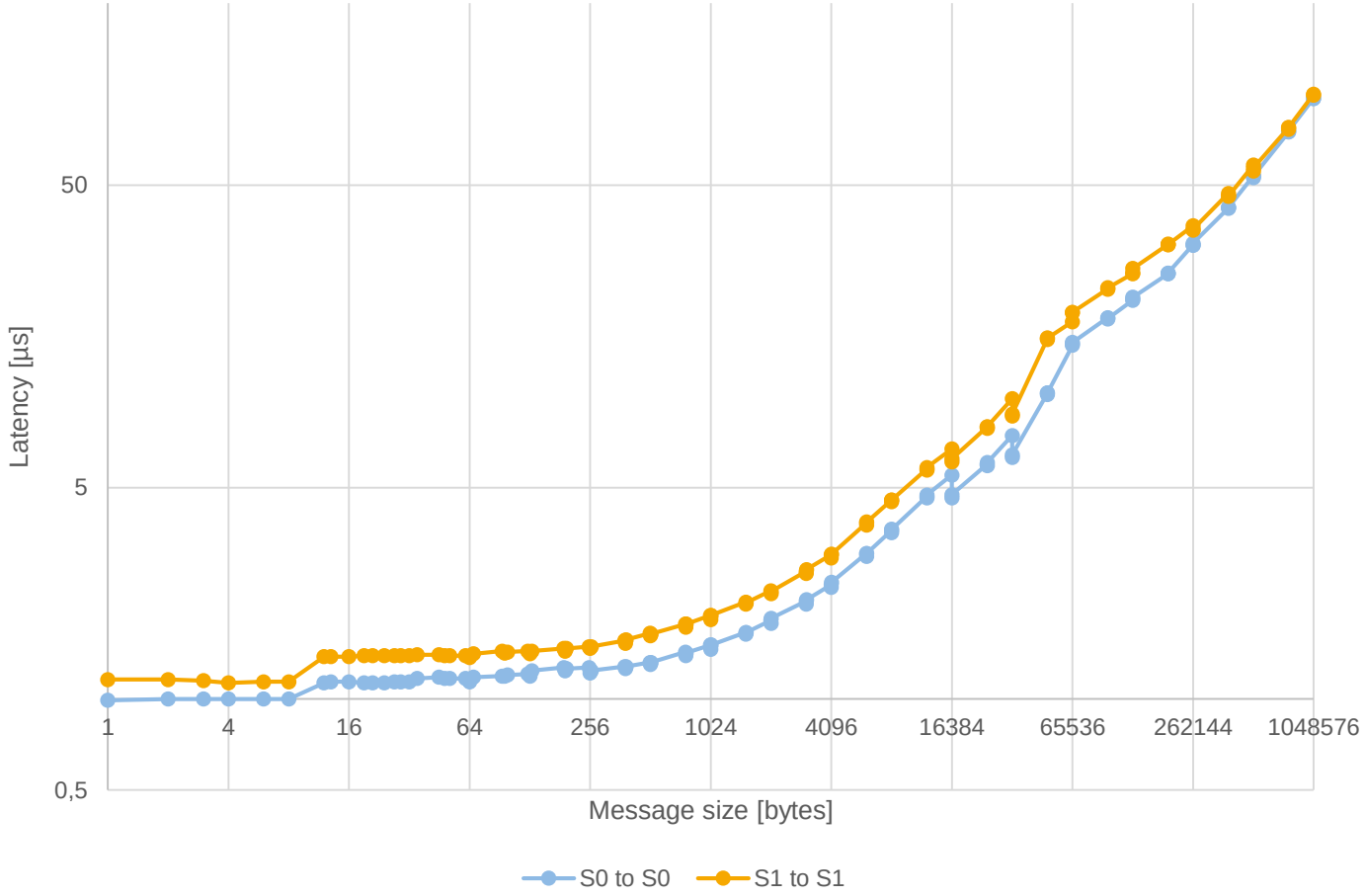
- OPA HFI-s are directly connected to one CPUs PCI-e bus
  - Data transfers from other socket(s) have to go over QPI
  - OPA is fast enough that the QPI delay becomes visible

	Socket 0	Socket 1
Socket 0	0,98 µs	1,04 µs
	91,5 Gbps	91,2 Gbps
Socket 1	1,05 µs	1,12 µs
	91,1 Gbps	90,4 Gbps

