Garbage Collection for Multicore NUMA Machines

Sven Auhagen [1] **Lars Bergstrom** [1] Matthew Fluet [2]
John Reppy [1]

[1]Department of Computer Science
University of Chicago

[2]Computer Science Department
Rochester Institute of Technology

MSPC 2011

Outline

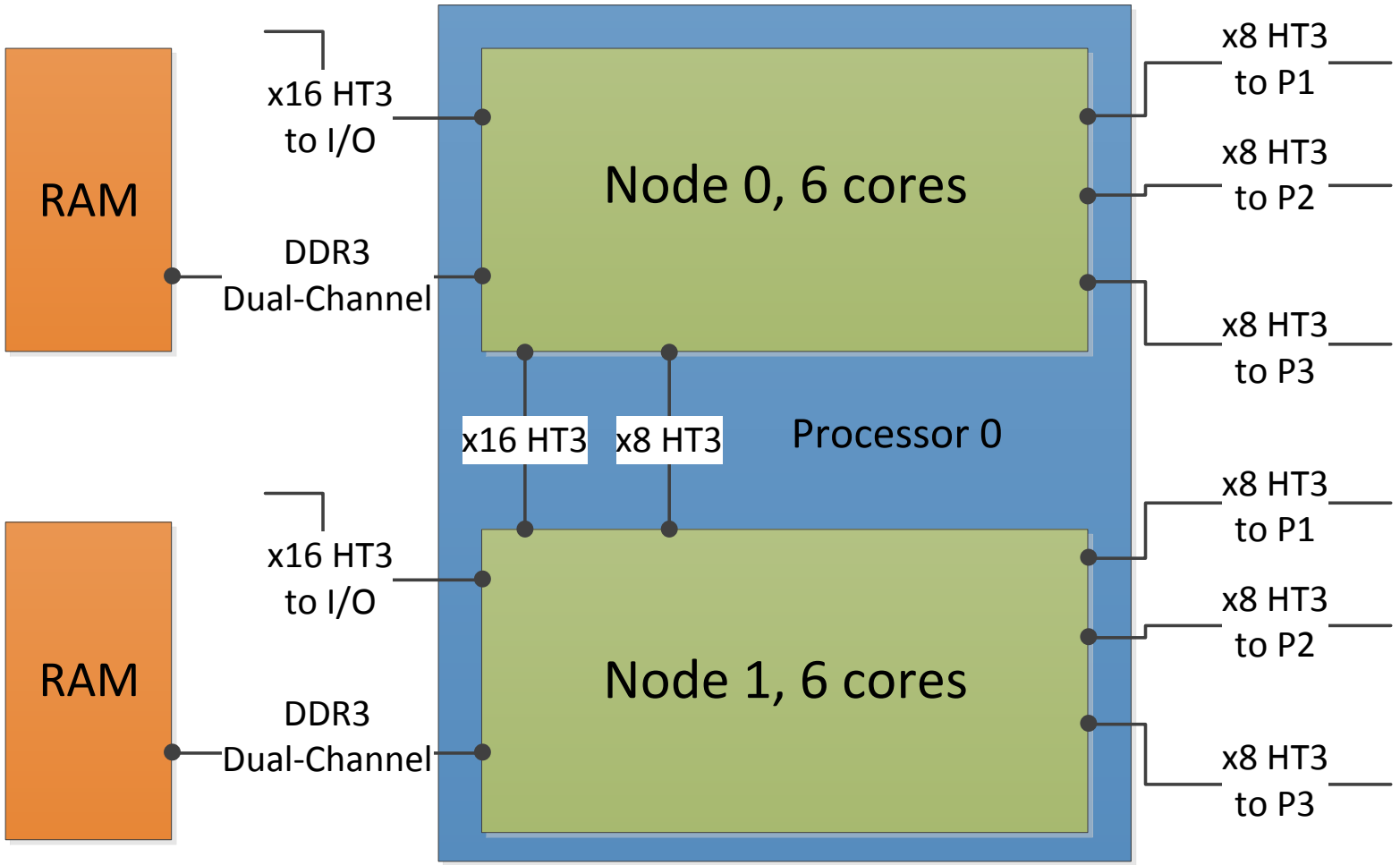
In non-uniform memory architecture (NUMA) computers,
not all memory is equally close.

