Optimizing MPI Communication Within Large Multicore Nodes with Kernel Assistance

S. Moreaud, B. Goglin, *D. Goodell*, R. Namyst

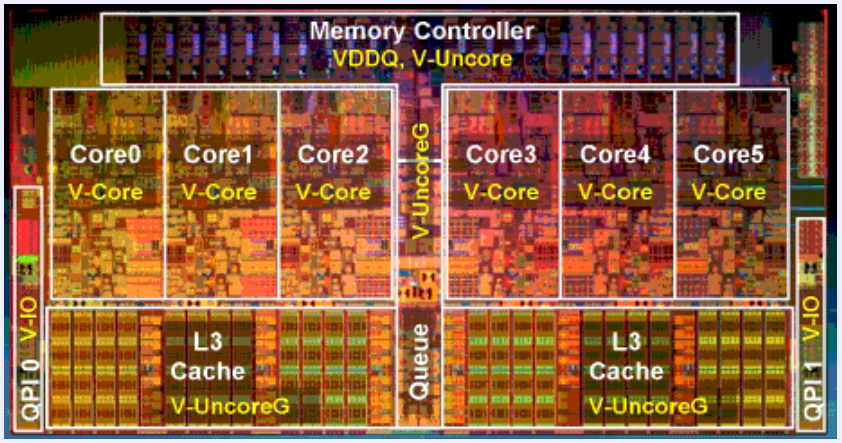
University of Bordeaux
RUNTIME team, LaBRI – INRIA, France

Argonne National Laboratory

Context

Interprocess communication

→ MPI dominates



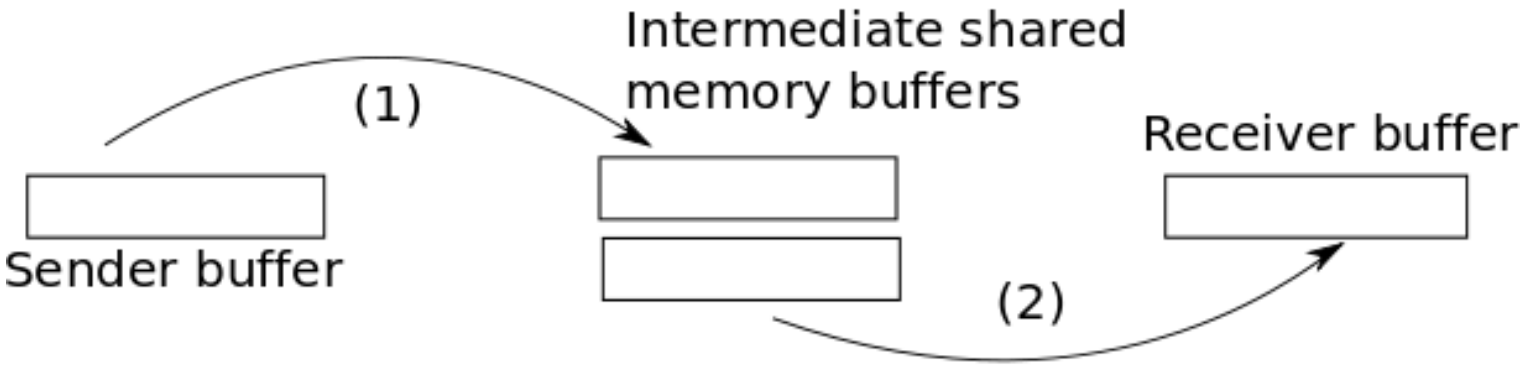
Increasing number of cores

Increasing complexity of computing nodes

→ Need for efficient intranode communication

Dedicated intranode communication model

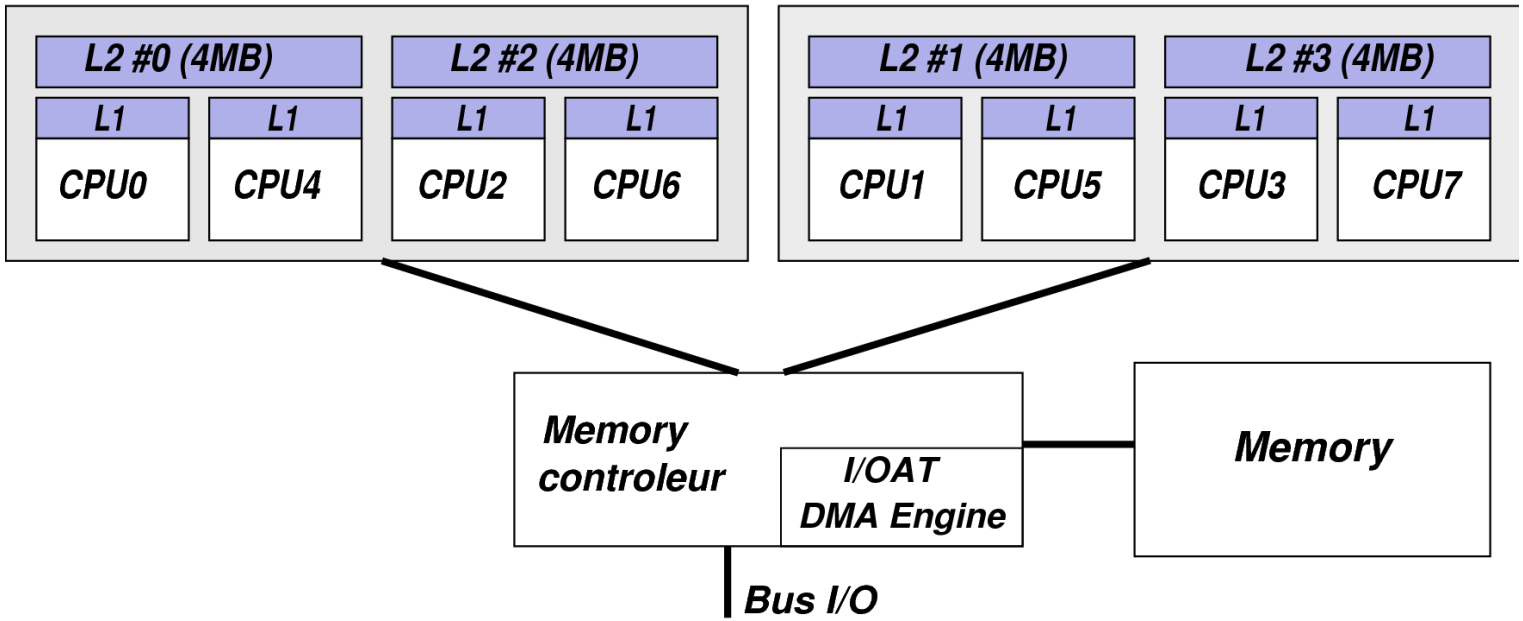
Traditional intranode MPI



Double-Buffering

- Using shared memory buffers
- 2 copies required
- Pipelined strategy

Double-Buffering experiment



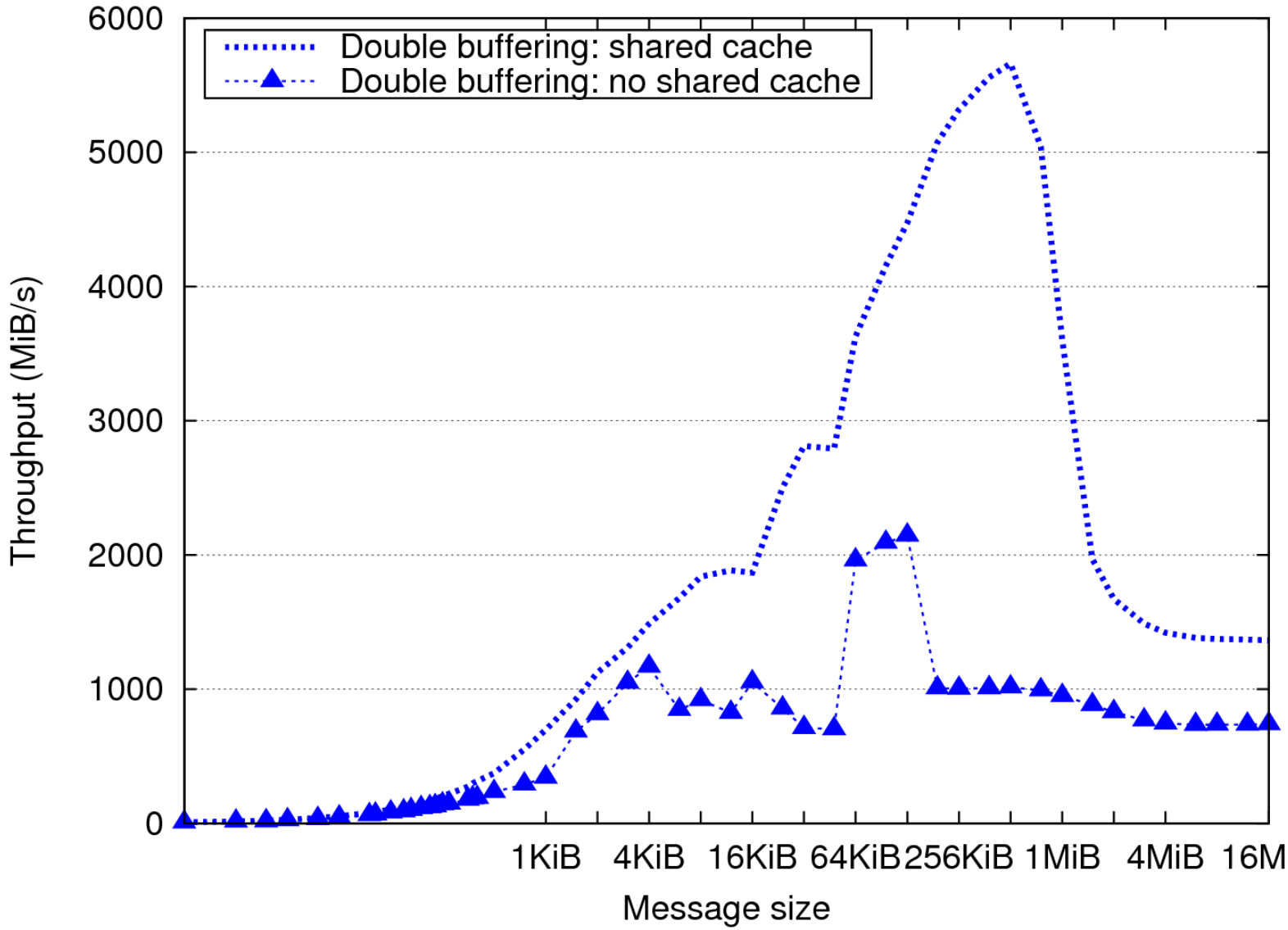
Dual Quad-Core Intel Xeon 2.33 GHz *Clovertown*

