Bringing hardware affinity information into MPI communication strategies

Brice Goglin (and Stéphanie Moreaud)
Inria Runtime Team-Project – Bordeaux

JLPC – Rennes – 2012/06/14

Hardware is increasingly complex

Many nodes

Hierarchical interconnects between them

Multiple processors per node

Many cores per processor

Multiple levels of (shared) caches

NUMA memory

NUIOA peripherals

Outline

1. Why affinity matters and how to deal with it
2. Affinity-aware intra-node MPI
3. Affinity-aware inter-node MPI
4. Software support for managing hardware affinities
5. Conclusions

1

Why affinity matters

and how to deal with it

Two ways to deal with affinities

1. Adapt placement to affinities
   - Place tasks according to hardware/software affinity

2. Given a placement, optimize the execution
   - Adapt communication strategies to process' locality

Ideally we would do both at the same time.

Affinity-aware placing of tasks

1. Place processes when launching them
2. Reorder process' roles at runtime
   - MPI_Dist_graph_create

➢ Define an affinity metric and build a task tree
  - Amount of messages, communication volume, etc.

➢ Map the task tree onto the hardware topology tree

for MPI and other paradigms

(see F. Tessier's talk tomorrow)

Affinity-aware communication

Assuming the task placement is chosen in advance

Thread synchronization

- Hierarchical barriers

Data movement inside machines

- Explicit memory migration or implicit remote memory access in NUMA systems ?
- Map host memory in the GPU or explicit DMA transfer in CUDA ?

MPI communication (topic of this talk)

What about MPI communication?