- Remote memory accesses have higher latency
- Interconnects between packages saturate,
  reducing bandwidth and increasing latency

Garbage collectors and parallel code can saturate these interconnects.
We show:

- Careful page allocation increases performance by roughly 7%
- Even with good locality and page allocation, you may need to
  distribute or replicate high-contention data

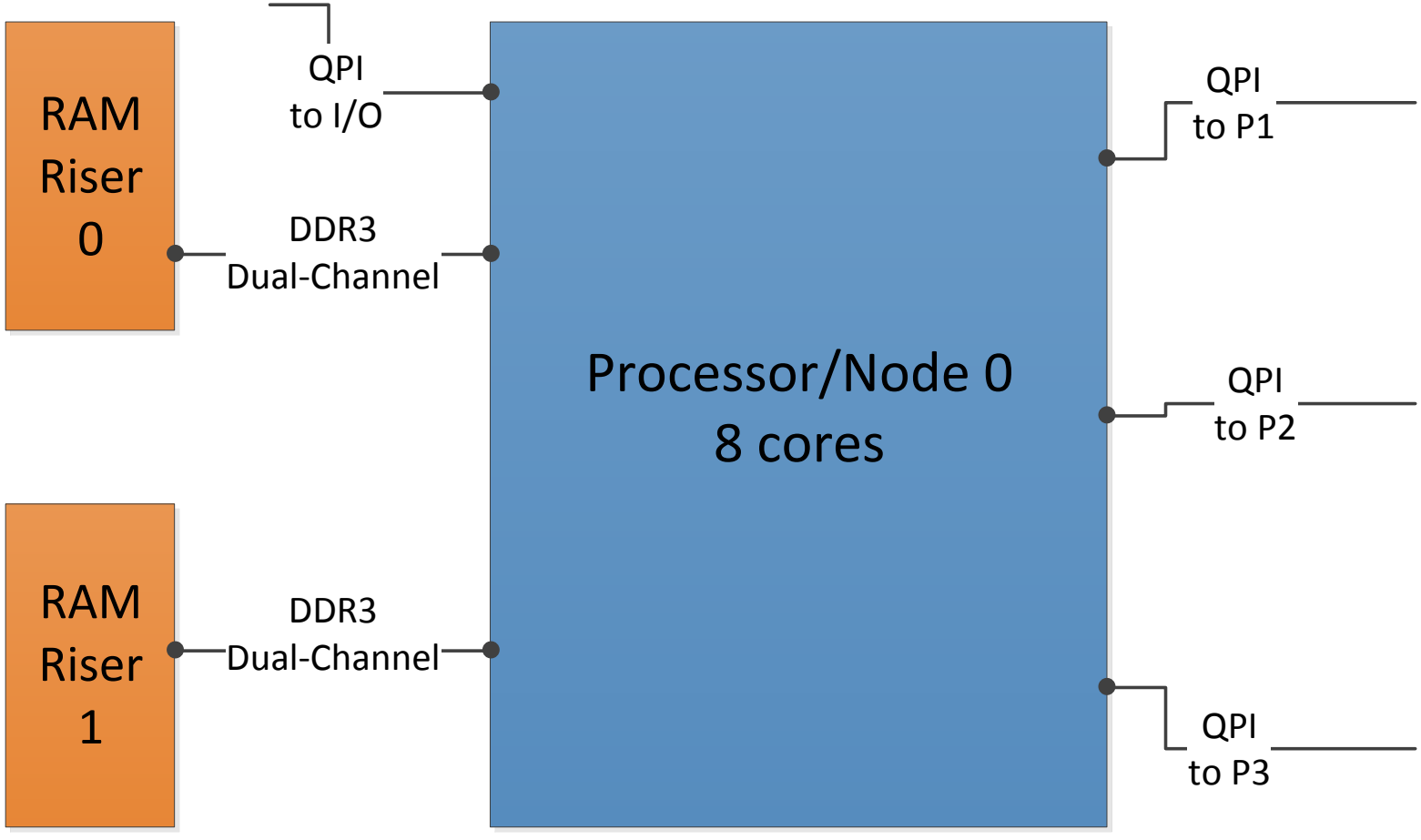
AMD Opteron $\times$ 4 $=$ 48 cores



x8 HT3 = 6.4 GB/s

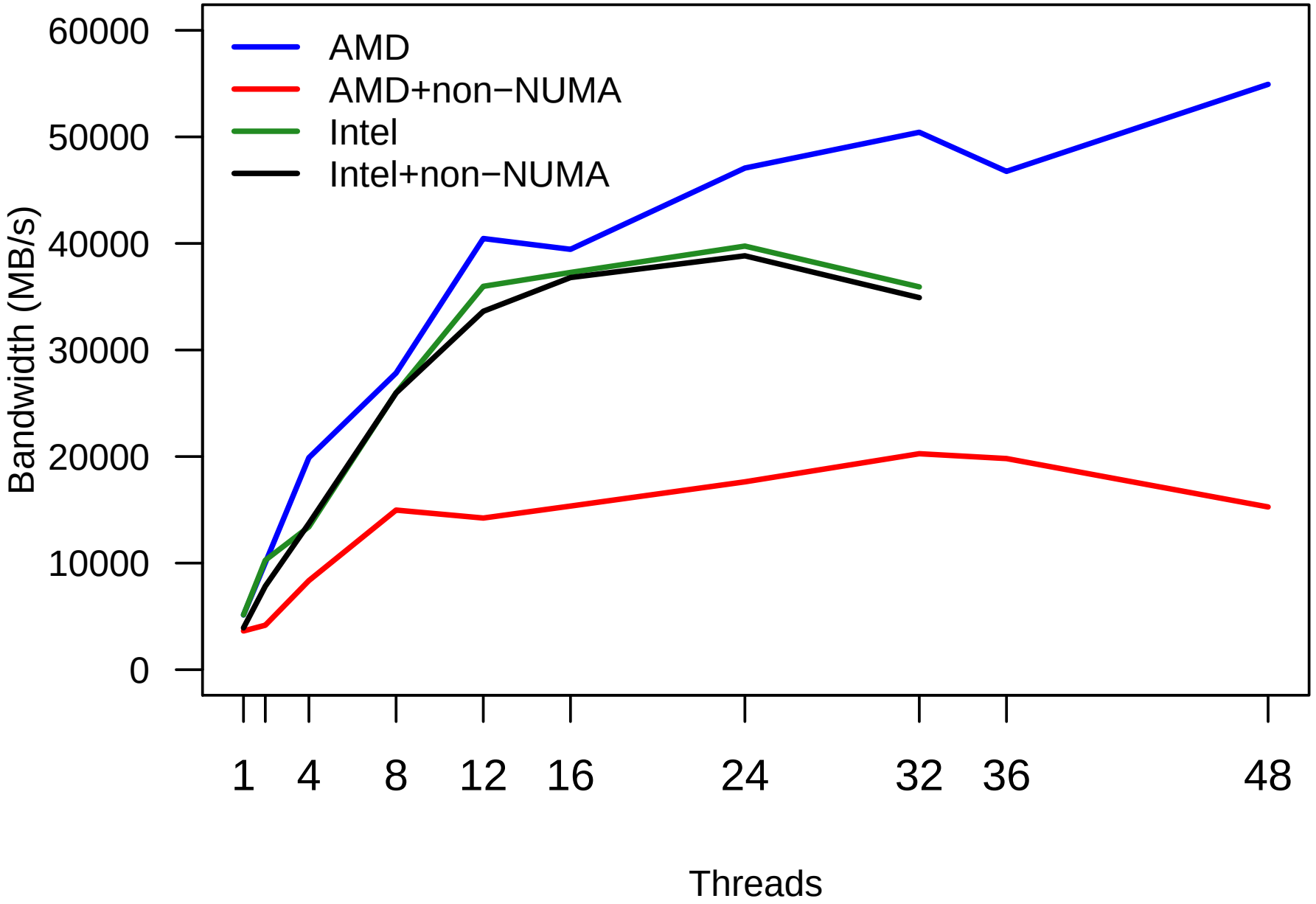
DDR3 1333 MHz = 21.3 GB/s

Intel Xeon $\times$ 4 $=$ 32 cores