Intel MPI Benchmark *(IMB) Pingpong*

MPICH2-nemesis MPI implementation

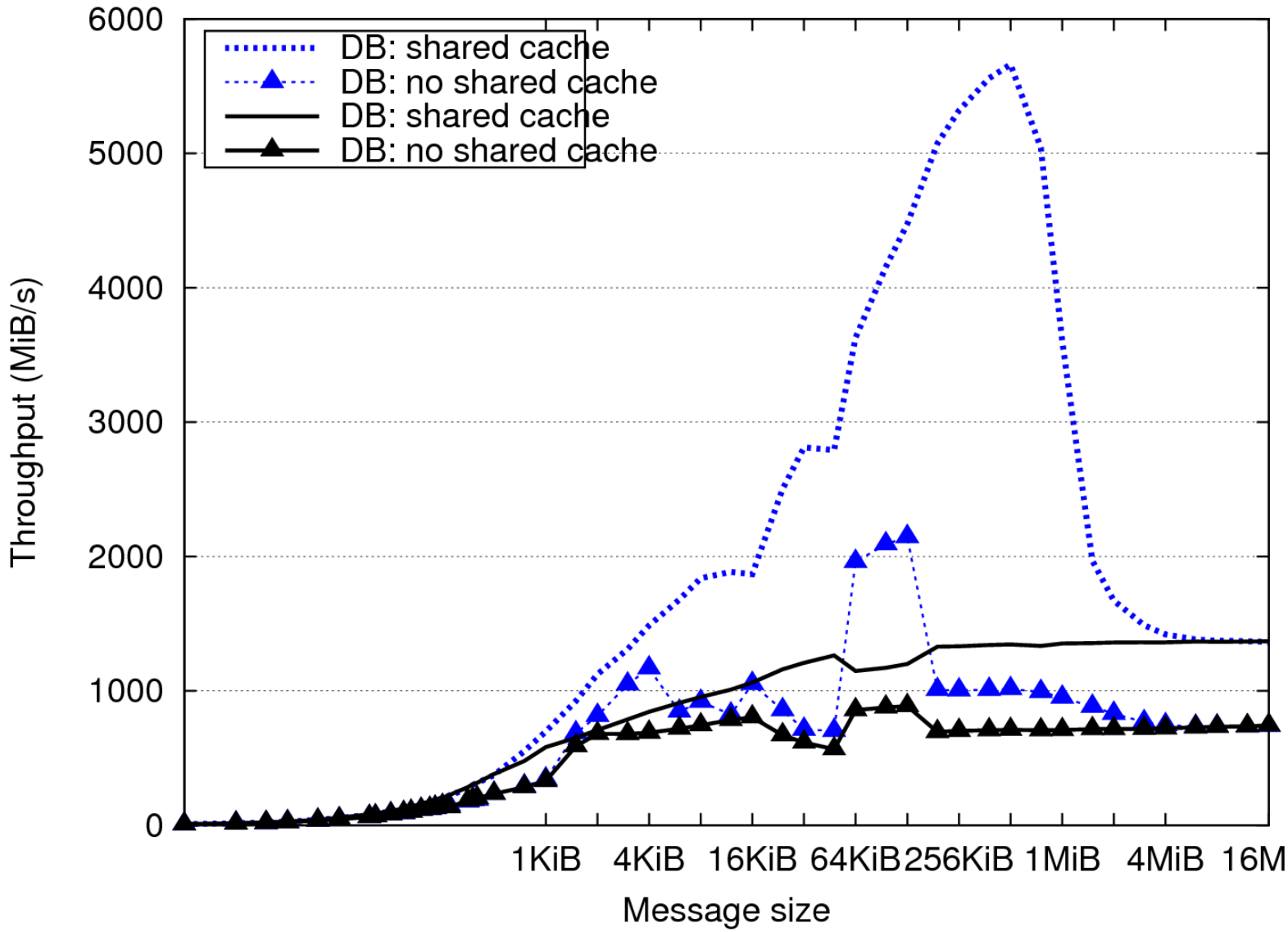
IMB Pingpong with Double-Buffering

(2.33GHz Clovertown)



IMB Pingpong with Double-Buffering no cache re-usage

(2.33GHz Clovertown)



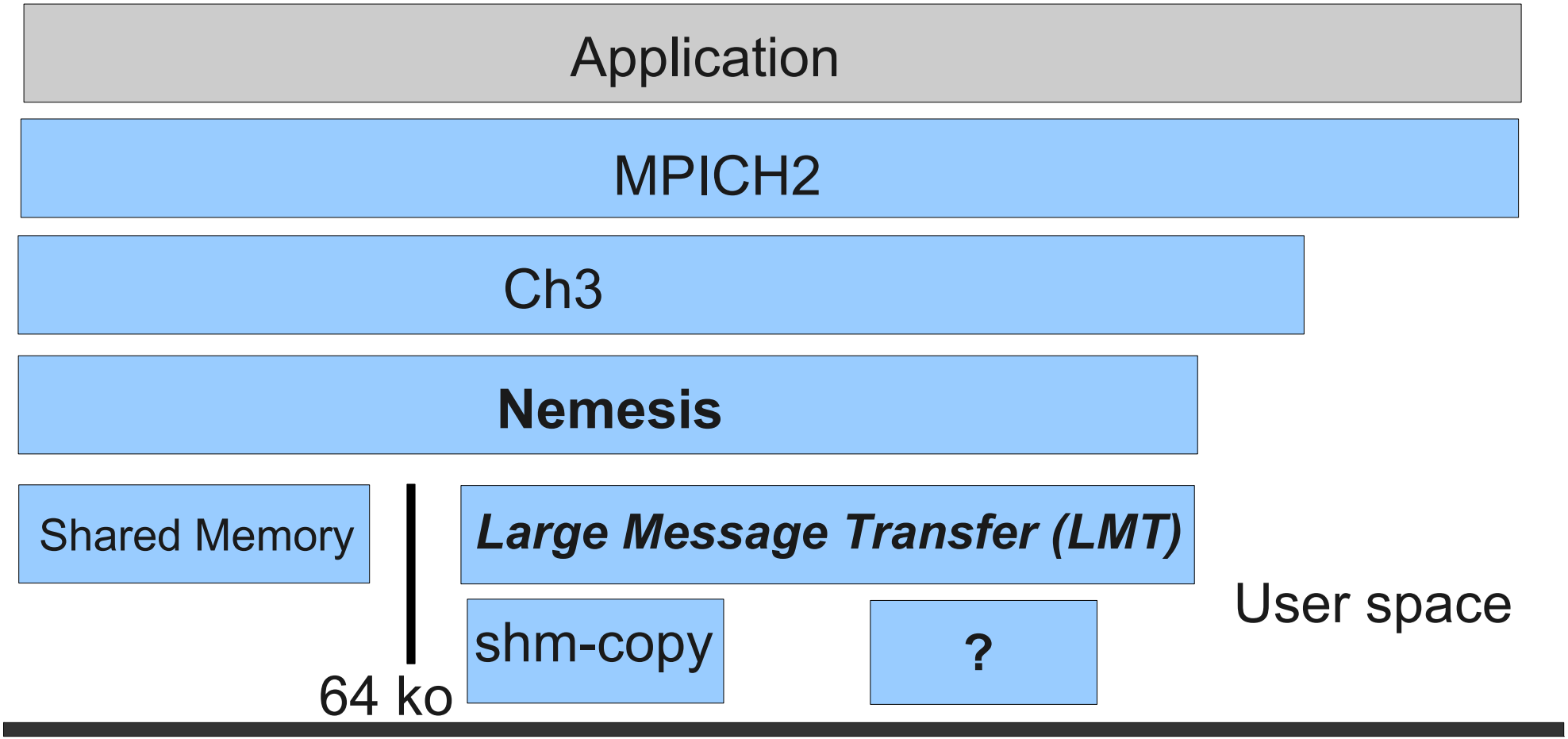
Double-Buffering So what?