Inside a node

- Locality of communicating processes

Between the nodes

- Locality of communicating process and NICs

2

Affinity-aware intra-node MPI

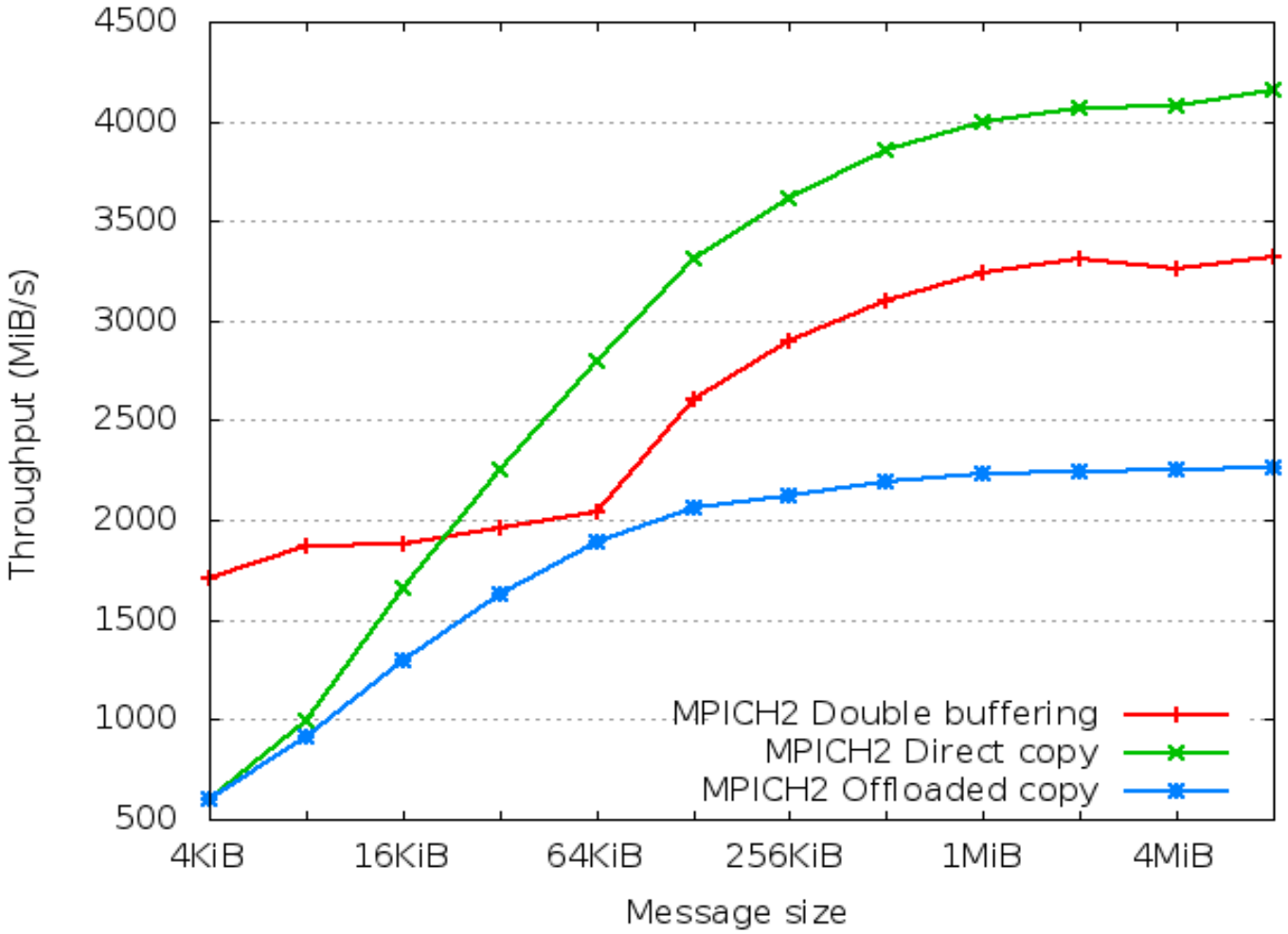
(too?) many communication strategies

1. Double-buffering across a shared memory buffer
2. Direct copy between process address-spaces
   - KNEM, CMA, etc.
   - Sender writing or receiving reading
3. Offloading to specialized copy hardware

Choose between them depending on hardware locality

- NUMA distance, shared caches, etc.

Example
Ping-pong on dual-Nehalem (shared cache)



Adapting thresholds to locality

Double-buffering likes shared caches

➢ Preferred strategy inside sockets

Double buffering does not like NUMA distance

➢ Direct copy preferred between sockets

Copy offload only useful for overlap

➢ Is the MPI call asynchronous ?

- Only if the copy hardware is close to the buffers ?

Adapting thresholds to locality (2/2)

Past collaborations with ANL (and UTK)

The hardware landscape changed a lot since then

- NUMA is everywhere

To be revived

Ongoing PhD thesis (B. Putigny) at Inria Bordeaux

- Modeling data movement performance to ease the choosing of the right strategy

3

Affinity-aware inter-node MPI

Why affinities matter for inter-node MPI

Non Uniform I/O Access (NUIOA)

- I/O chipset is close to a single socket

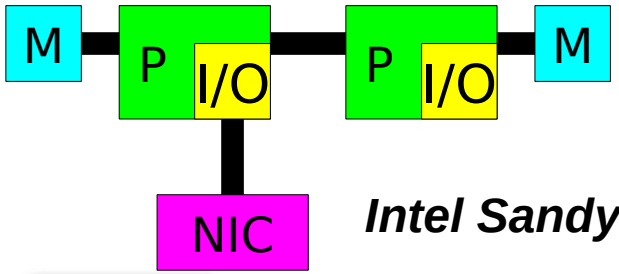
➢ Data-I/O locality affects I/O throughput and latency