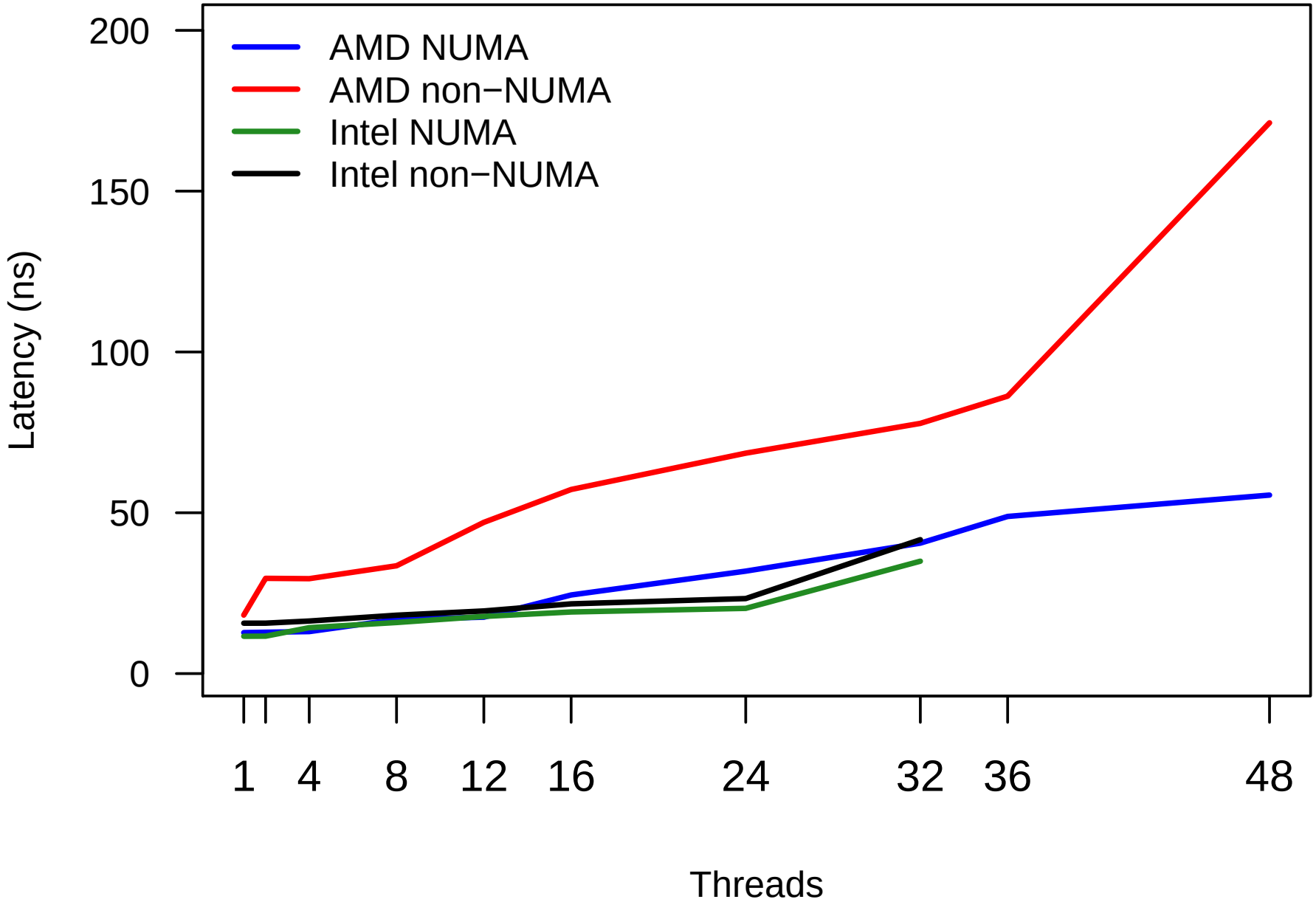


QPI = 25.6 GB/s
DDR3 1066 MHz = 17.1 GB/s

`a[i] = b[i];` bandwidth



`a[i] = b[i];` latency



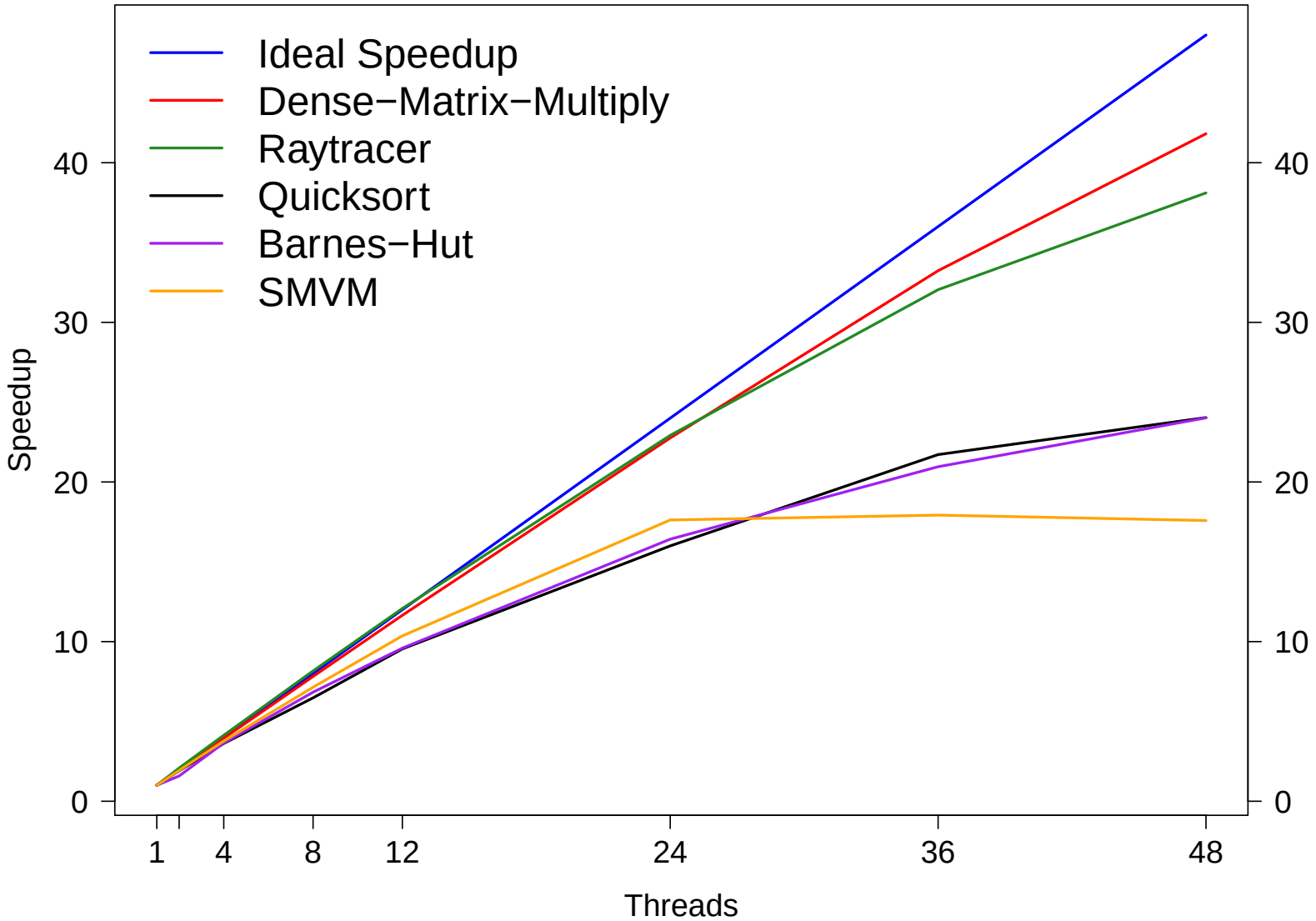
Manticore language & runtime

Manticore supports a parallel dialect of Standard ML.

- Side-effect free subset of SML
- NESL/Nepal/DPH style data-parallel arrays:
  `[|...|]` instead of `[...]`
- PThreads mapped to virtual processors (VProc)s
  - Pinned to a node or core
  - Use work-stealing to balance parallelism

Manticore benchmark speedups (AMD baseline)

We have best in class functional language parallel scalability.



