Performance Evaluation of the 48-core SCC Processor

Aparna Chandramowlishwaran, Richard Vuduc

– Georgia Institute of Technology

Kamesh Madduri

– Lawrence Berkeley National Laboratory

LBNL ICCS 2011 Workshop

aparna@cc.gatech.edu

Performance Evaluation of the 48-core SCC Processor's On-chip Interconnect

Aparna Chandramowlishwaran, Richard Vuduc

– Georgia Institute of Technology

Kamesh Madduri

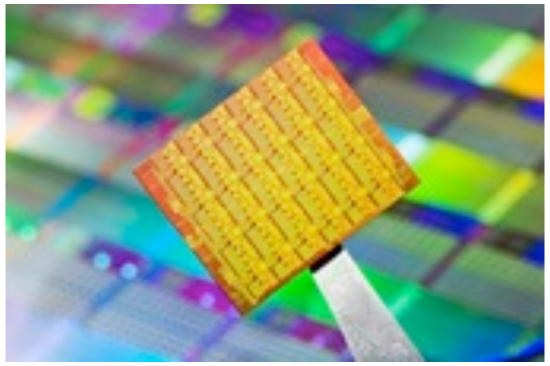
– Lawrence Berkeley National Laboratory

LBNL ICCS 2011 Workshop

aparna@cc.gatech.edu

Talk Summary

- We characterize the SCC on-chip interconnection network with micro-benchmarks
  - Observed point-to-point latency, bandwidth
  - Performance model
- We present new collective communication algorithms for the SCC
  - Broadcast 22x faster than prior approach
  - Reduce 6.4x faster than prior approach

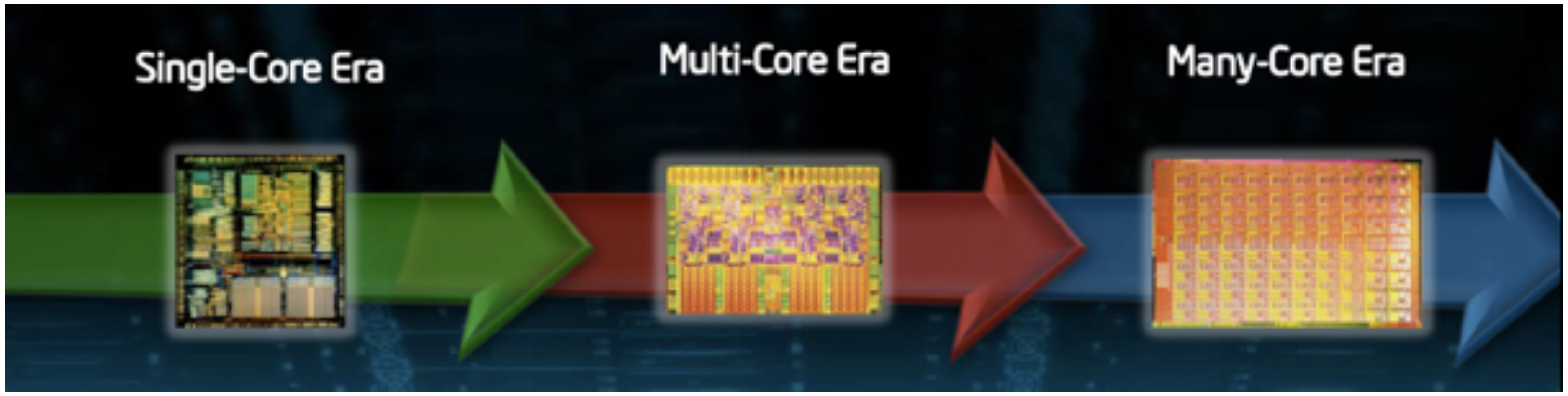


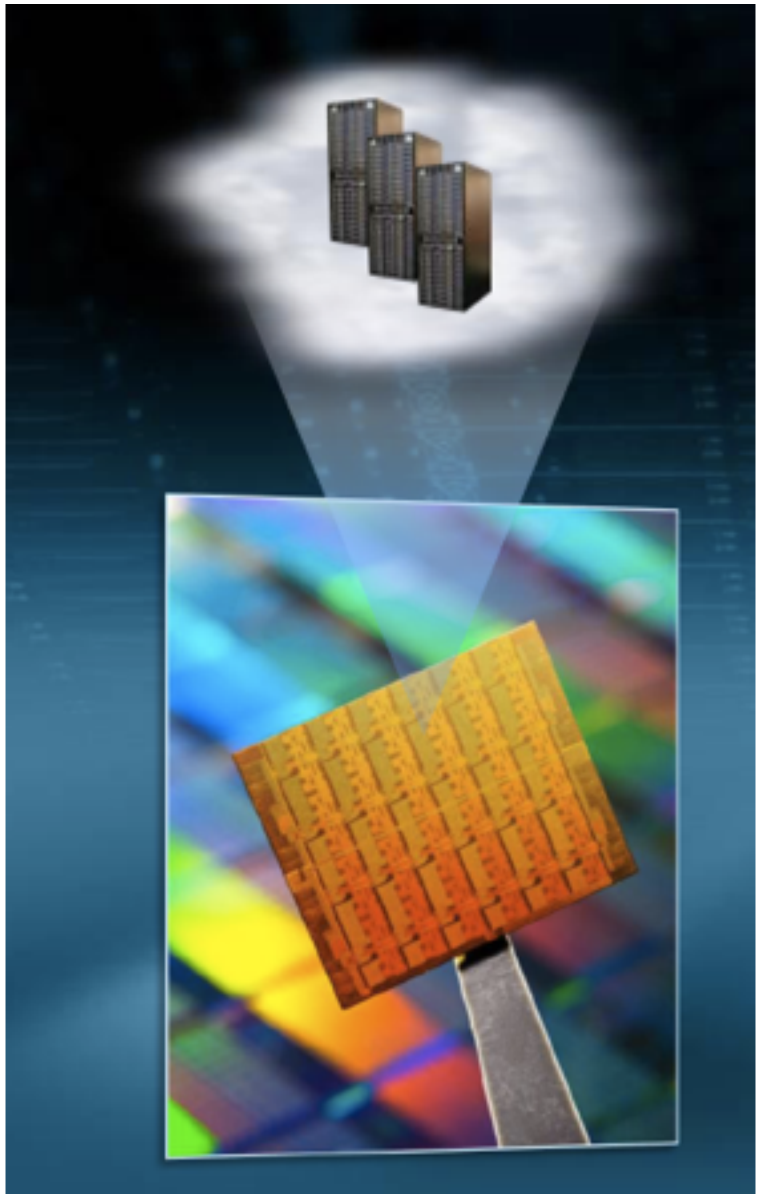
- SCC access to Georgia Tech provided through Intel's MARC (Many-Core Applications Research Community) initiative

Outline

- **Overview of Intel's *experimental* Single-chip Cloud Computer (SCC)**
- Interconnection Network and Messaging
- Microbenchmarks and Observations
- Collective Communication Algorithms
- Conclusions