Depends a lot on the architecture

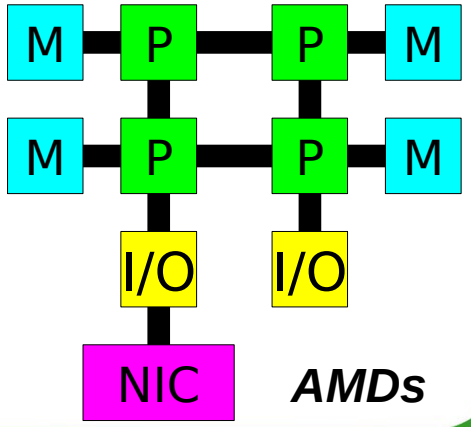
➢ Impact on DMA throughput can be asymmetrical

➢ Mostly matters for high-performance I/O

  - High-bandwidth and/or low latency



Intel Sandy-Bridge-EP



AMDs

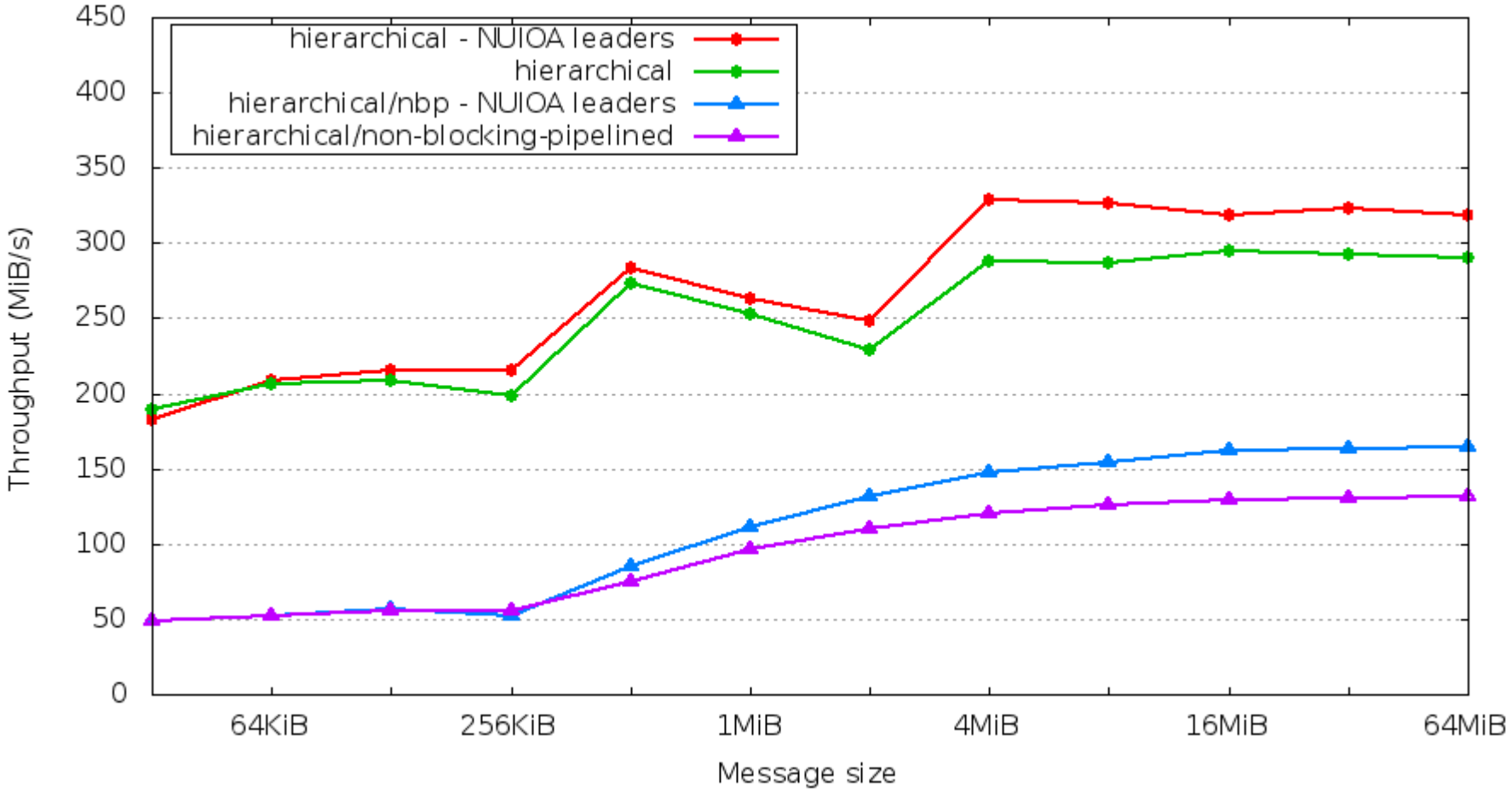
MPI vs. NUIOA

1. Try to move communication processing near the NIC
   - Collective operation leaders
   - Keep master thread near the NIC in hybrid programs

2. Use the local NIC first
   - Don't blindly use multirail
     - It stresses the internal interconnect more
     - Depends on collective patterns