NUMA allocation policies

Physical location is chosen when the page is touched based on the policy.[1]

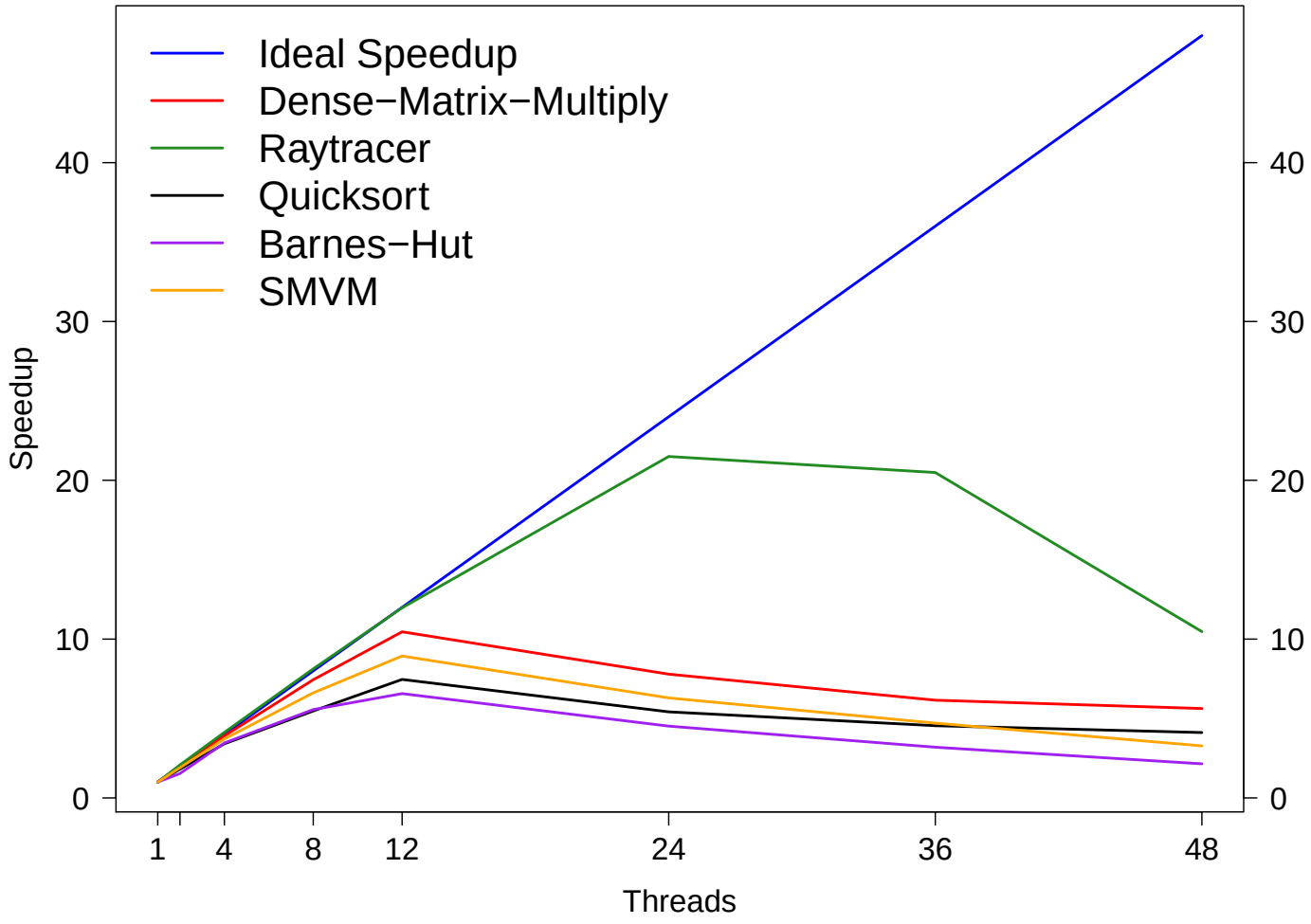
- Same-thread (default)
- Interleaved
- Preferred single location

[1] Unless direct NUMA allocation APIs are used in place of malloc/new/static declaration.