- ✔ Efficient for small messages
  - latency for 0-byte MPI message < 200ns
- ✗ High CPU use
- ✗ Cache pollution
- ✗ Bandwidth consumption

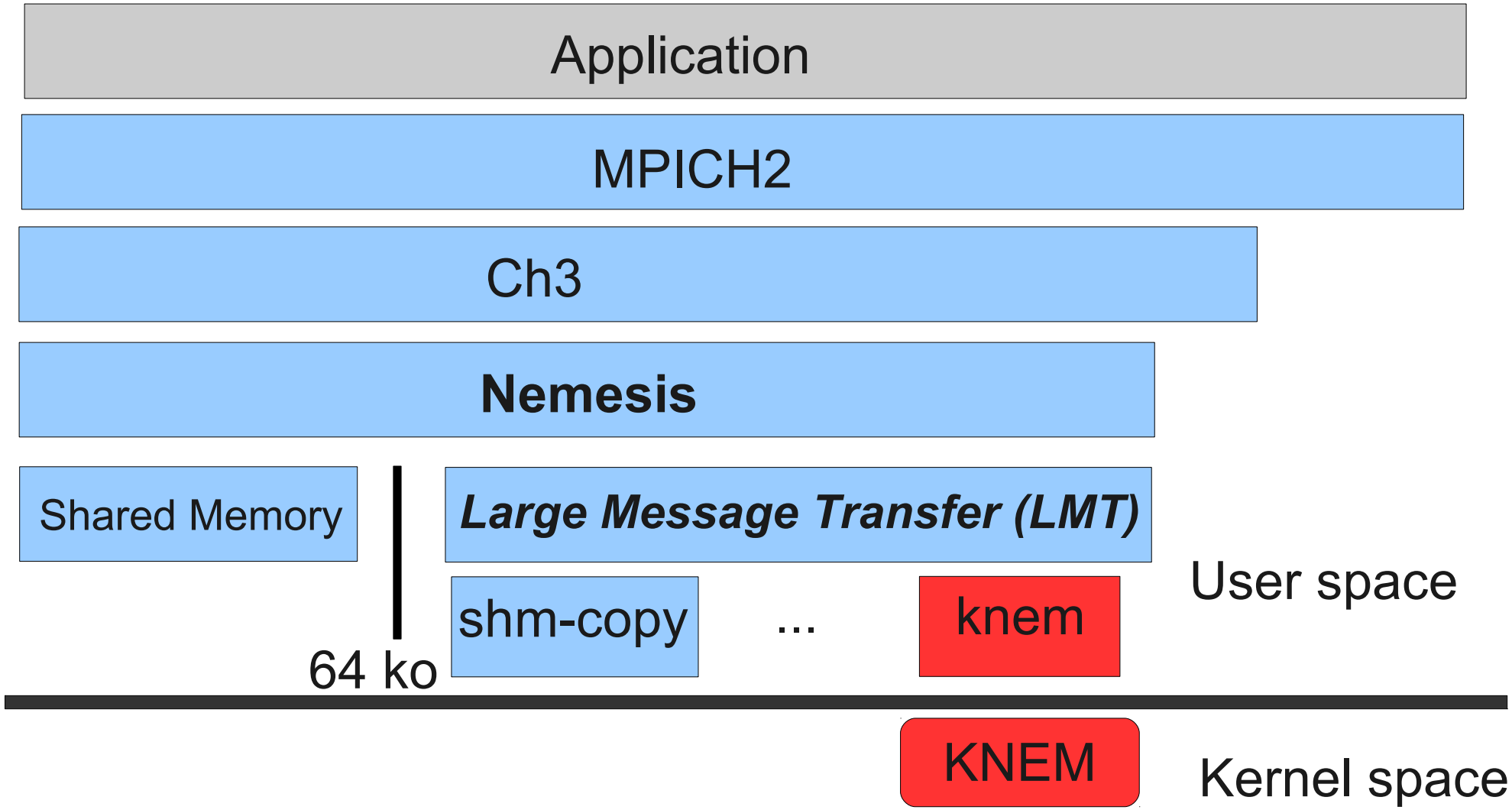
→ Need large shared cache to be efficient

How to improve performance for large data transfers?

MPICH2-Nemesis



MPICH2-Nemesis + KNEM



What's KNEM?

KNEM (Kernel Nemesis)

- Dedicated Linux kernel module
- Single memory copy
- Synchronous and asynchronous modes

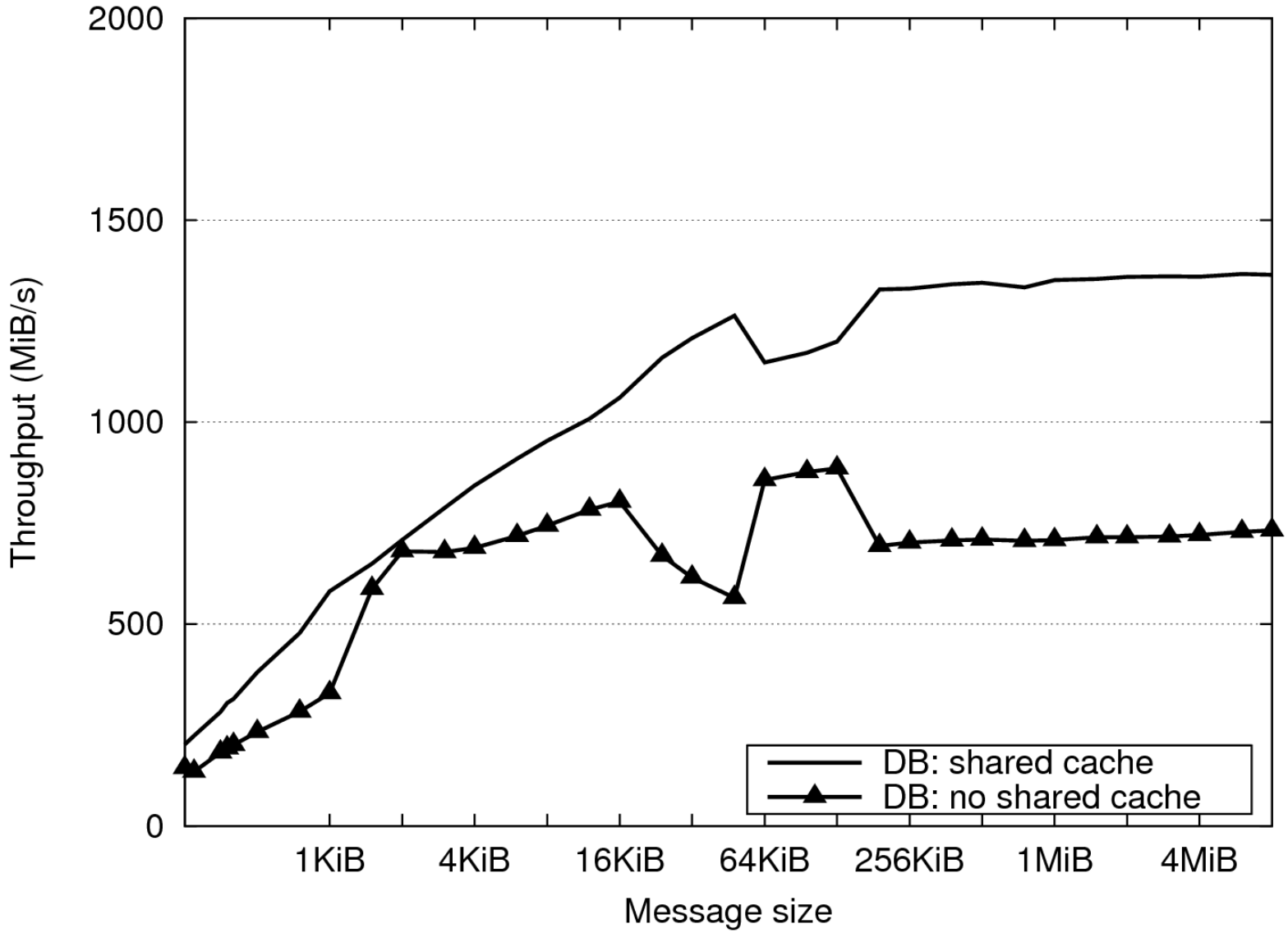
KNEM+I/OAT

- Offloading copy on Intel *Input/Output Acceleration Technology*
- Efficient copy in background
- CPU and cache not involved
- Performance does not depend on process location