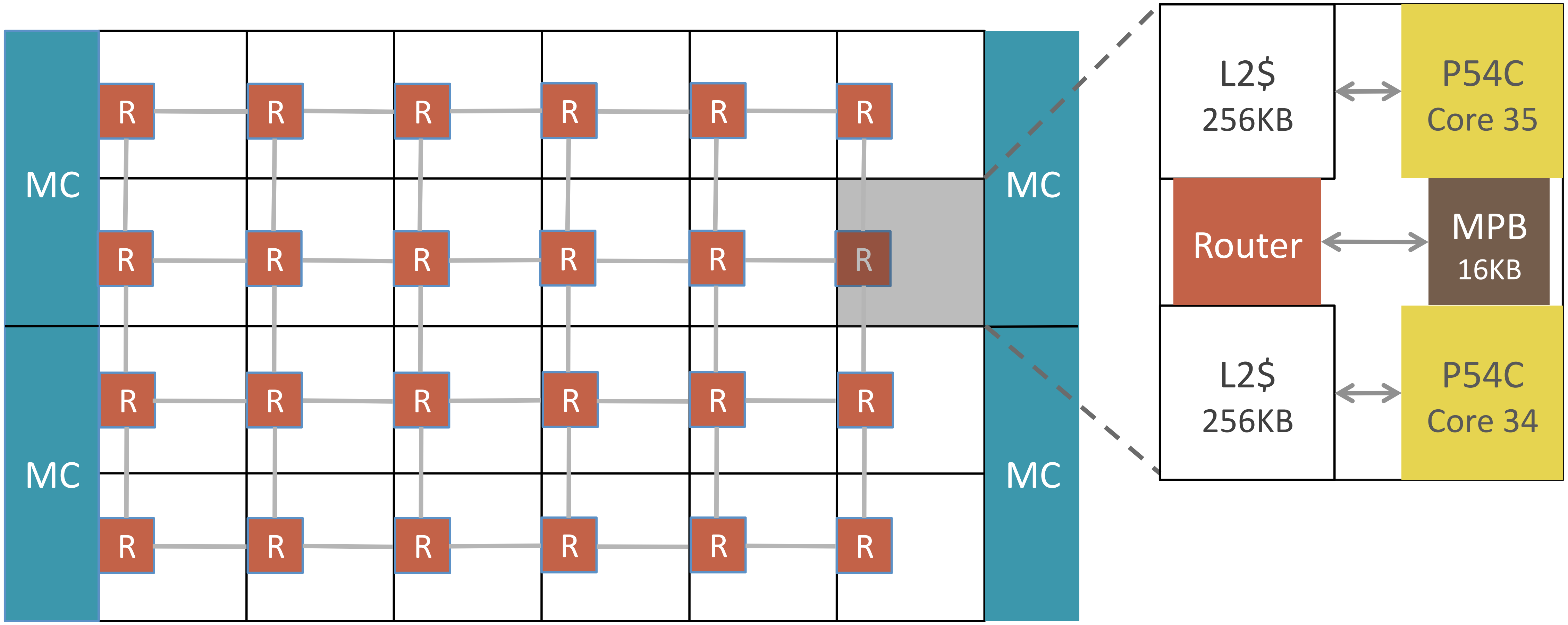
Single-chip Cloud Computer





- Experimental 48-core CPU
- Fine-grained power management
- Enables exploration of alternative 'scalable' programming models by removing hardware cache coherence

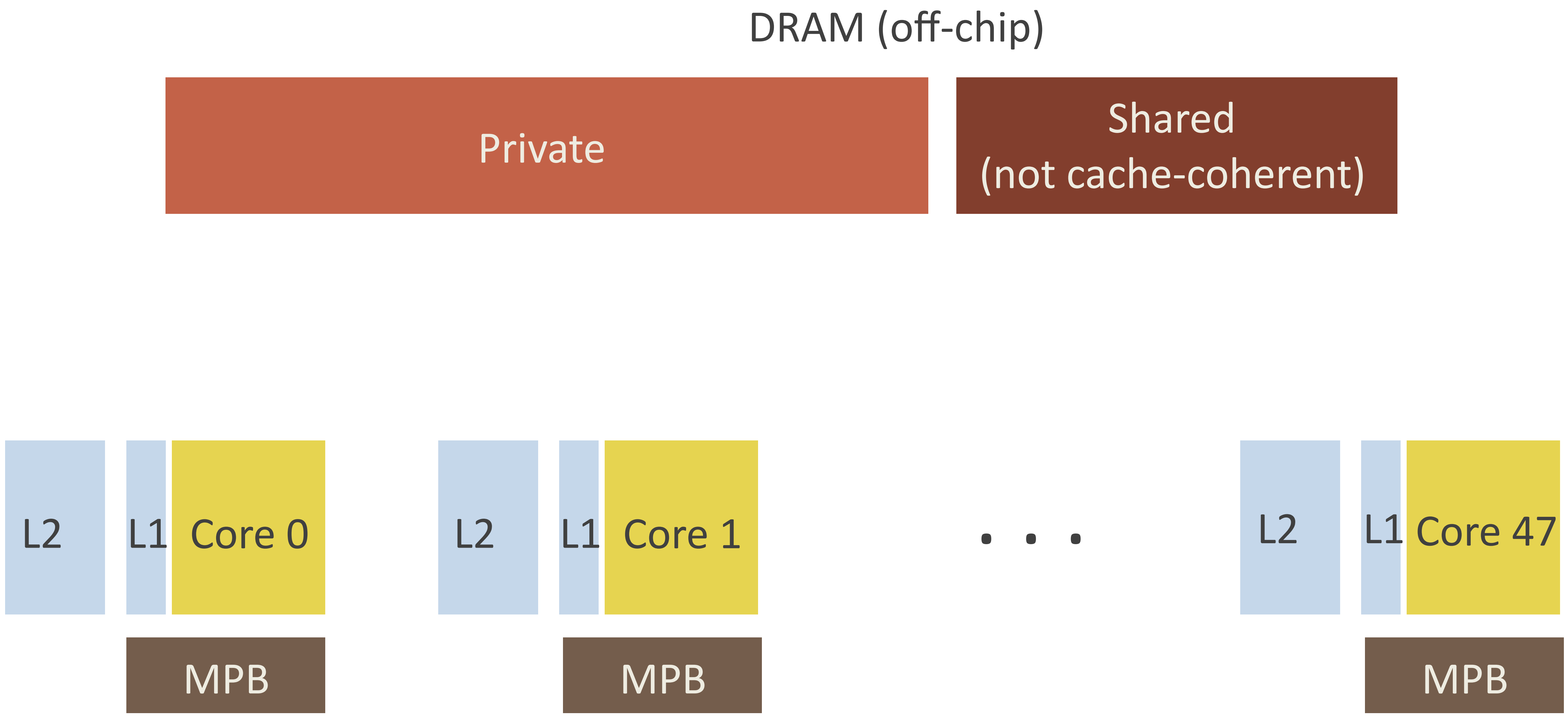
Source: Justin Rattner, "Intel SCC" announcement



- 2 P54C cores per tile
- 4 Integrated DDR3 memory controllers
- 2D mesh interconnect, 64 GB/s link bandwidth
- 8 KB *globally addressable* “Message Passing Buffer” per core
- 16 KB L1, 256 KB L2 cache per core

Outline

- Overview of Intel's *experimental* Single-chip Cloud Computer (SCC)
- **Interconnection Network and Messaging**
- Microbenchmarks and Observations
- Collective Communication Algorithms
- Conclusions



DRAM (off-chip)

Private

Shared
(not cache-coherent)

L2

L1

Core 0

MPB

L2

L1

Core 1

MPB

. . .