Benchmarks (AMD preferred0)

Don’t laugh!

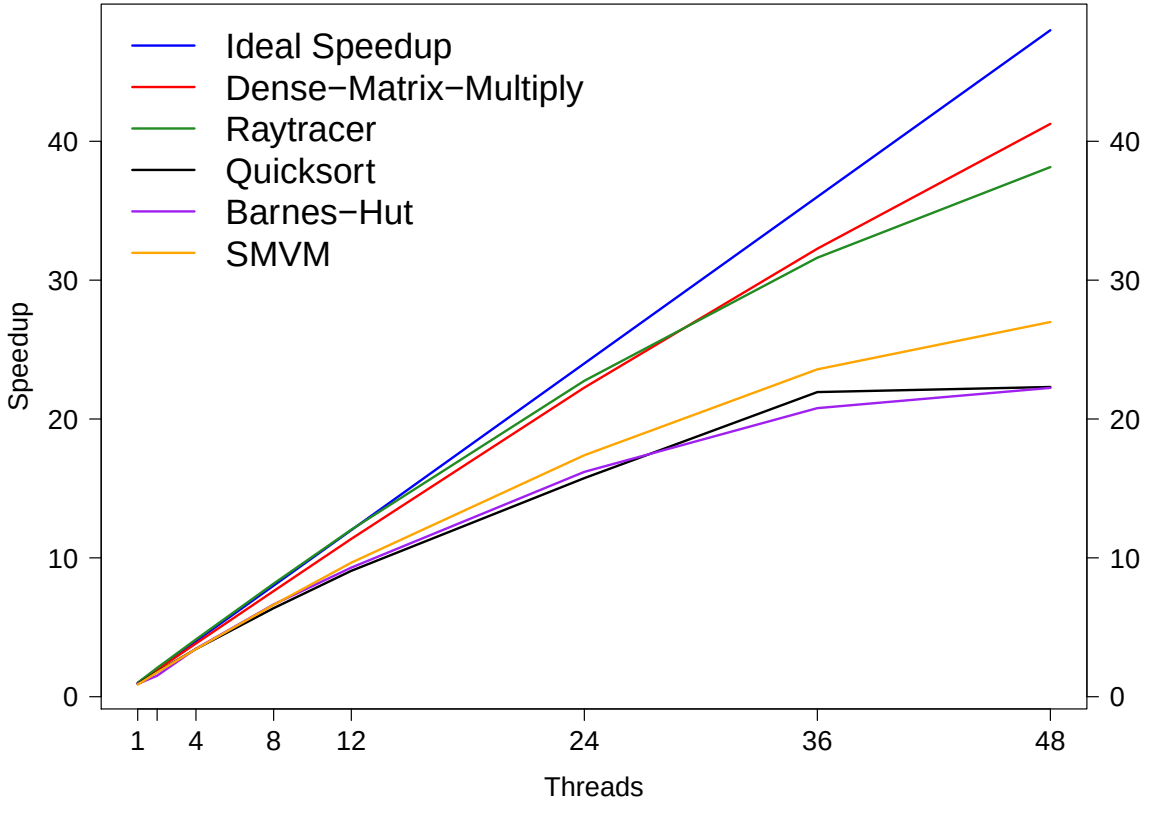
- Default behavior if you have a single-threaded allocator
- Scalability to 12 cores is good



Benchmarks (AMD interleaved)