Available in MPICH2 and soon in OpenMPI

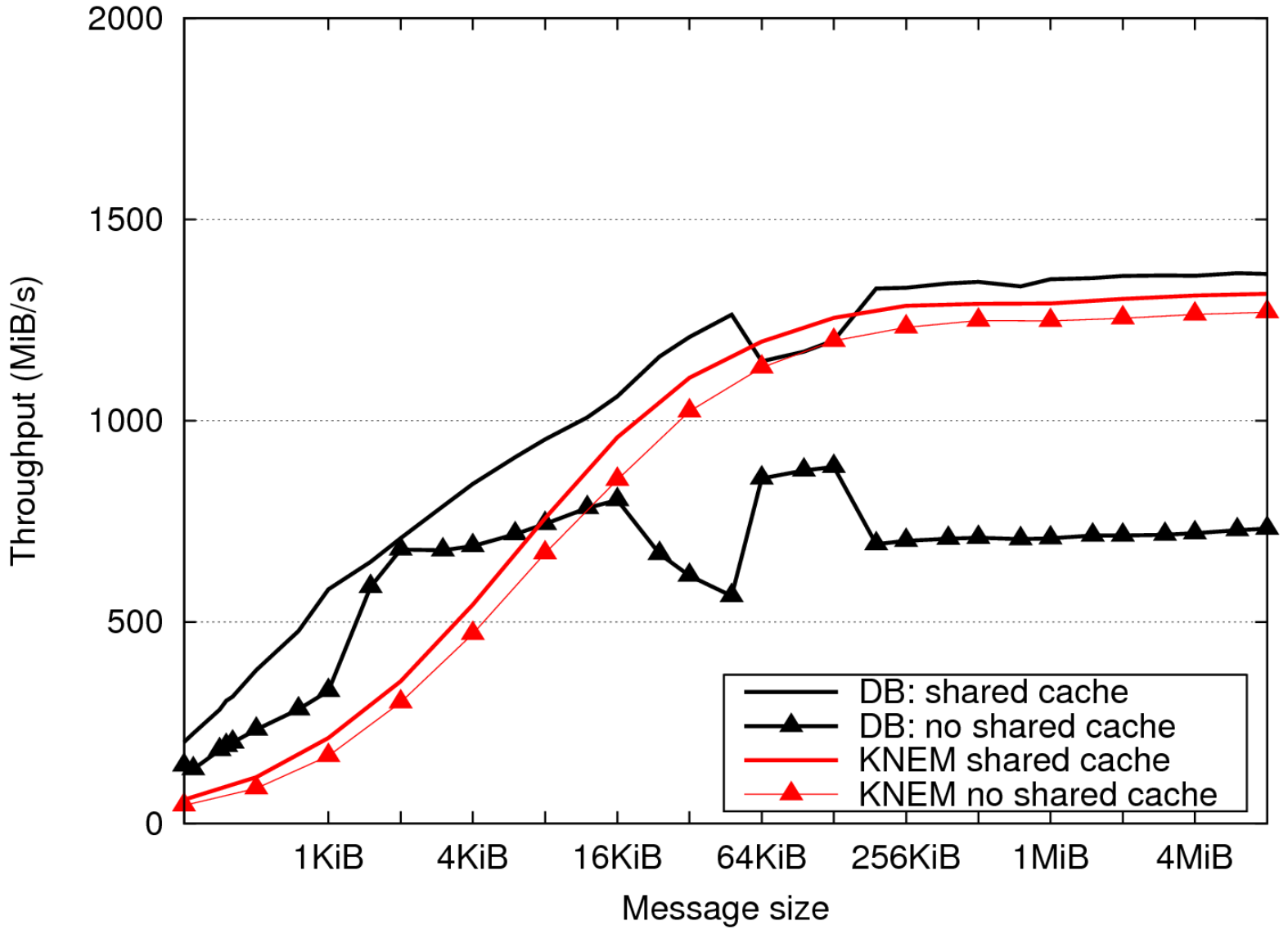
KNEM: Point to point performances

(2.33GHz Clovertown)



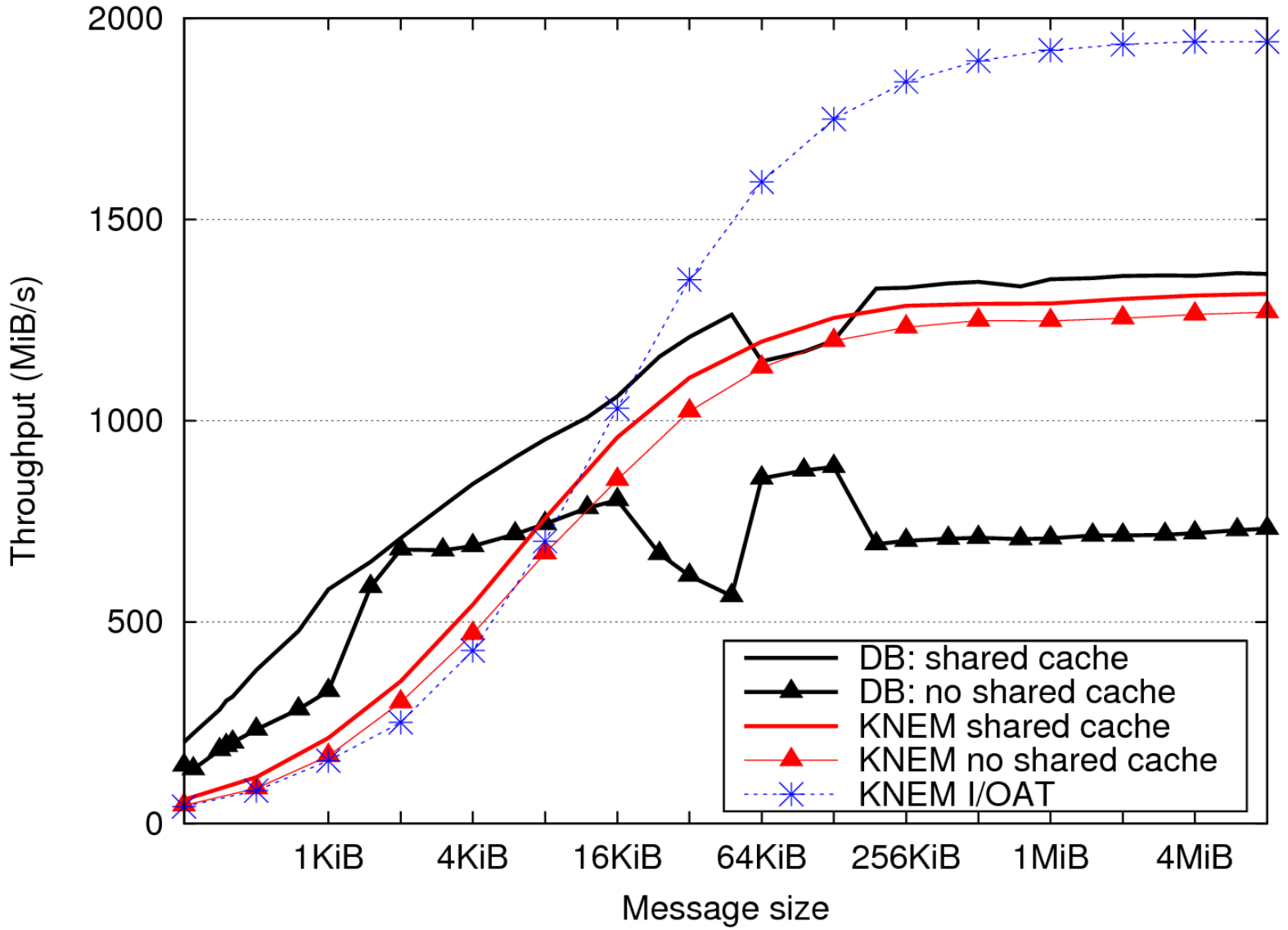
KNEM: Point to point performances

(2.33GHz Clovertown)



KNEM: Point to point performances

(2.33GHz Clovertown)



What to expect?

Intranode communication performance vary with

- Strategy (double buffering, KNEM, KNEM+I/OAT)
- Placement, cache size
- Optimization by dynamically switching between strategies?

What is the impact of the location on larger architectures?

What about contention?

- Collective operations?
- Application?

Experimentation Platform

4 Intel-based architectures

- Dual Quad-Core *Clovertown* 2.33 GHz
- Dual Quad-Core *Nehalem* 2.66 GHz
- Quad Hexa-Core *Dunnington* 2.66 GHz
- 4 Quad Hexa-Core *Dunnington* 2.66 GHz NUMA nodes (without I/OAT)

Various number of cores (8, 24, 64)