  - Effect is less visible with the QDR InfiniBand fabric on the Bull cluster

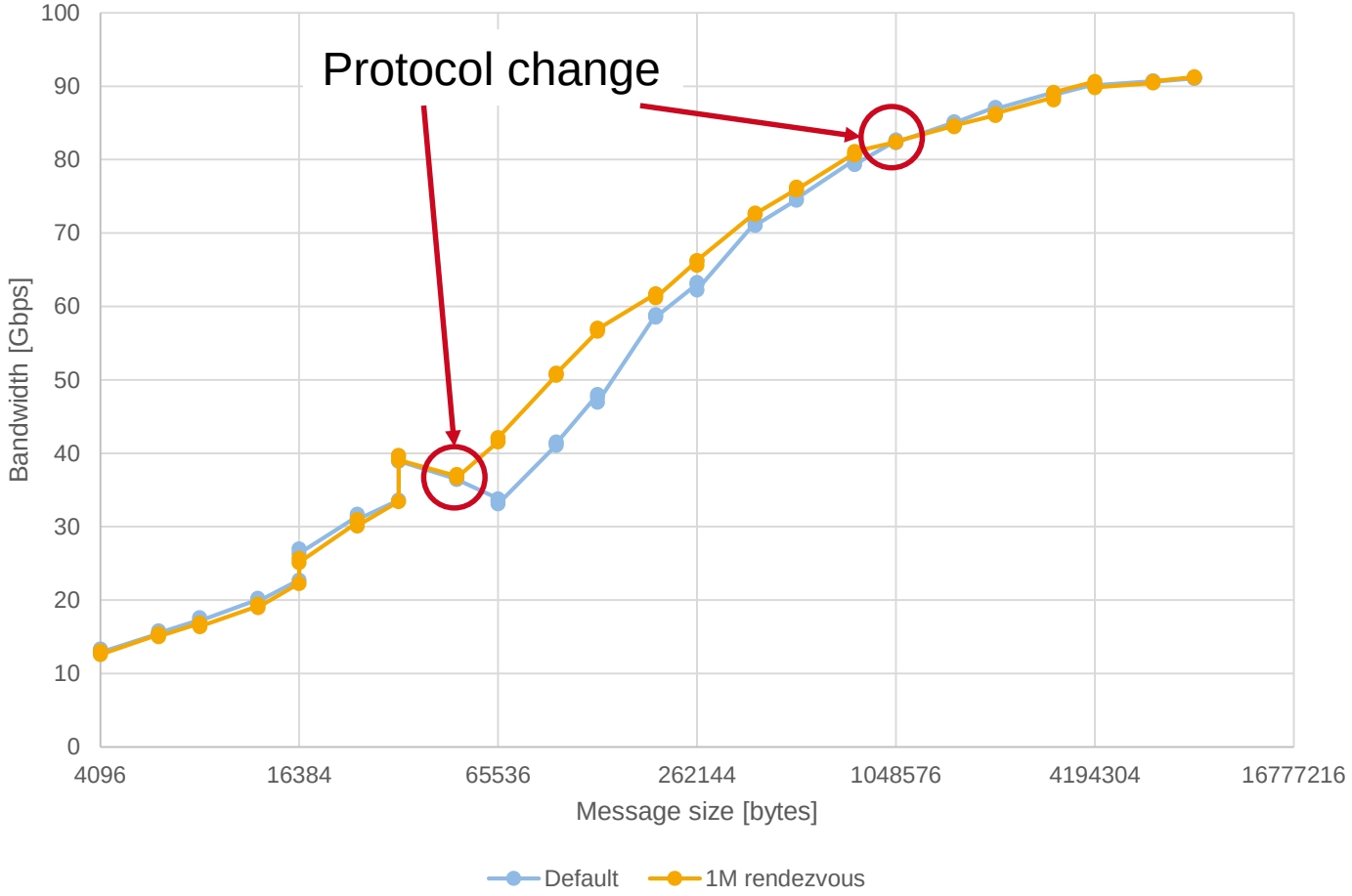
- Open MPI provides a *distance* mapper
  - `mpiexec ... --map-by dist:span --mca rmaps_dist_device hfi1_0 ...`
  - Processes are placed as close as possible to the OPA HFI (or any other PCI device specified in the `rmaps_dist_device` MCA)