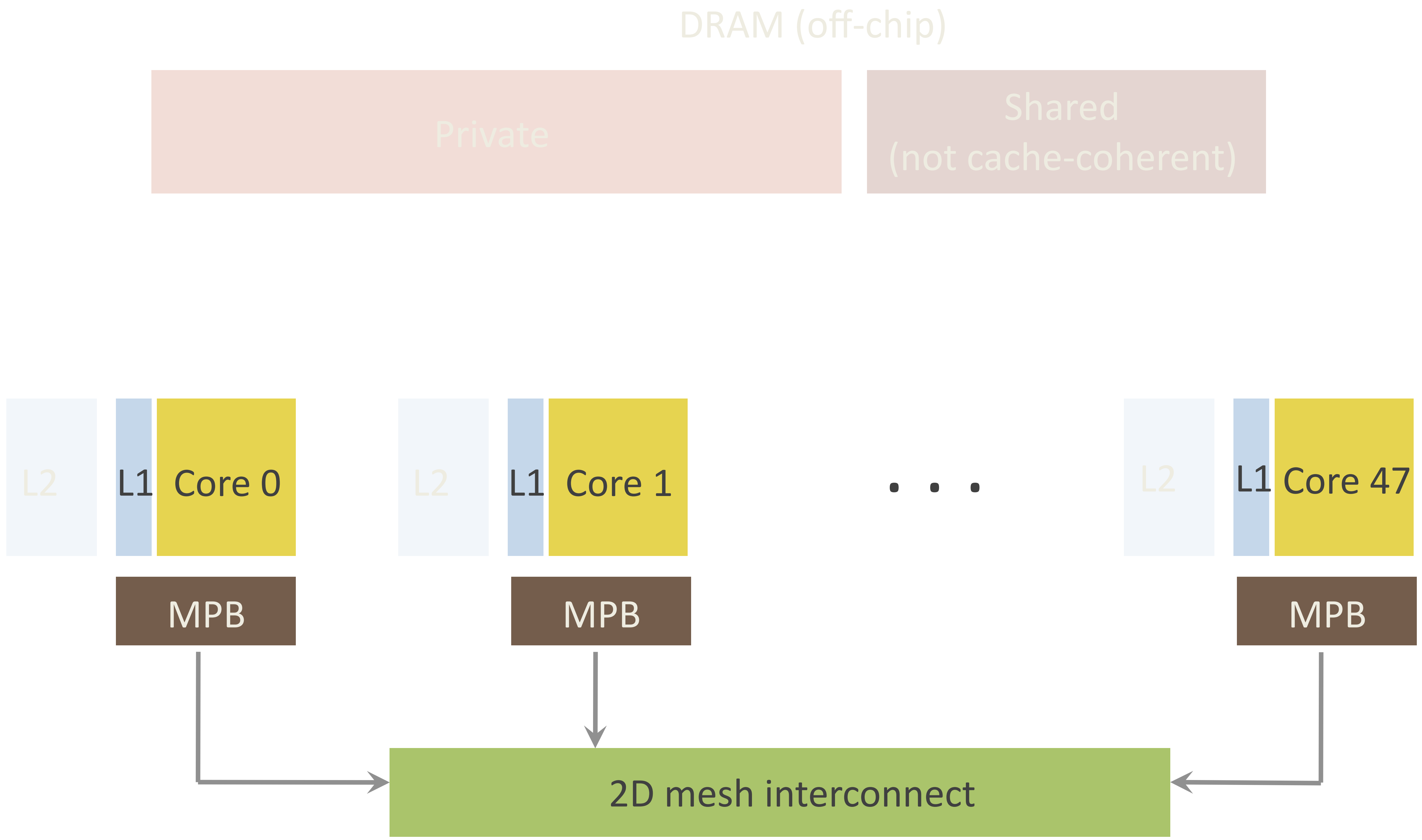
L2

L1

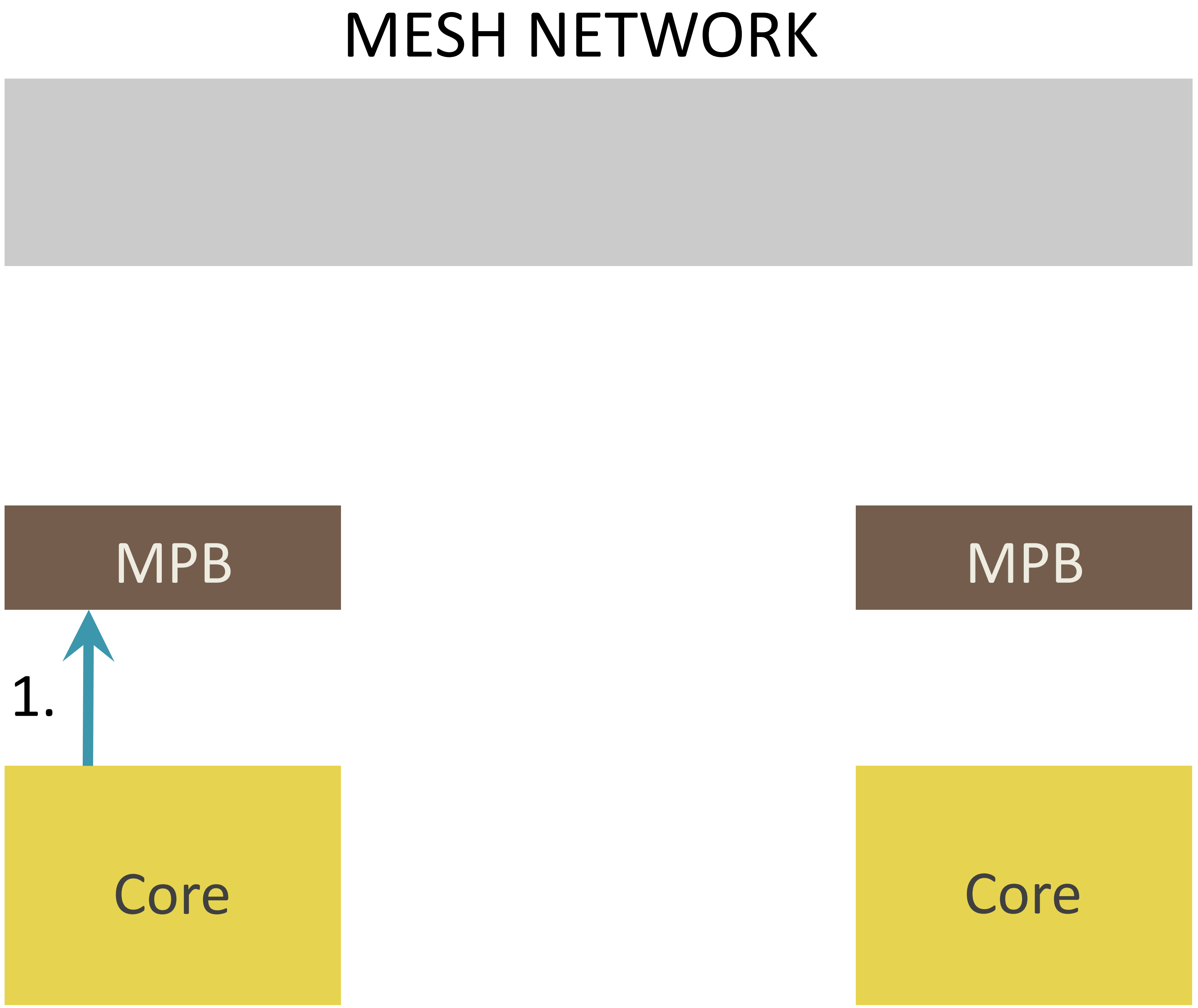
Core 47

MPB

SCC Memory Hierarchy

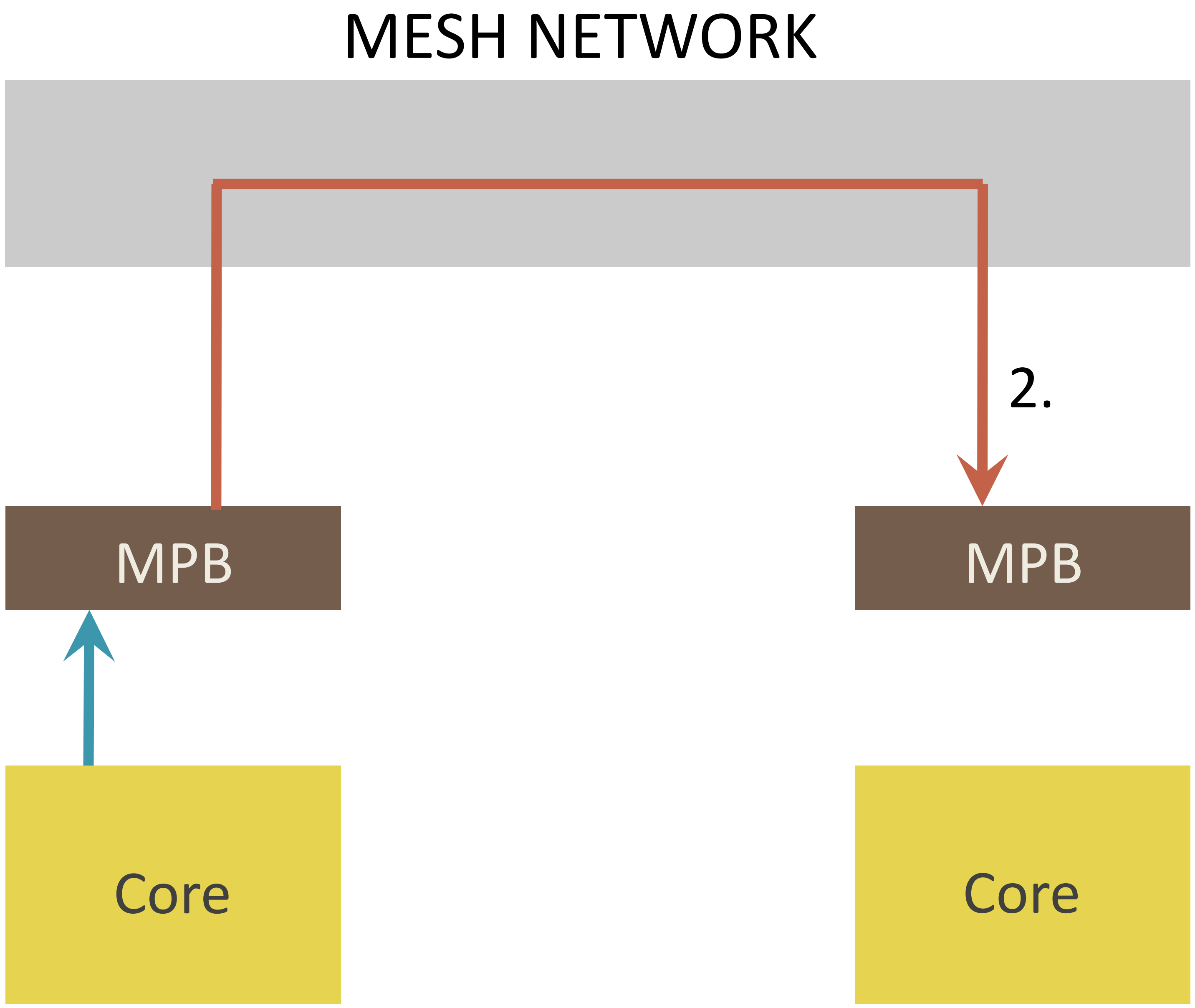


SCC Memory Hierarchy



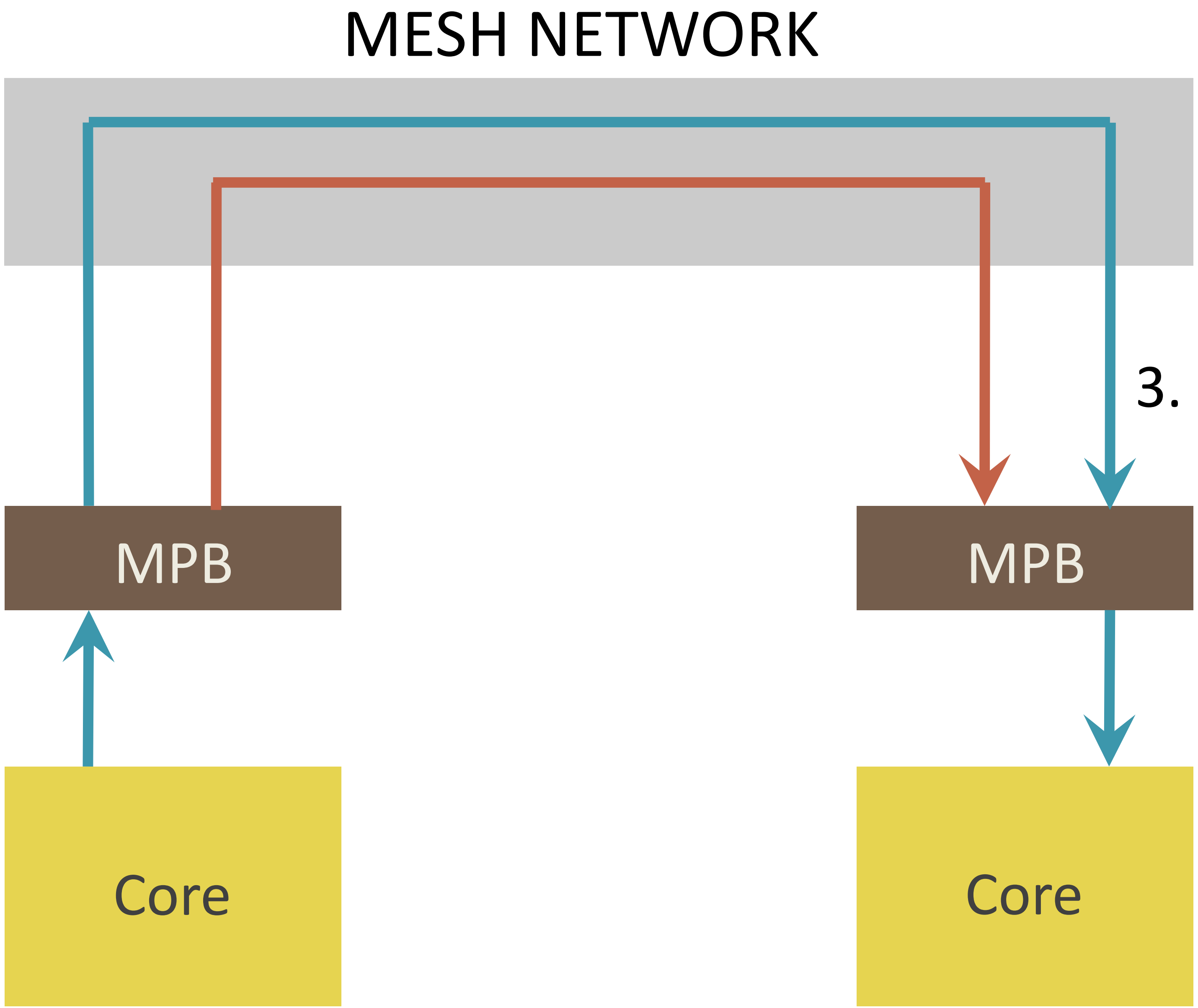
Synchronous two-sided messaging

1. Copy: local memory to MPB



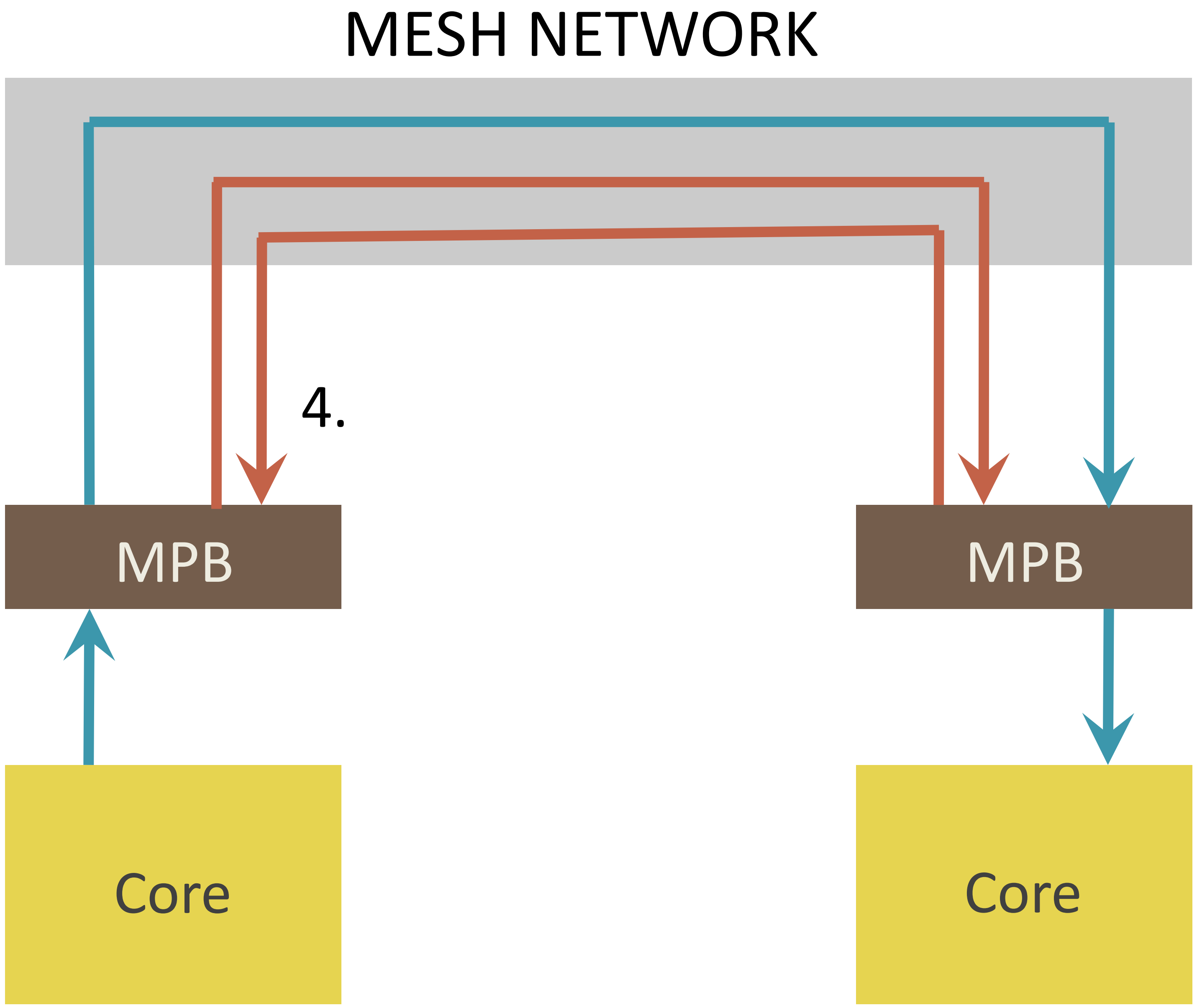
Synchronous two-sided messaging

2: Signal: data ready



Synchronous two-sided messaging

3: Copy: remote MPB to local MPB



Synchronous two-sided messaging

4: Signal: Acknowledgement

Programming models for SCC

- Use one-sided get/put, expose MPB to the programmer
  - expert programmer, fine-grained control
- Two-sided synchronous send/recv with messaging details hidden from programmer
  - MPI-like
- RCCE: communication environment developed by Intel
  - Supports both these models