Different shared caches

Experimentation Platform

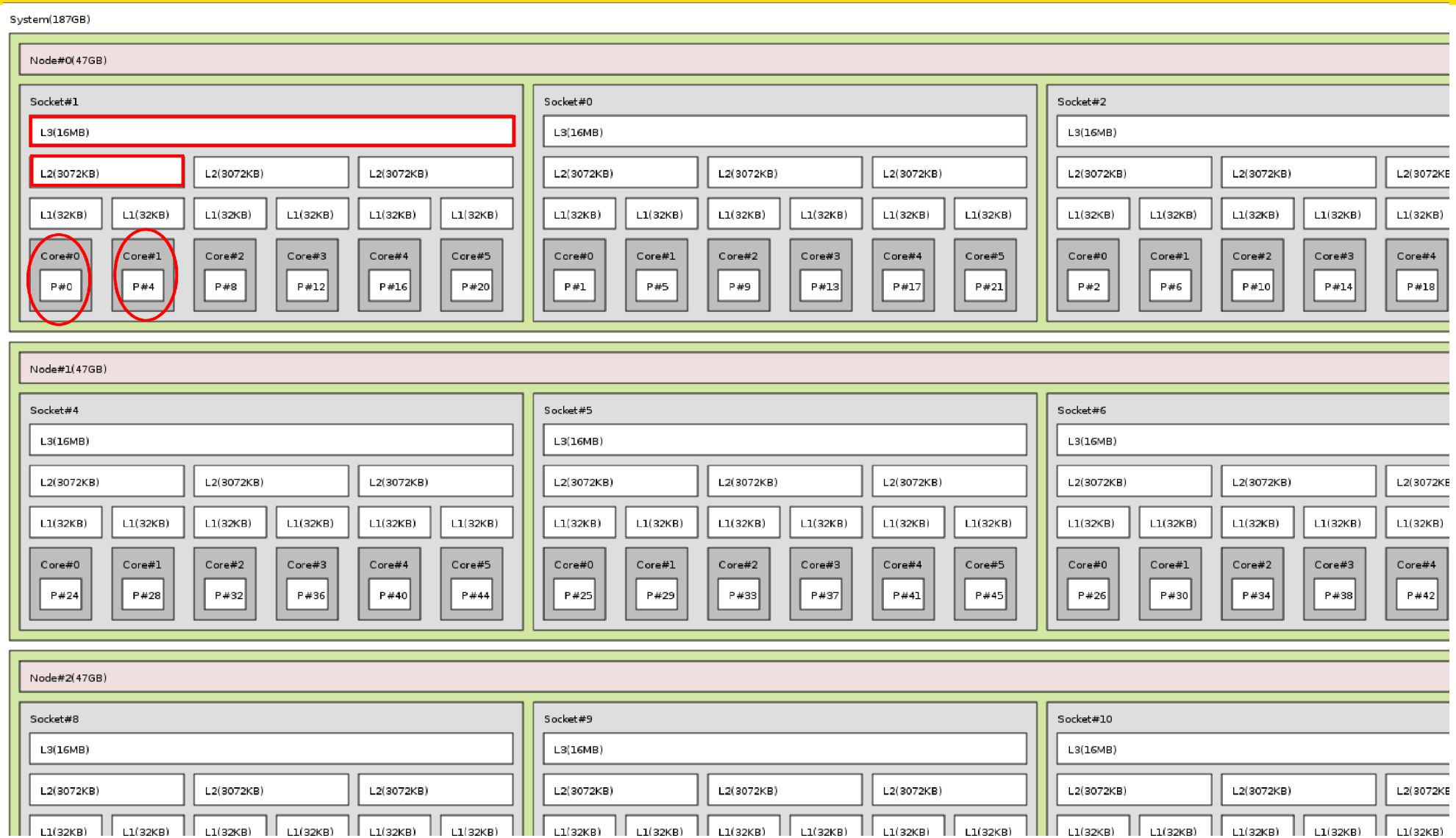
Relative placements

Experimentation Platform

Relative placements

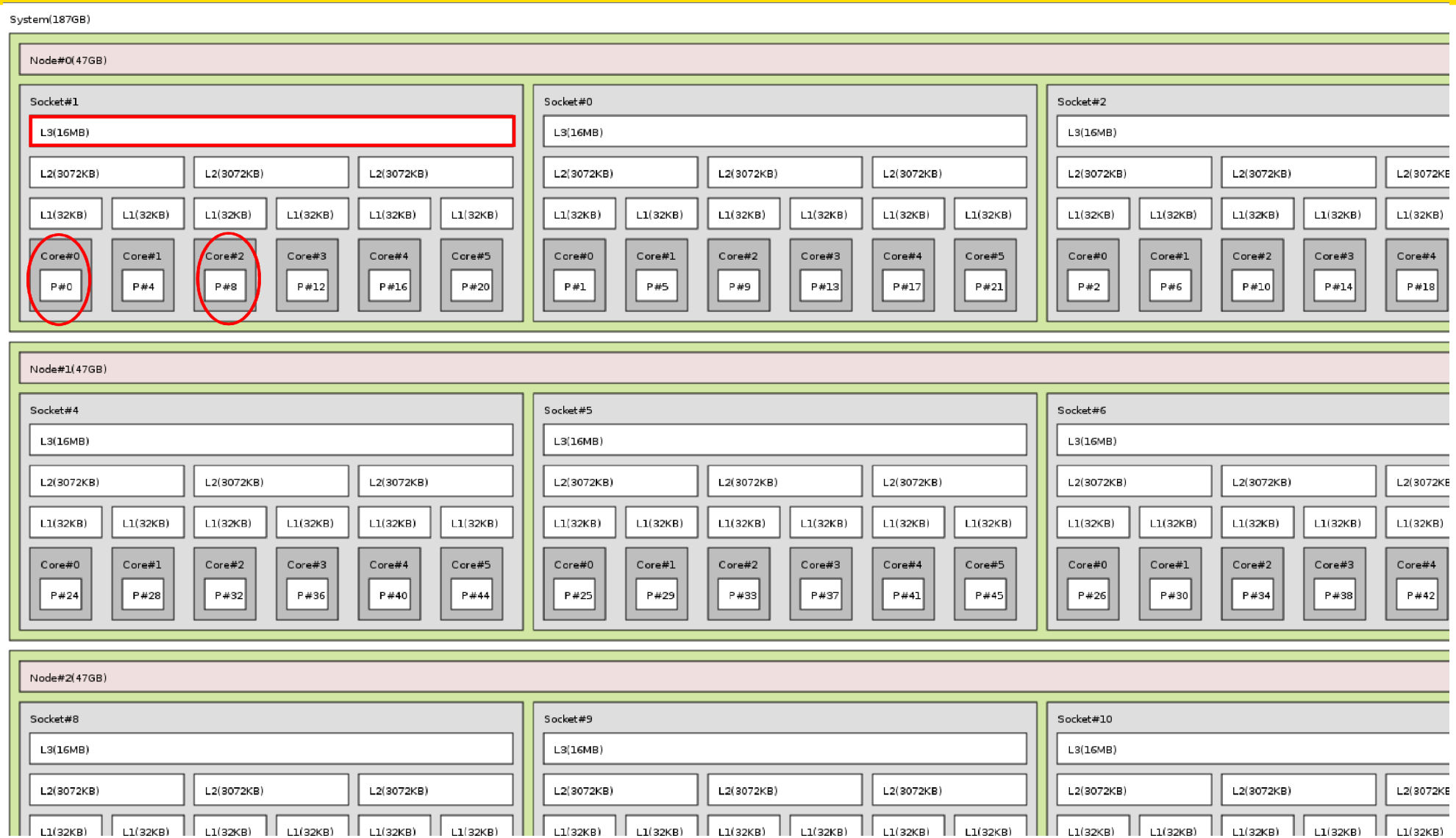
Experimentation Platform

Relative placements



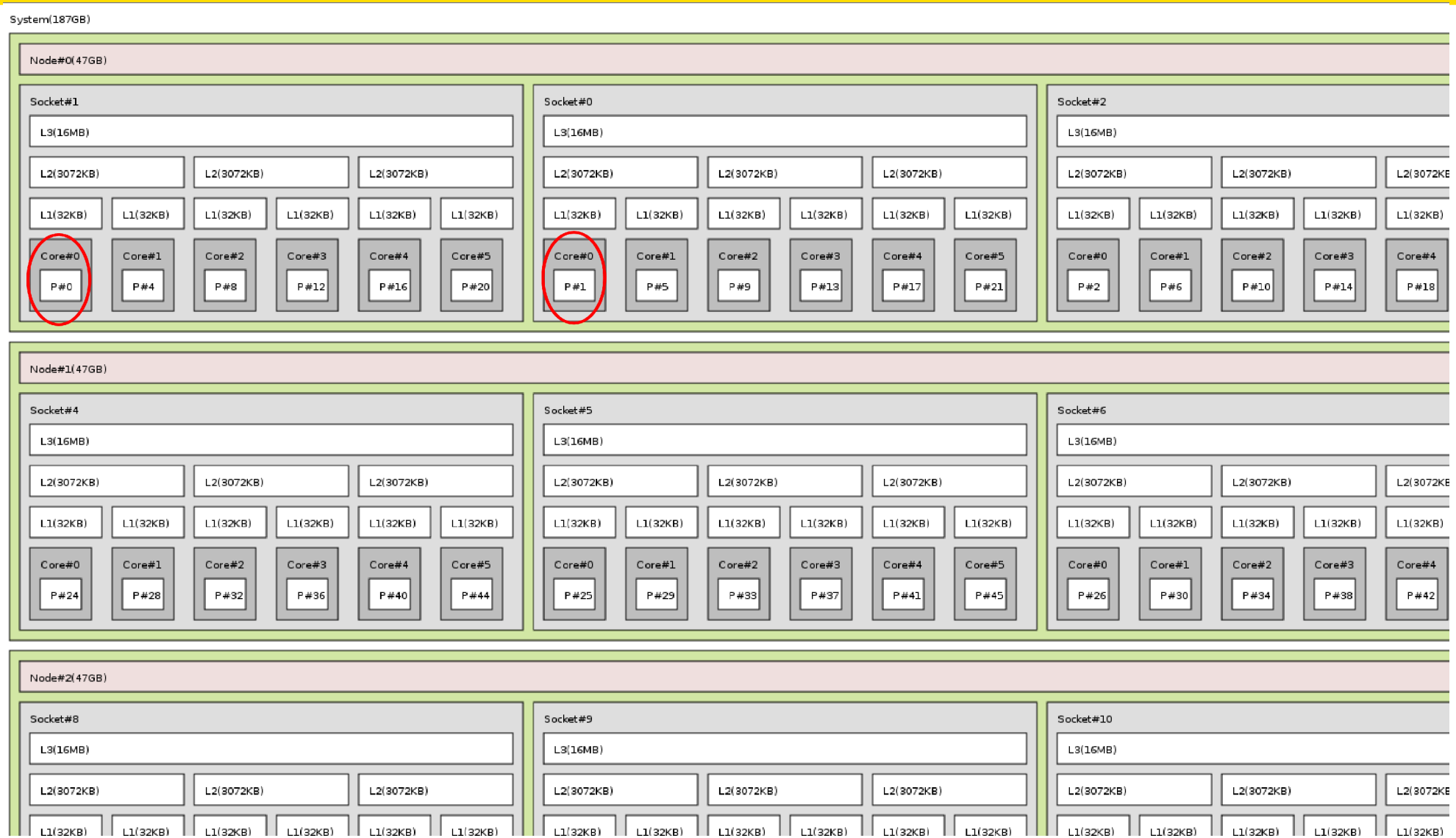
Experimentation Platform