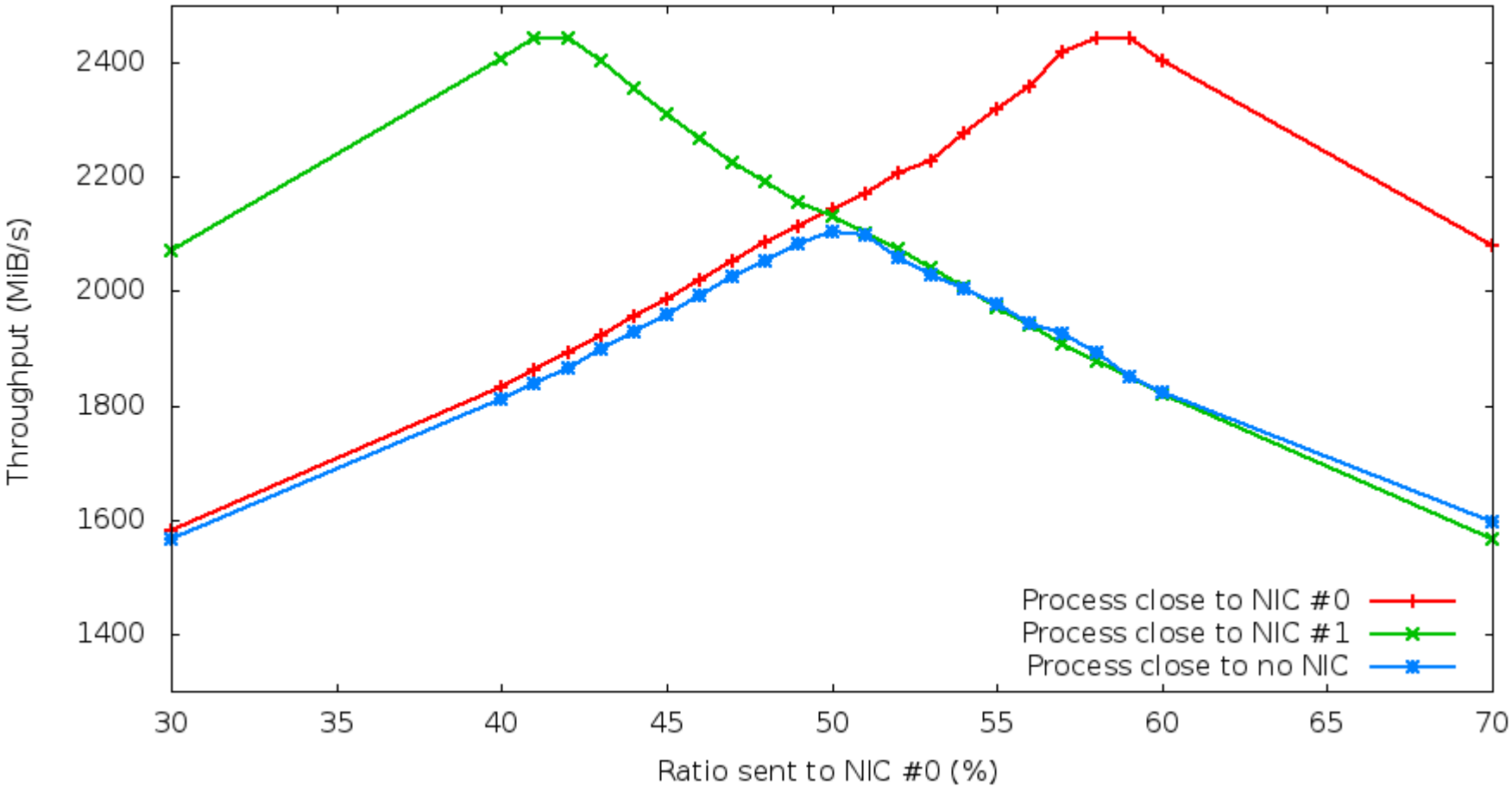
NUIOA leader election for collectives

Bcast between 8 nodes x 8 cores



NUIOA NIC selection, point-to-point

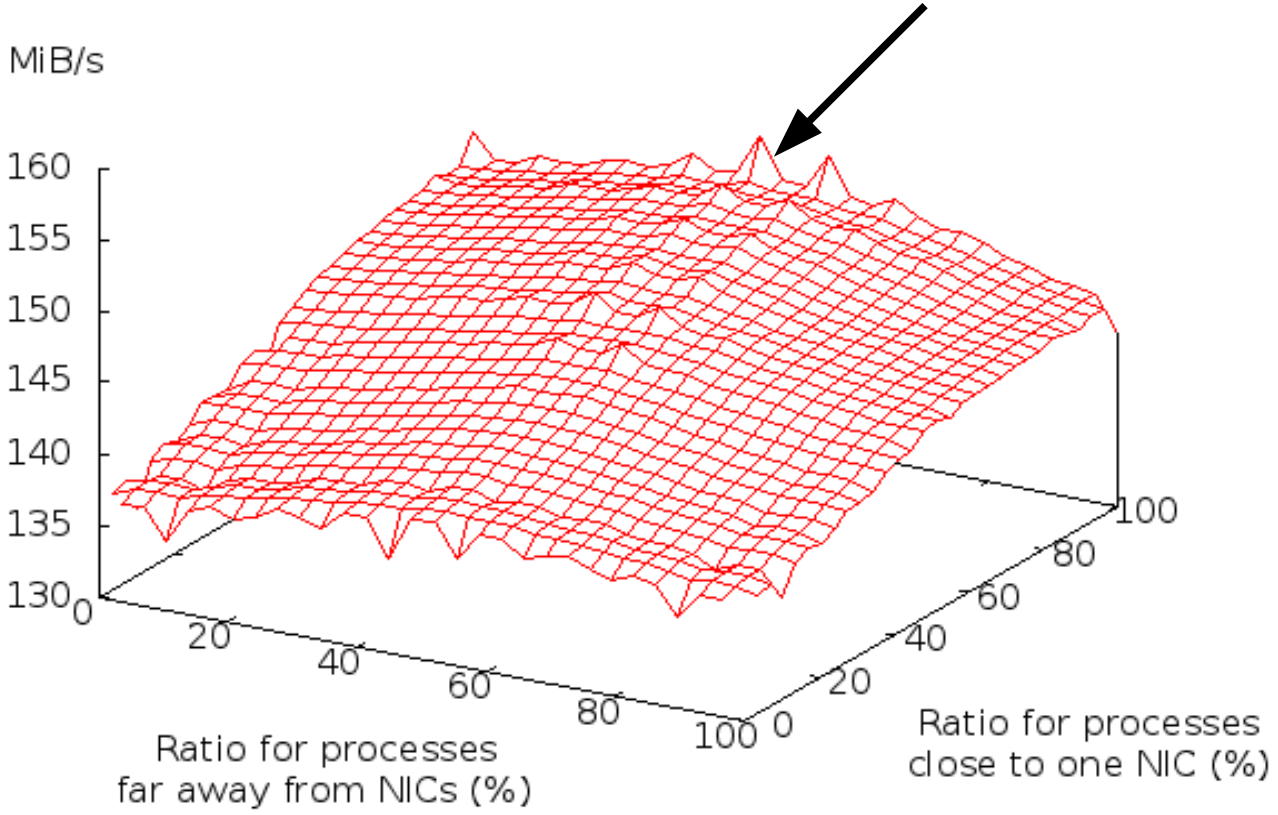
Quad-Opteron, with two IB NICs



NUIOA NIC selection, all-to-all

Quad-Opteron, with two IB NICs

Processes should only use the local NIC if any.
Otherwise, send half to each NIC.