Goals and Contributions

- Evaluation of the on-chip interconnection network
  - Microbenchmarks to identify cost of various messaging building blocks
  - Performance model from observations
- Design optimized collective communication schemes, assuming a high-level “MPI”-like programmer interface

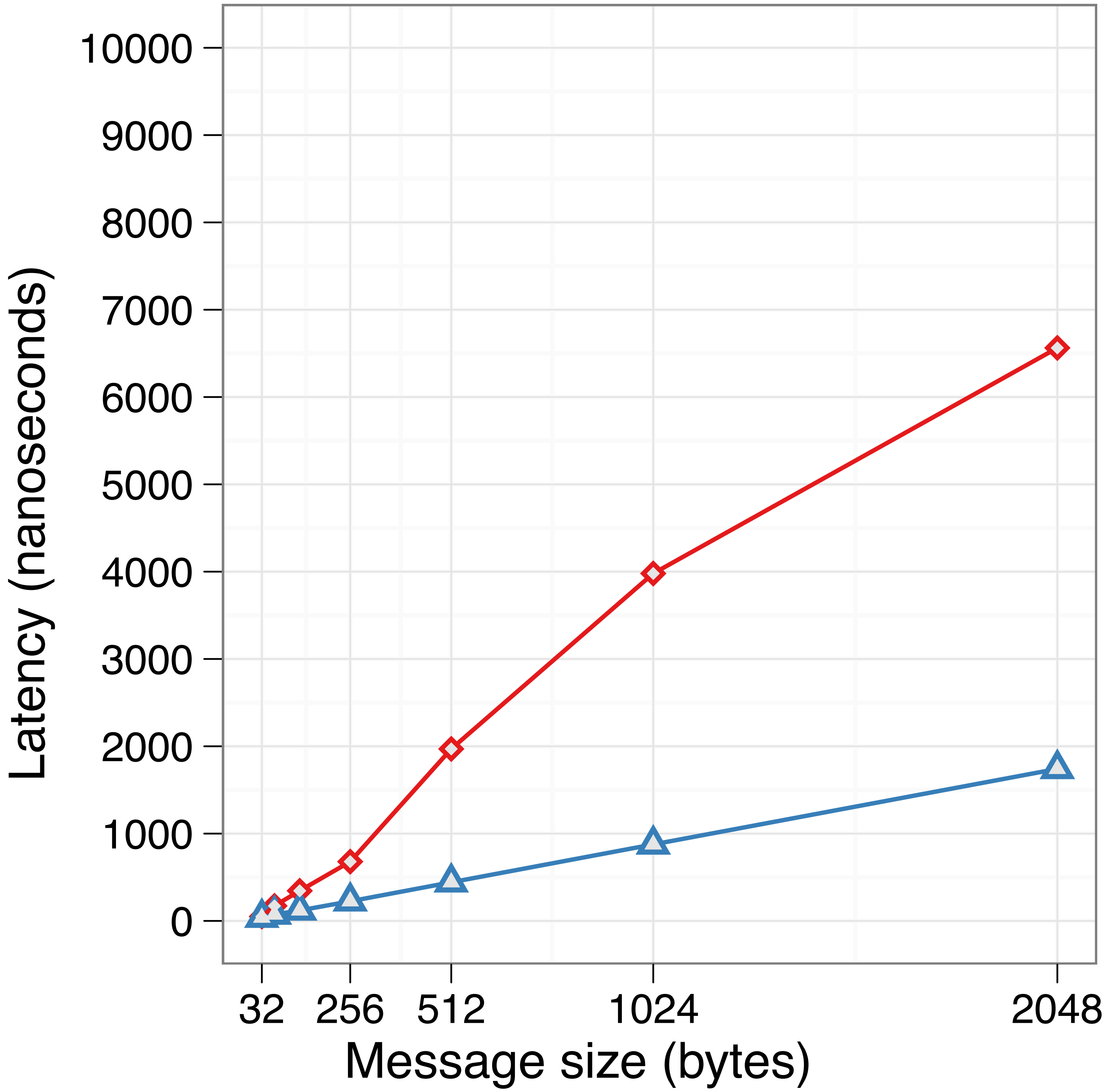
Outline

- Overview of Intel's *experimental* Single-chip Cloud Computer (SCC)
- Interconnection Network and Messaging
- **Microbenchmarks and Observations**
- Collective Communication Algorithms
- Conclusions

Experimental Setup

- Configuration: Cores at 533 MHz, Router at 800 MHz, DDR3 800 memory
- L2 disabled
- DRAM to MPB copy step not timed
- 8KB/core limit on on-chip messaging
  - Pipeline longer messages