Relative placements



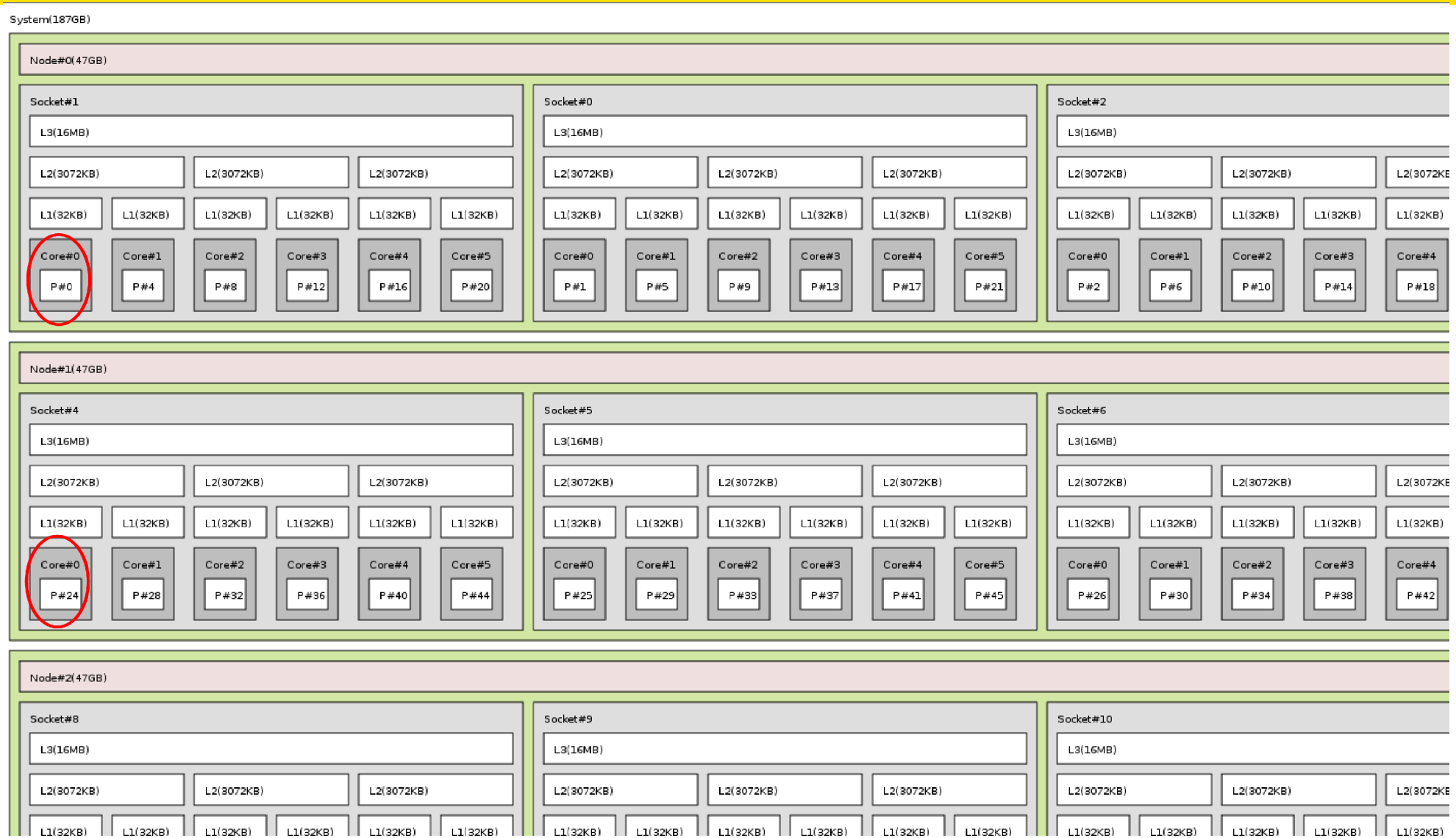
Experimentation Platform

Relative placements



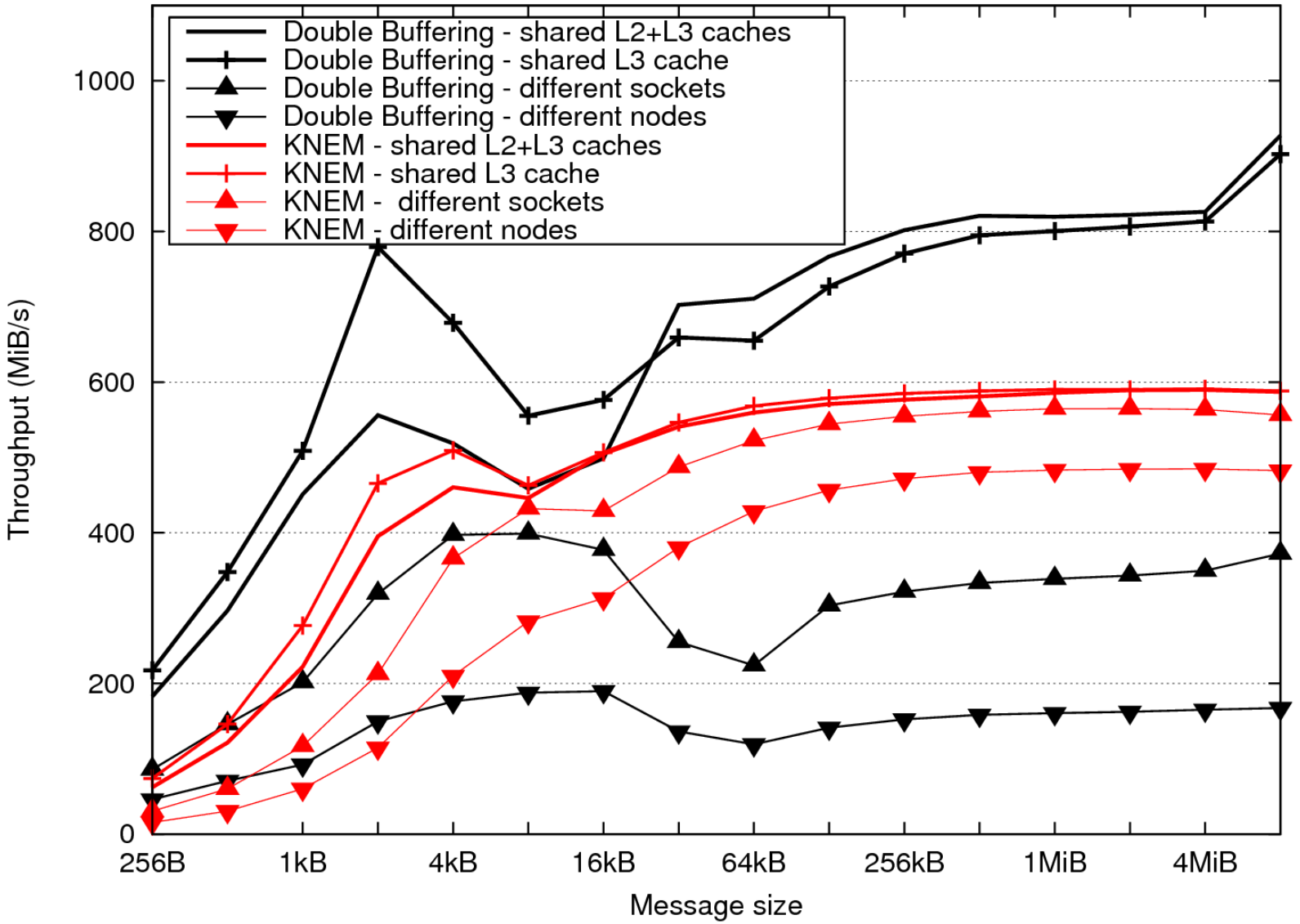
Experimentation Platform

Relative placements



Impact of process placement

IMB Pingpong *(NUMA Dunnington)*

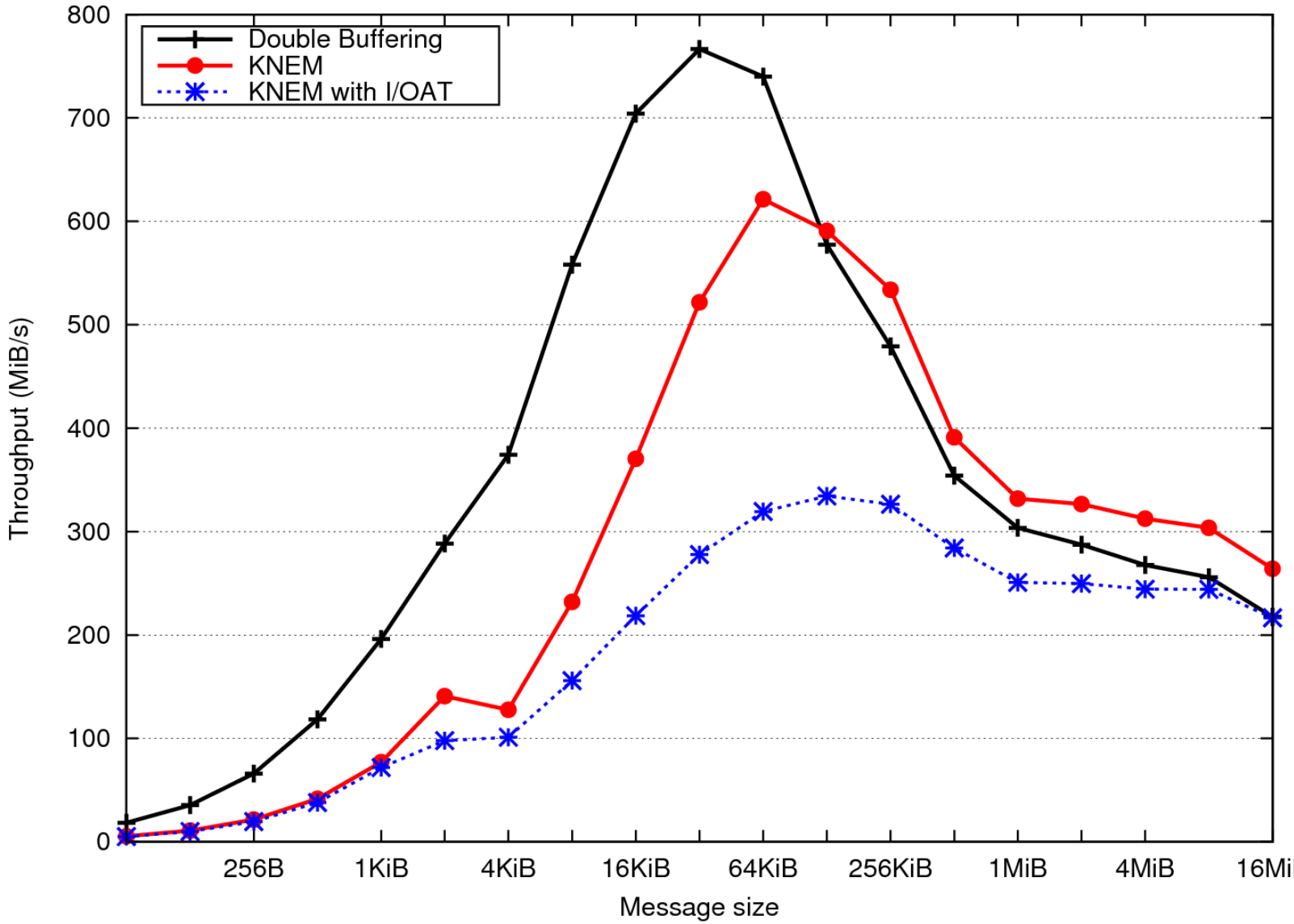


Impact of process placement

- Shared memory double copy faster than KNEM for small messages
- KNEM better for medium and large messages without shared cache (> 16 KiB)
- I/OAT interesting starting at 16 KiB, except on Nehalem
- Switching between strategies point-to-point model based
- What about collective operations?