4

Software support for managing hardware affinities

hwloc
Portable Hardware Locality

- Initially developed for affinity-based hierarchical thread scheduling at Inria Bordeaux
- Extracted as a standalone library for MPI users
- Merged with Open MPI PLPA
- Portable to many OSes
- Wide community, no serious competitor

- Now used by most MPI implementations, some batch schedulers, parallel libraries, etc.
- Mostly for binding, based on user-given policies

hwloc's view of the hardware

Tree of ressource objects

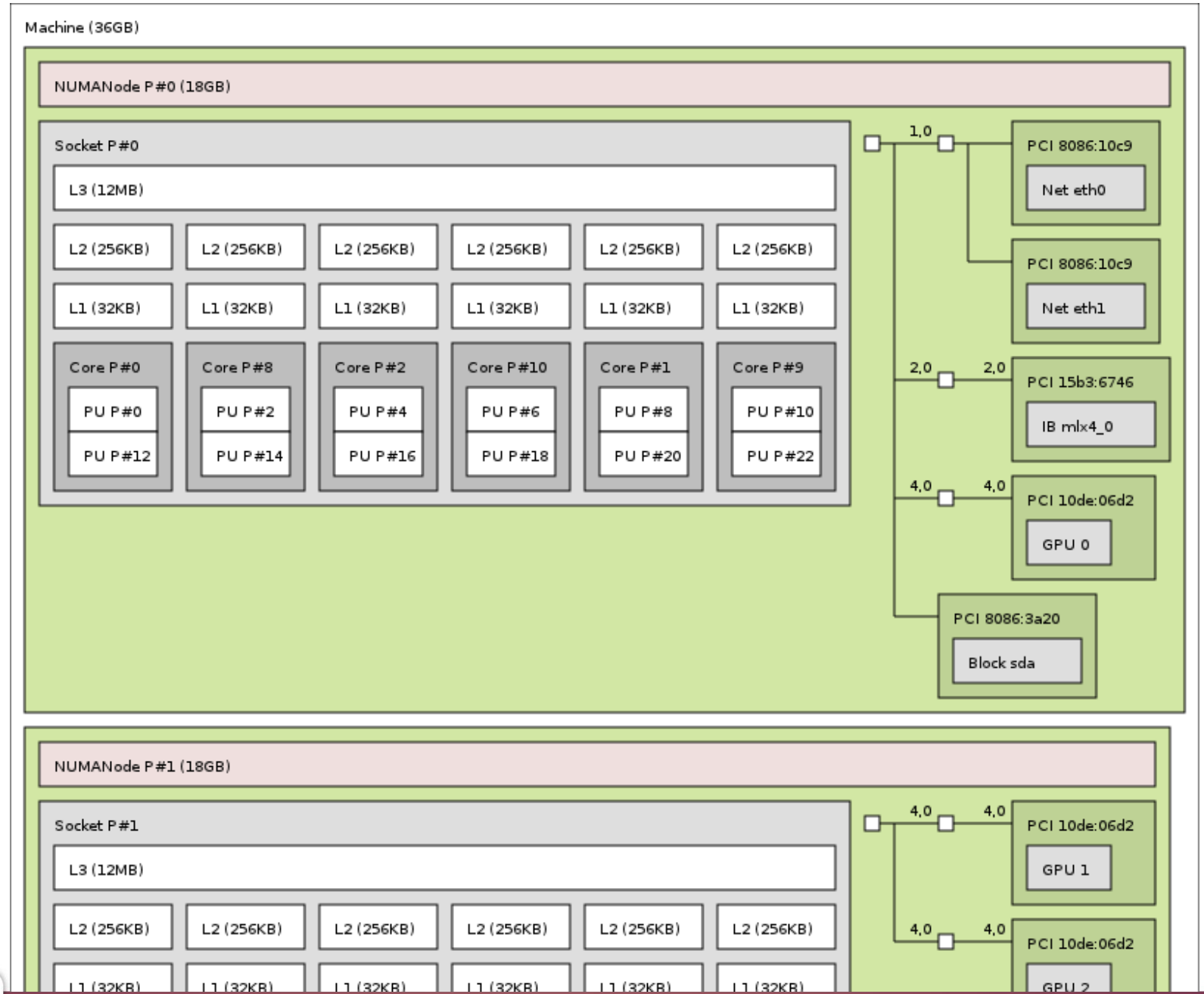
- Sockets, Caches, Cores, Threads, Memory, etc.
- Logical identifiers
  - Portability is the rule
- Which caches are shared by which cores ?