Local MPB read performance

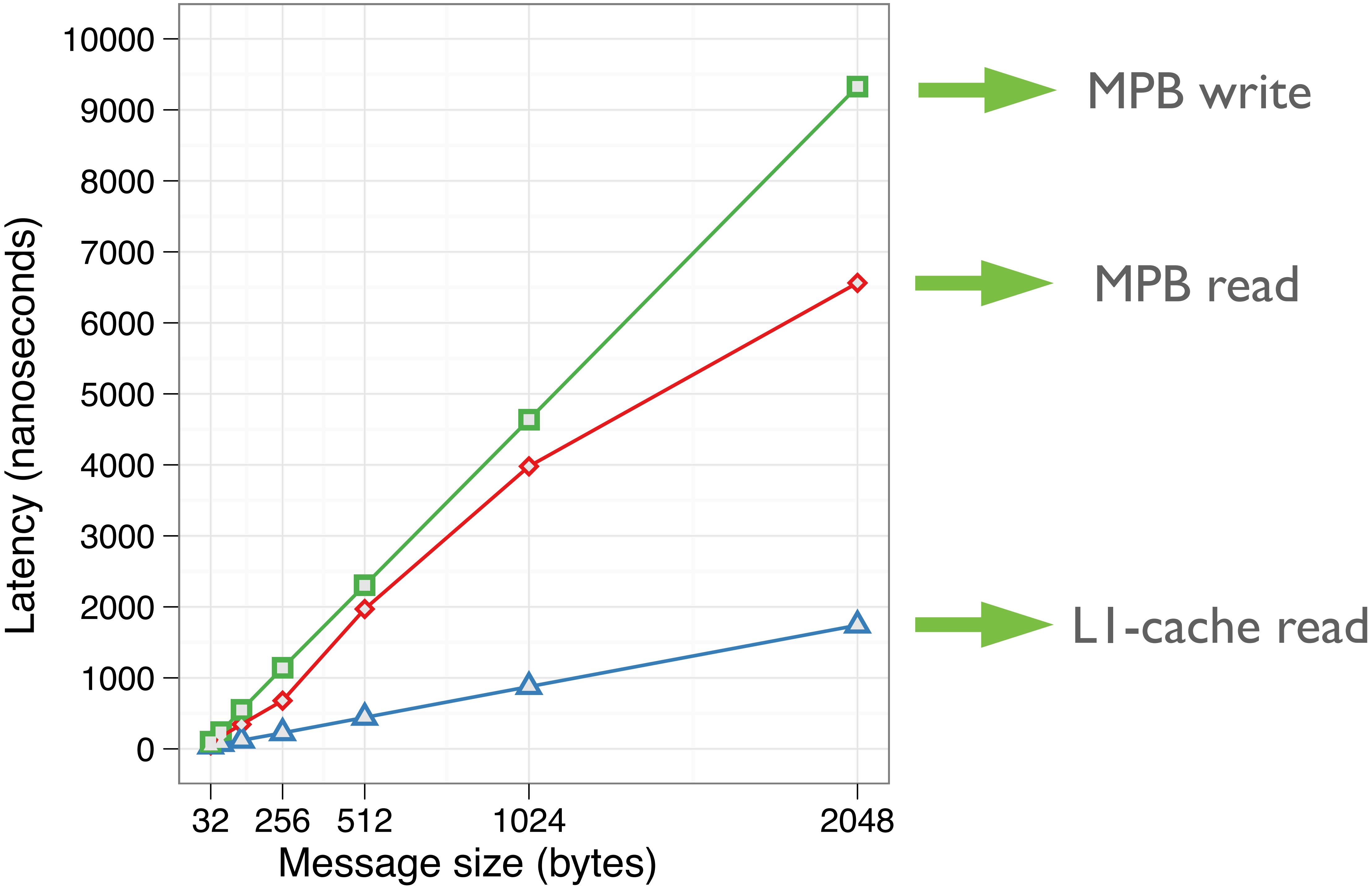


MPB read

L1-cache read

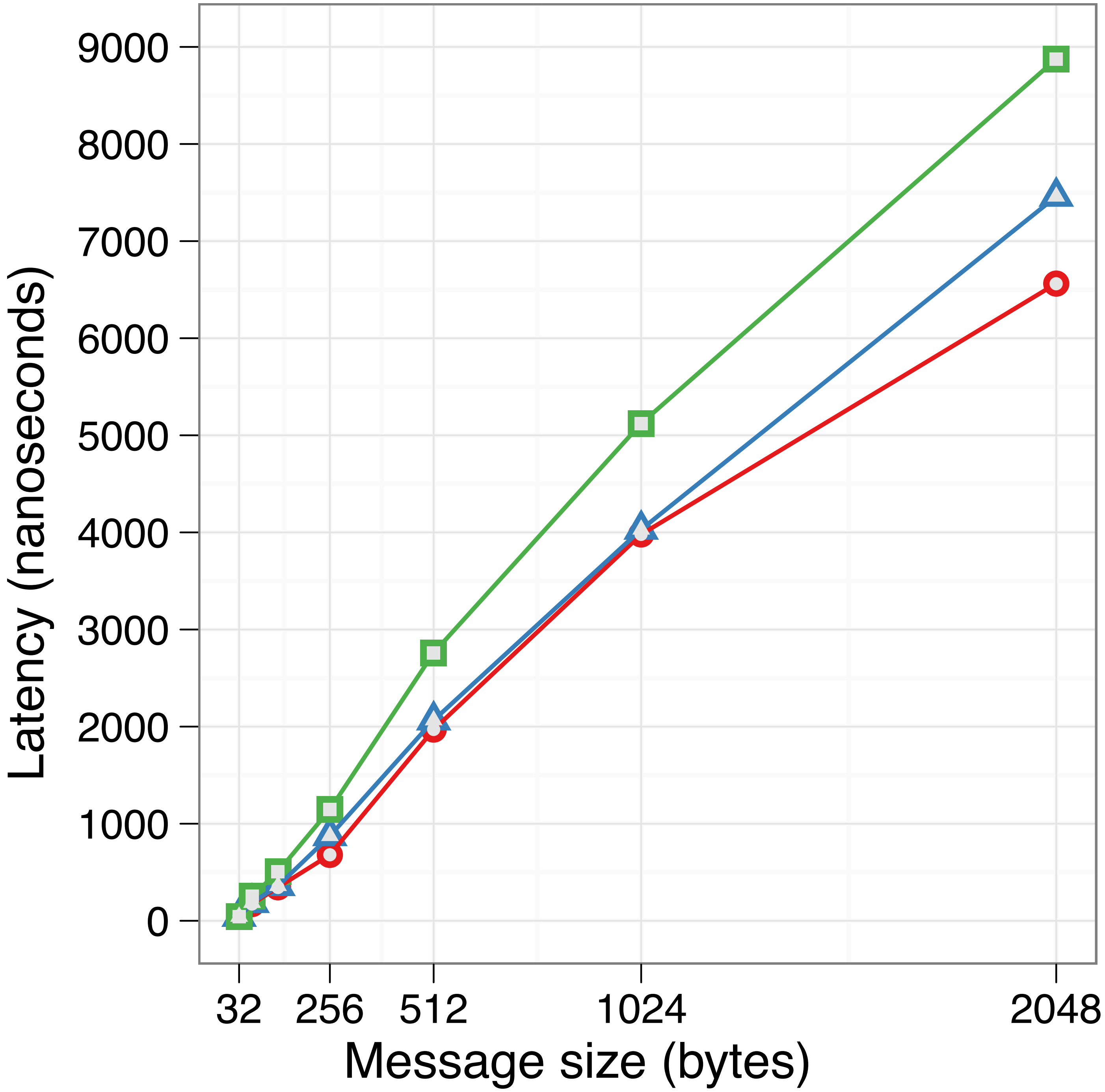
MPB read bandwidth ~ 4x lower than L1 bandwidth