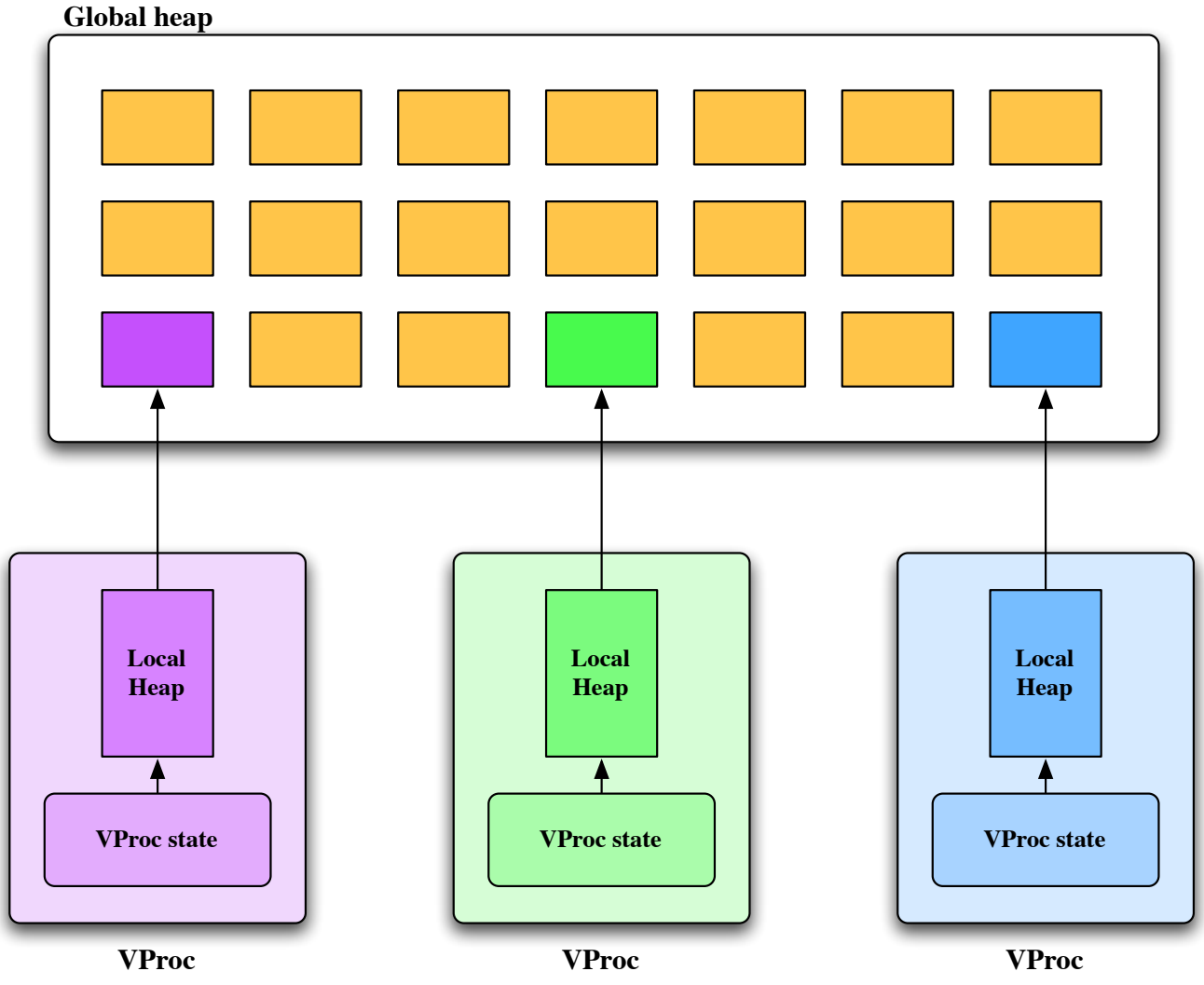
Interleaved is worse *unless* there is a piece of high-contention data.

Benchmark	vs. baseline
DMM	-1.3%
Raytracer	0%
Quicksort	-7.2%
Barnes-hut	-7.3%
SMVM	50%



Heap architecture

Initial goal: avoid synchronization and communication between VProcs.



Garbage collector highlights

- Based on a combination of Doligez-Leroy-Gonthier local heaps and Appel's SML/NJ collector
- Completely asynchronous minor collections
- Major collections only synchronize if a new page needs to be allocated
- Parallel global collection
- Pointer invariant avoids write barriers; mechanisms used to reduce data promotion

Emphasis on parallelism lead directly to NUMA locality

Conclusion

Thread placement and memory allocation locations impact the performance of runtimes on NUMA machines.

- More details on our garbage collector are available in the paper.

Thanks to my collaborators, reviewers, and the Intel Manycore Testing Laboratory. The AMD machine used for the benchmarks was supported by National Science Foundation Grant CCF-1010568 and this work is additionally supported in part by National Science Foundation Grant CCF-0811389.