Support for multiple nodes

- Global view of clusters, etc.

Attached I/O objects

- Which cores/memory are close a NIC or GPU ?


Machine (36GB)
NUMANode P#0 (18GB)
Socket P#0
L3 (12MB)
L2 (256KB)
L2 (256KB)
L2 (256KB)
L2 (256KB)
L2 (256KB)
L2 (256KB)
L1 (32KB)
L1 (32KB)
L1 (32KB)
L1 (32KB)
L1 (32KB)
L1 (32KB)
Core P#0
PU P#0
PU P#12
Core P#8
PU P#2
PU P#14
Core P#2
PU P#4
PU P#16
Core P#10
PU P#6
PU P#18
Core P#1
PU P#8
PU P#20
Core P#9
PU P#10
PU P#22
1,0
PCI 8086:10c9
Net eth0
PCI 8086:10c9
Net eth1
2,0
2,0
PCI 15b3:6746
IB mlx4_0
4,0
4,0
PCI 10de:06d2
GPU 0
PCI 8086:3a20
Block sda
NUMANode P#1 (18GB)
Socket P#1
L3 (12MB)
L2 (256KB)
L2 (256KB)
L2 (256KB)
L2 (256KB)
L2 (256KB)
L2 (256KB)
L1 (32KB)
L1 (32KB)
L1 (32KB)
L1 (32KB)
L1 (32KB)
L1 (32KB)
4,0
4,0
PCI 10de:06d2
GPU 1
4,0
4,0
PCI 10de:06d2
GPU 2

The need for quantitative criterias

hwloc only provides *logical* distances

- Useful for placement decision
- Not for knowning if placement choices are critical
- Not for choosing between communication strategies
  - Is the NUMA interconnect fast enough to hide the distance?
  - Is the cache too slow/small to help much?

➢ Annotate the hwloc tree with quantitative information

  - Ongoing work with Inria Grenoble

5
Conclusions

Locality and affinity matter

Hardware affinities are everywhere

You can live without looking at them

The impact on performance depends on lot on the hardware

But you can easily get small improvements everywhere in the HPC stack

(and hwloc can help you)

Thank you!
Questions?

Brice.Goglin@inria.fr

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


informatics mathematics
Inria