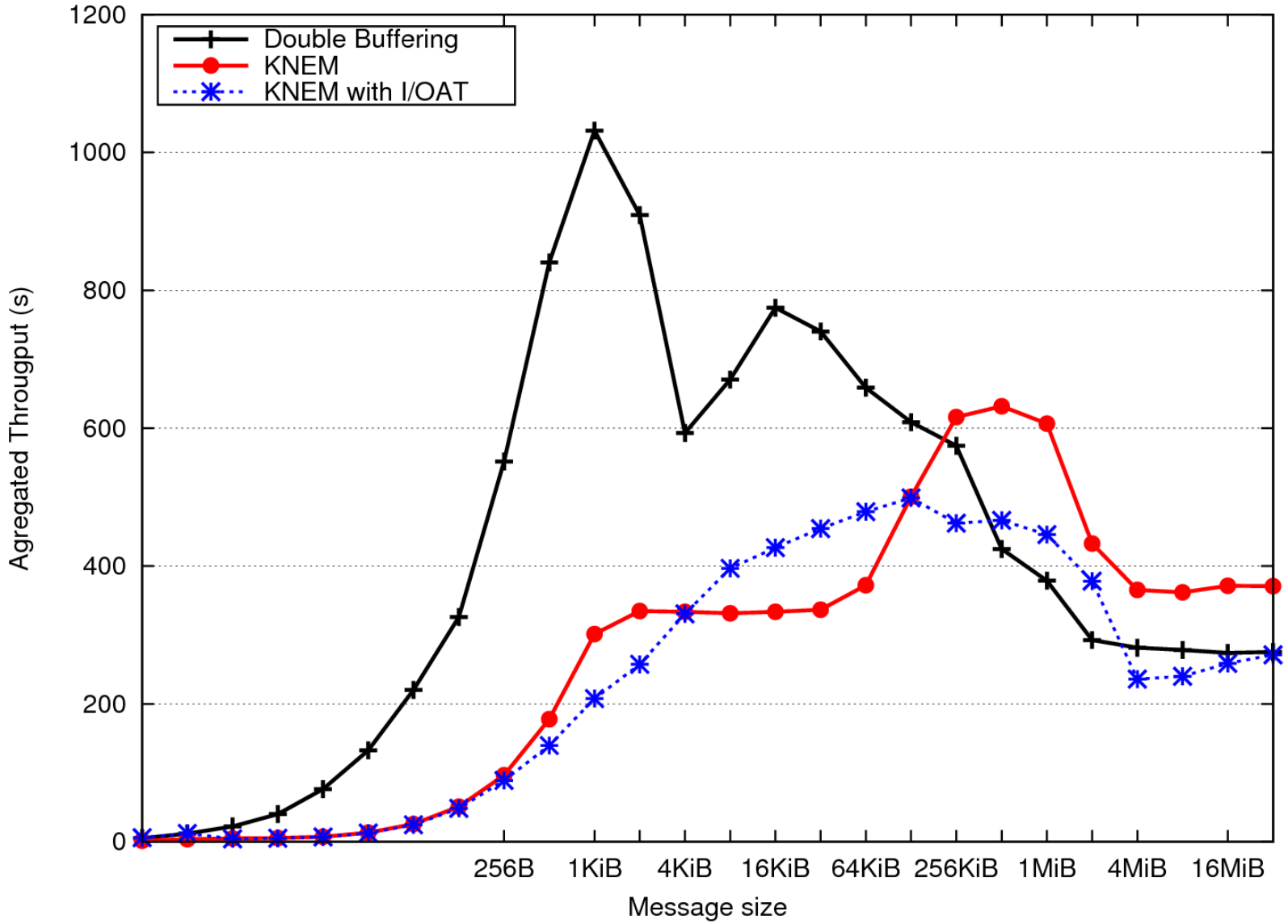
Collectives: *N-to-One* Pattern

IMB Reduce *(2.66 GHz Nehalem, np=8)*



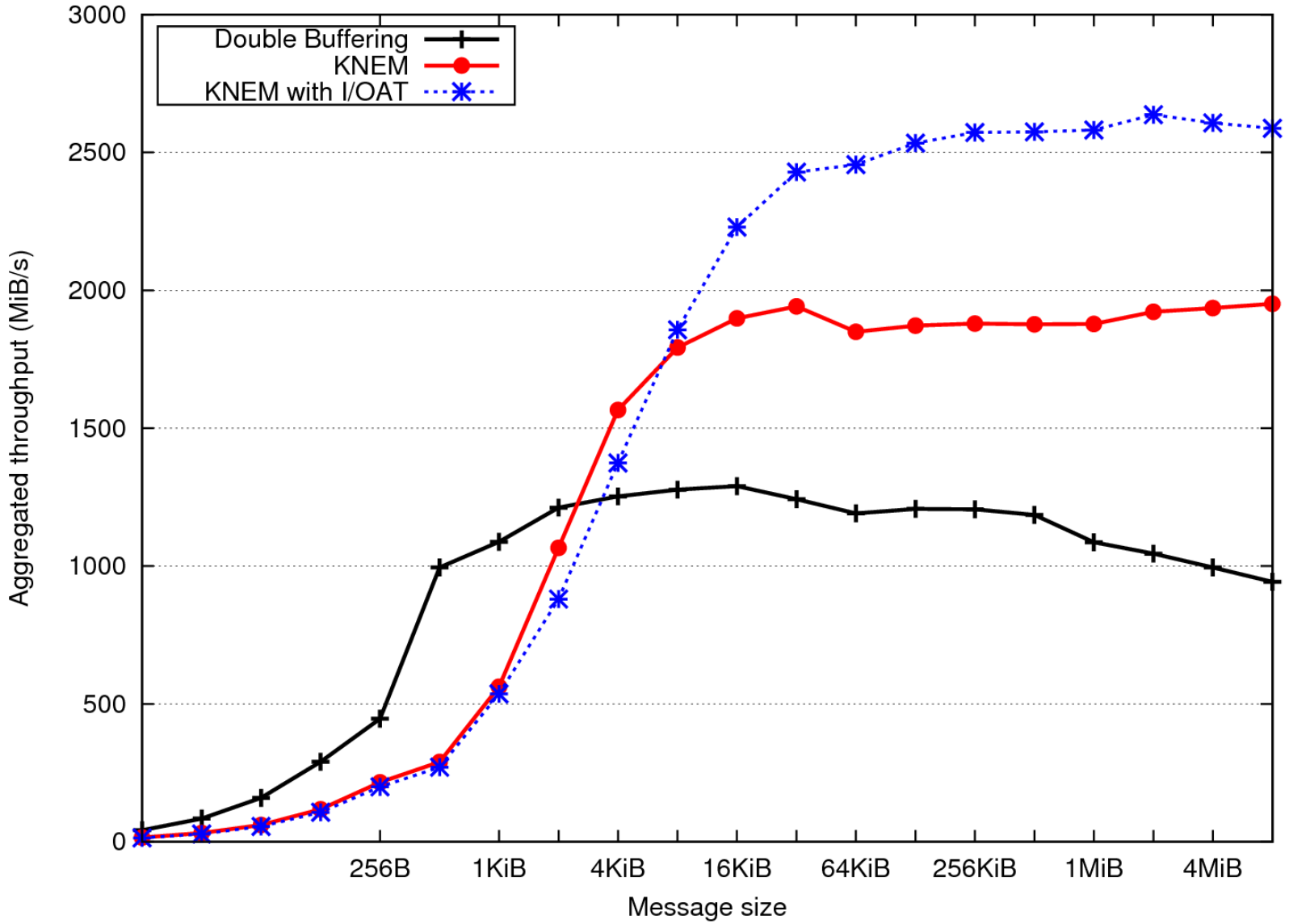
Collectives: *One-to-N* Pattern

IMB Scatter *(2.66 GHz Dunnington, np=16)*



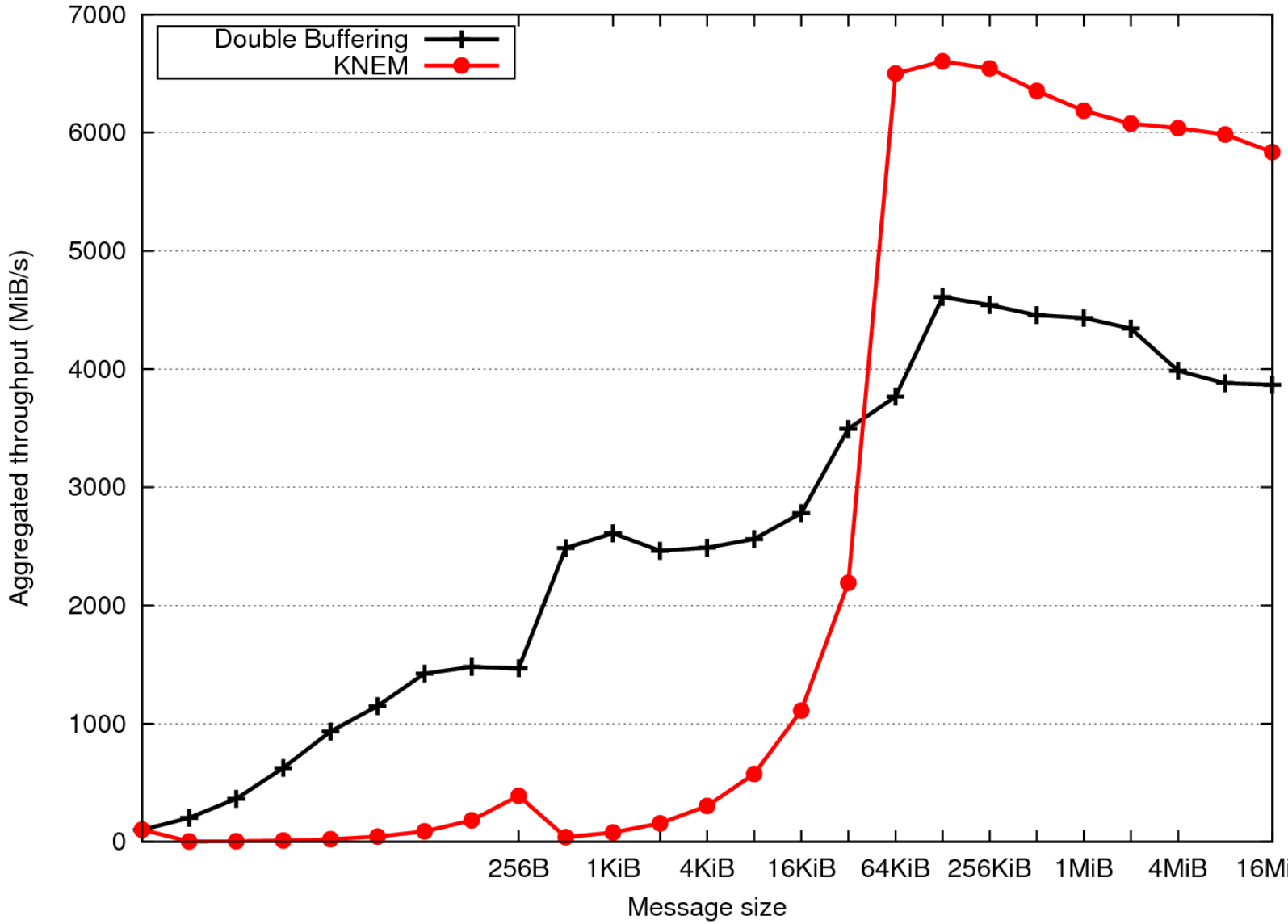
Collectives: *N-to-N* Pattern

IMB Alltoall (2.33 GHz Clovertown, np=8)



Collectives: *N-to-N* Pattern

IMB Alltoall *(NUMA Dunnington, np=64)*



Collective operations

All-to-N, One-to-N Pattern

- Double buffering still faster for small messages
- KNEM more interesting for larger messages (>128KiB)
- Behavior and threshold vary slightly with the hosts

N-to-N Pattern

- Alltoall: biggest dependency of placement and topology
  - KNEM improve performance on all architectures
    → from 50% to 100%
    → significantly reducing contention

NAS Parallel Benchmark

Machine	Benchmark	Nemesis	KNEM	Speedup
Hannibal8	ft.B.8	14.60s	13.90s	**+5.0%**
	ft.C.8	63.77s	60.31s	**+5.7%**
	is.B.8	0.70s	0.60s	**+16.6%**
	is.C.8	2.81s	2.41s	**+16.6%**
Bill8	ft.B.8	40.43s	36.02s	**+12.2%**
	ft.C.8	175.46s	158.36s	**+10.8%**
	is.B.8	2.42s	1.89s	**+12.2%**
	is.C.8	10.04s	8.02s	**+25.2%**
Idkonn24	ft.B.16	24.14s	21.70s	**+11.2%**
	ft.C.16	97.65s	85.73s	**+13.9%**
	is.B.16	1.29s	0.99s	**+30.3%**
	is.C.16	5.88s	4.43s	**+32.7%**
Bertha96	ft.C.64	31.91s	28.82s	**+10.7%**
	ft.D.64	727.37s	645.36s	**+12.7%**
	is.C.64	3.17s	2.29s	**+38.4%**
	is.D.64	65.00s	52.72s	**+23.3%**

Conclusion

Analysis of intranode MPI communication

- Double Buffering, KNEM kernel-assisted single copy
- Different multicore machines

→ Double-Buffering more sensitive to placement

→ KNEM interesting for large message

→ Complex behavior of collective operations

- Dynamically choosing the best strategy is difficult

→ I/OAT only useful for obsolete architectures

→ Large performance improvement with KNEM on NAS

Future Works

Improving the KNEM interface

- send/receive-oriented → multiple accesses to a single buffer
- receiver-directed data transfer relaxed → All-to-one

Improving dynamic adaptation to strategy

- Inside collective operation algorithms
- Processes location, message size...

Questions?

stephanie.moreaud@labri.fr
(goodell@mcs.anl.gov)

http://runtime.bordeaux.inria.fr/knem
http://www.mcs.anl.gov/research/projects/mpich2