Local MPB write performance



Writes to local MPB ~ 1.4x more expensive than read

Remote MPB read vs Message size



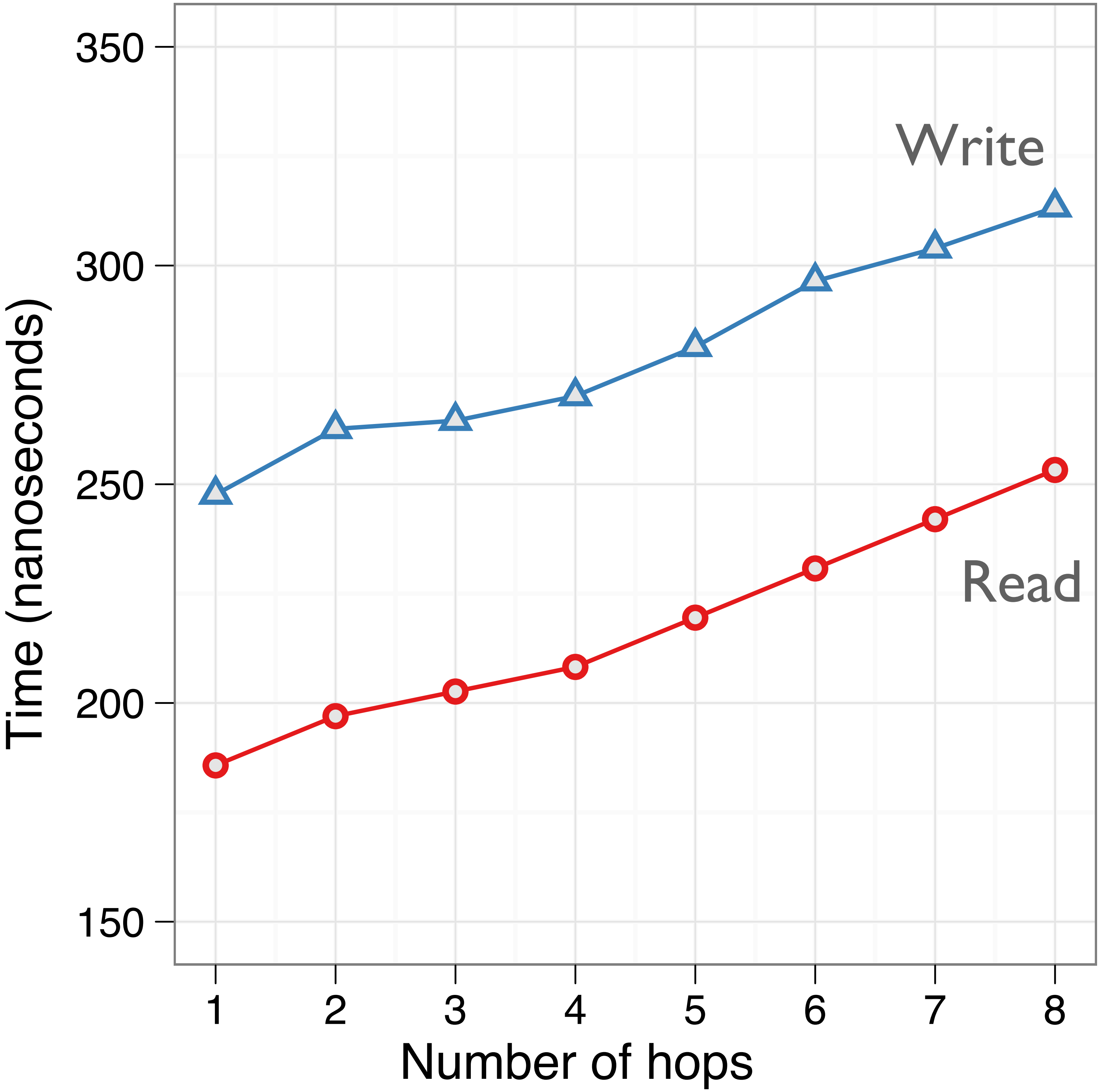
→ farthest tile

→ neighboring tile

→ same tile

Remote read time increments proportional to number of hops

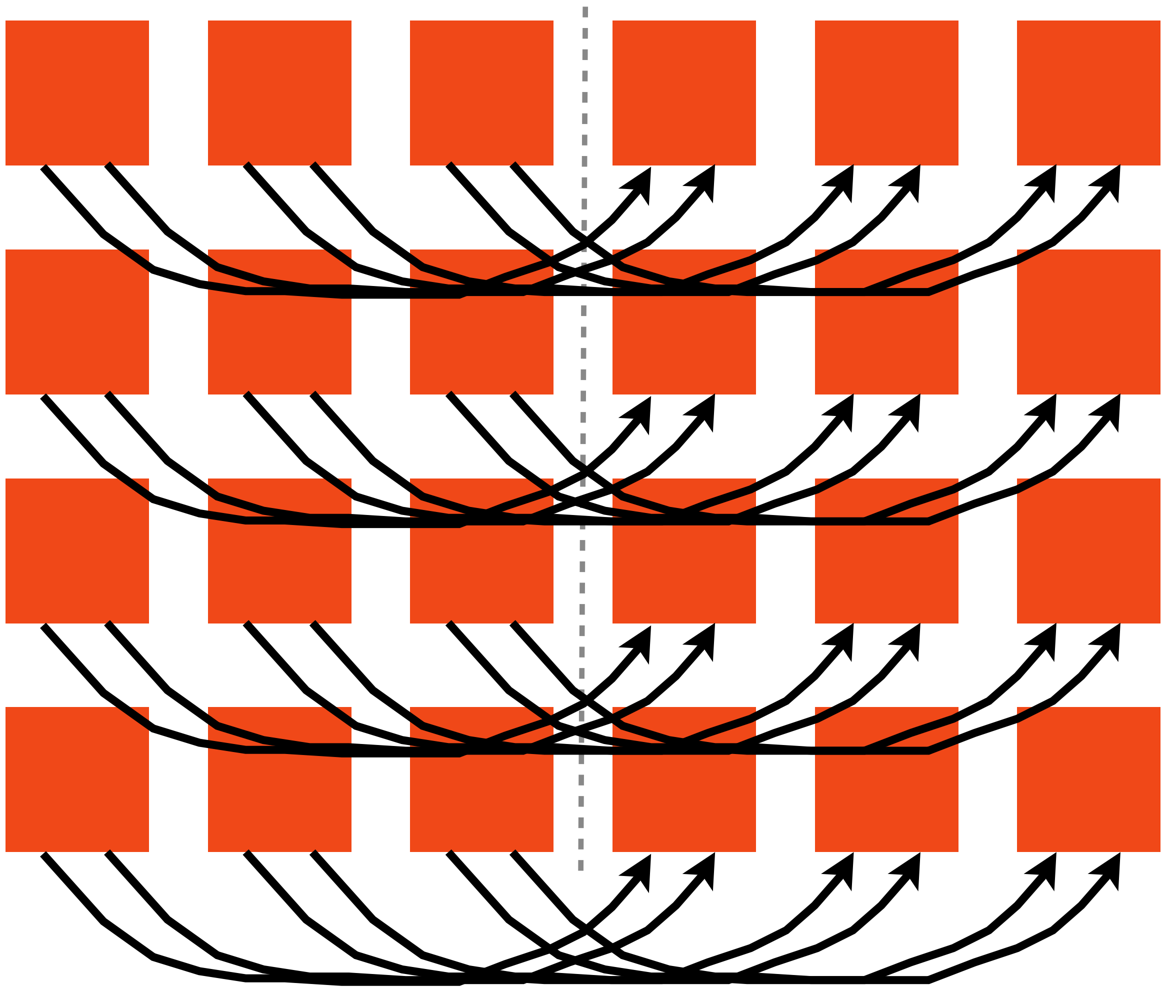
Remote MPB read/write vs Hop count



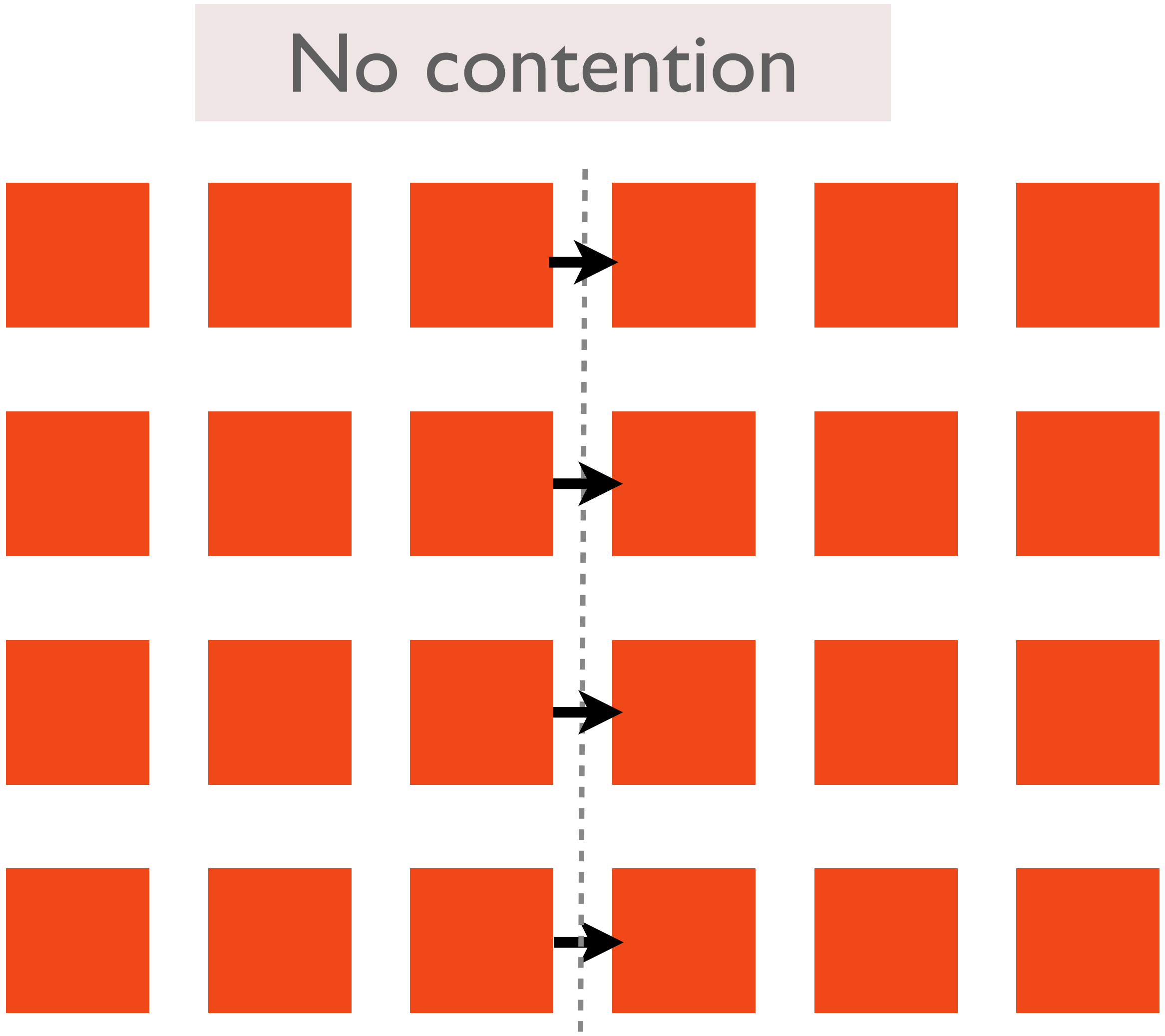
64 B message. Slope gives router latency: ~ 5 ns.