Process placement matters



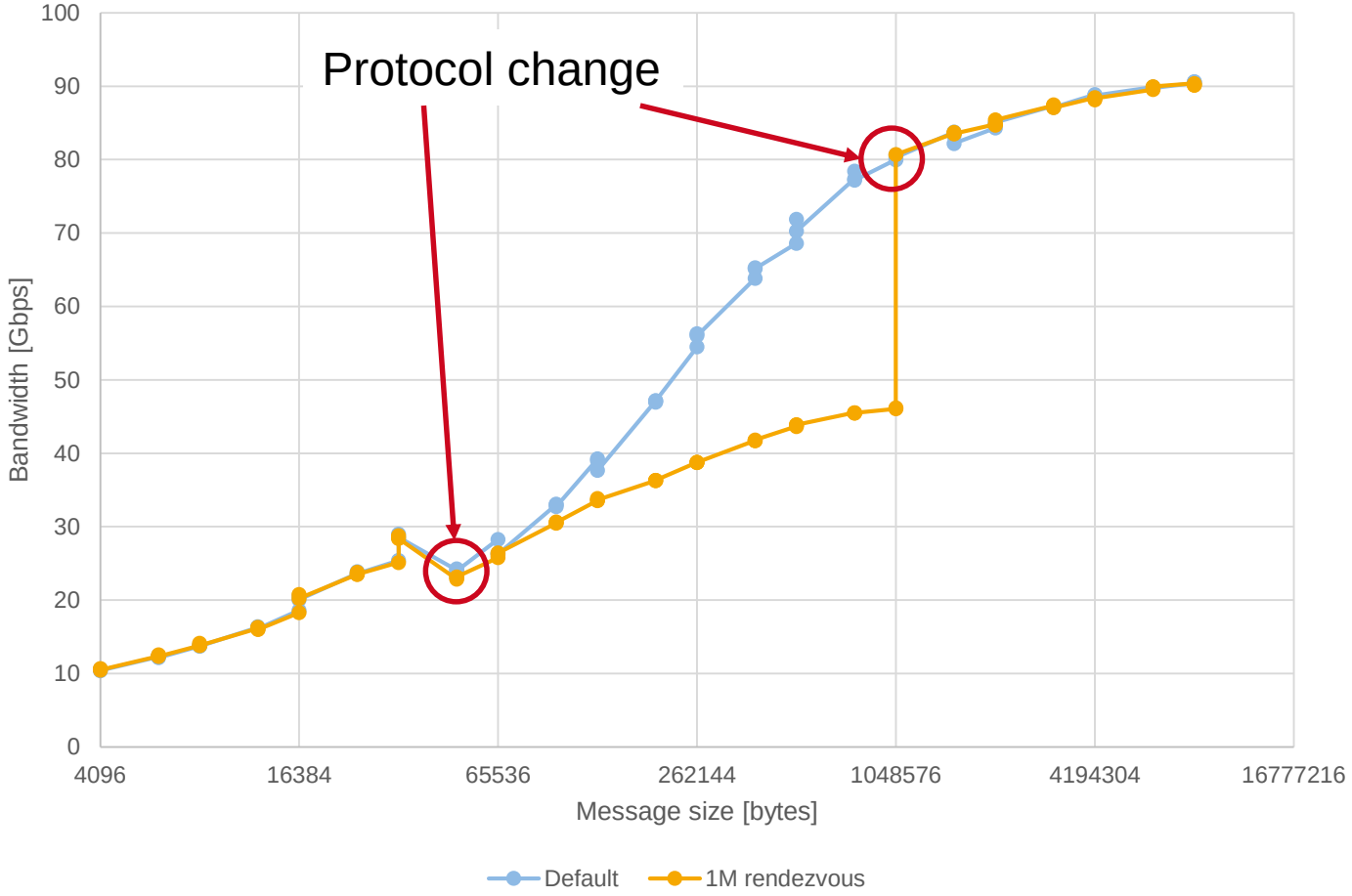
Know Your System

Socket 0 to socket 0 communication – Eager vs Rendezvous



Know Your System

Socket 1 to socket 1 communication – Eager vs Rendezvous



Instead of a Recap

Tuning recipe

- Get to know the system
  - **lstopo** for the NUMA and PCI device topology
  - **Likwid** benchmarks for the node properties (memory bandwidth, etc.)
  - **NetPIPE** or **IMB** for the network properties
- Get to know the application
  - **Score-P**, **mpiP**, etc. provide message size statistics
  - Is there a prevalent message size?
- Enable process binding
  - https://doc.itc.rwth-aachen.de/display/CC/MPI+process+binding
- Prevent fall-backs
  - Force InfiniBand: `--mca pml ob1 --mca btl self,sm,openib`
  - Force OPA: `--mca pml cm --mca mtl psm2`

Instead of a Recap

Tuning recipe

- Optimise for the specific message size
  - Message size < 1 MiB?
    - Tweak the rendezvous threshold
    - Place communicating processes close to the OPA HFI

```
--map-by dist:span –mca rmaps_dist_device hfi1_0 \
–x PSM2_MQ_RNDV_HFI_THRESH=<size> -x PSM2_MQ_RNDV_HFI_WINDOW=<size>
```

- Optimise the collective algorithms
  - Lots of MCA parameters to tweak (changed in 2.x+)

```
ompi_info --param coll tuned --level 9

--mca coll_tuned_use_dynamic_rules 1 \
--mca coll_tuned_allreduce_algorithm 3 \
--mca coll_tuned_allreduce_algorithm_segmentsize 4096 \
--mca coll_tuned_allreduce_algorithm_tree_fanout 4
```

Thank You
for Your Attention!