Measuring Bisection bandwidth

2048 B message, comparison to "no contention" case



19.93 microseconds



19.2 microseconds

Performance Model & Observations

Local MPB read	(# of cache lines) * 105 ns
Remote MPB read	(# of cache lines) * (105 ns + (# of hops) * 5 ns)
Local MPB write	(# of cache lines) * 145 ns
Remote MPB write	(# of cache lines) * (145 ns + (# of hops) * 5 ns)

Observed Bandwidths (MB/s)

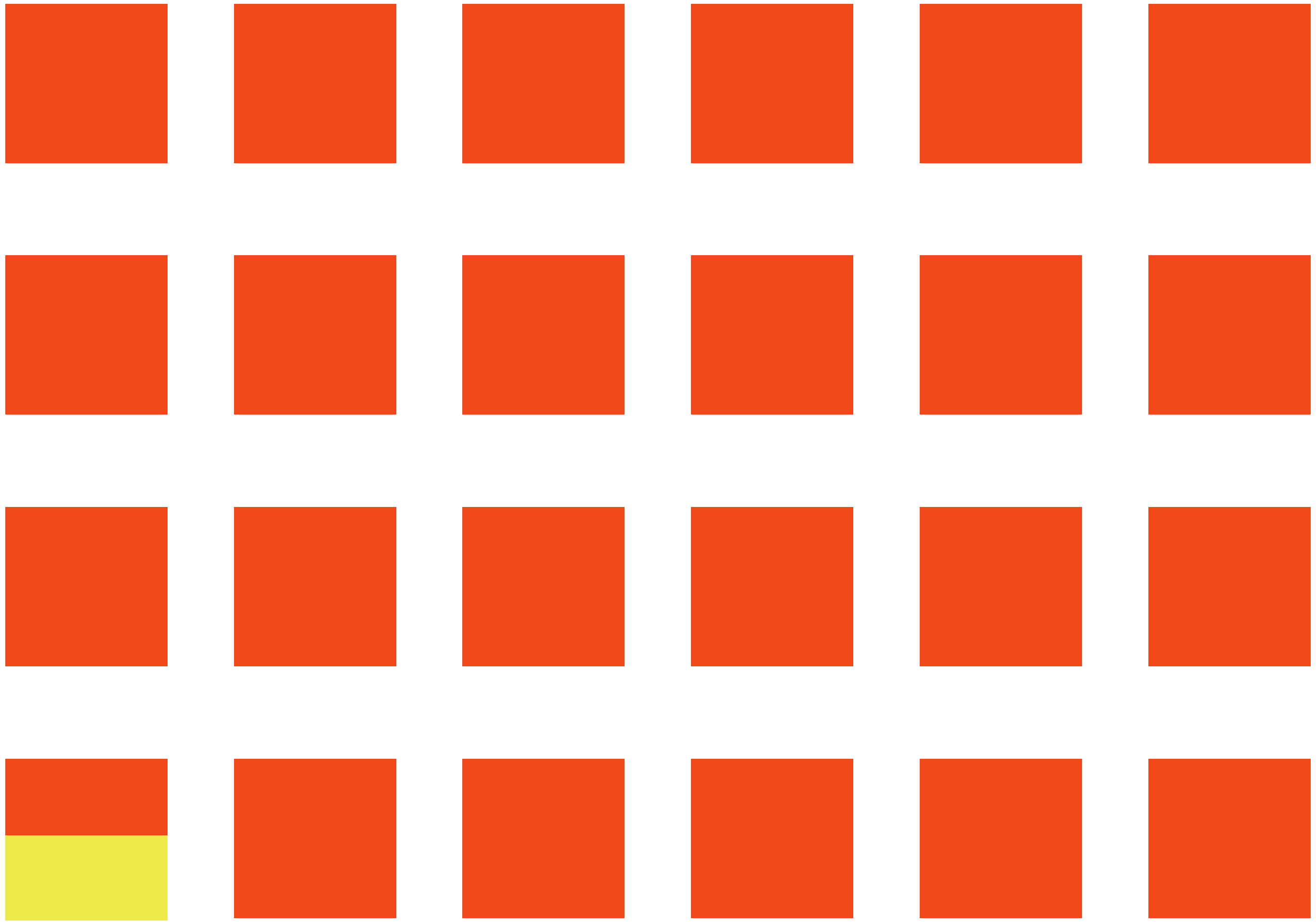
Local MPB read	305
Remote MPB (8-hop) read	220
Local MPB write	220
Remote MPB (8-hop) write	173
L1	1220
DRAM/core (1 core)	160
DRAM/core (12 cores)	130

Outline

- Overview of Intel's *experimental* Single-chip Cloud Computer (SCC)
- Interconnection Network and Messaging
- Microbenchmarks and Observations
- **Collective Communication Algorithms**
- Conclusions

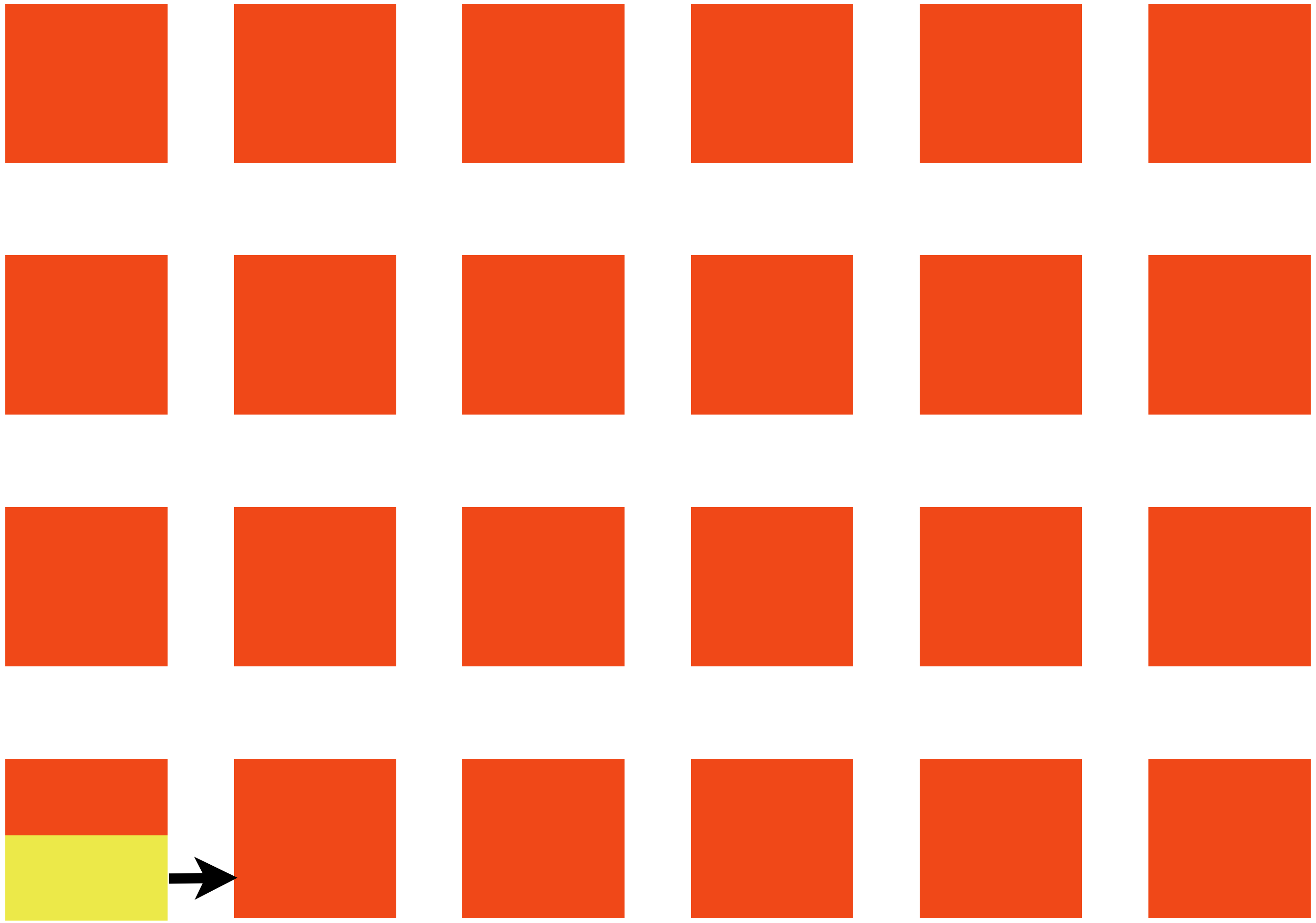
Designing collectives

- Performance model guides design of optimal scheme
- Two case studies:
  - Broadcast
  - Reduce



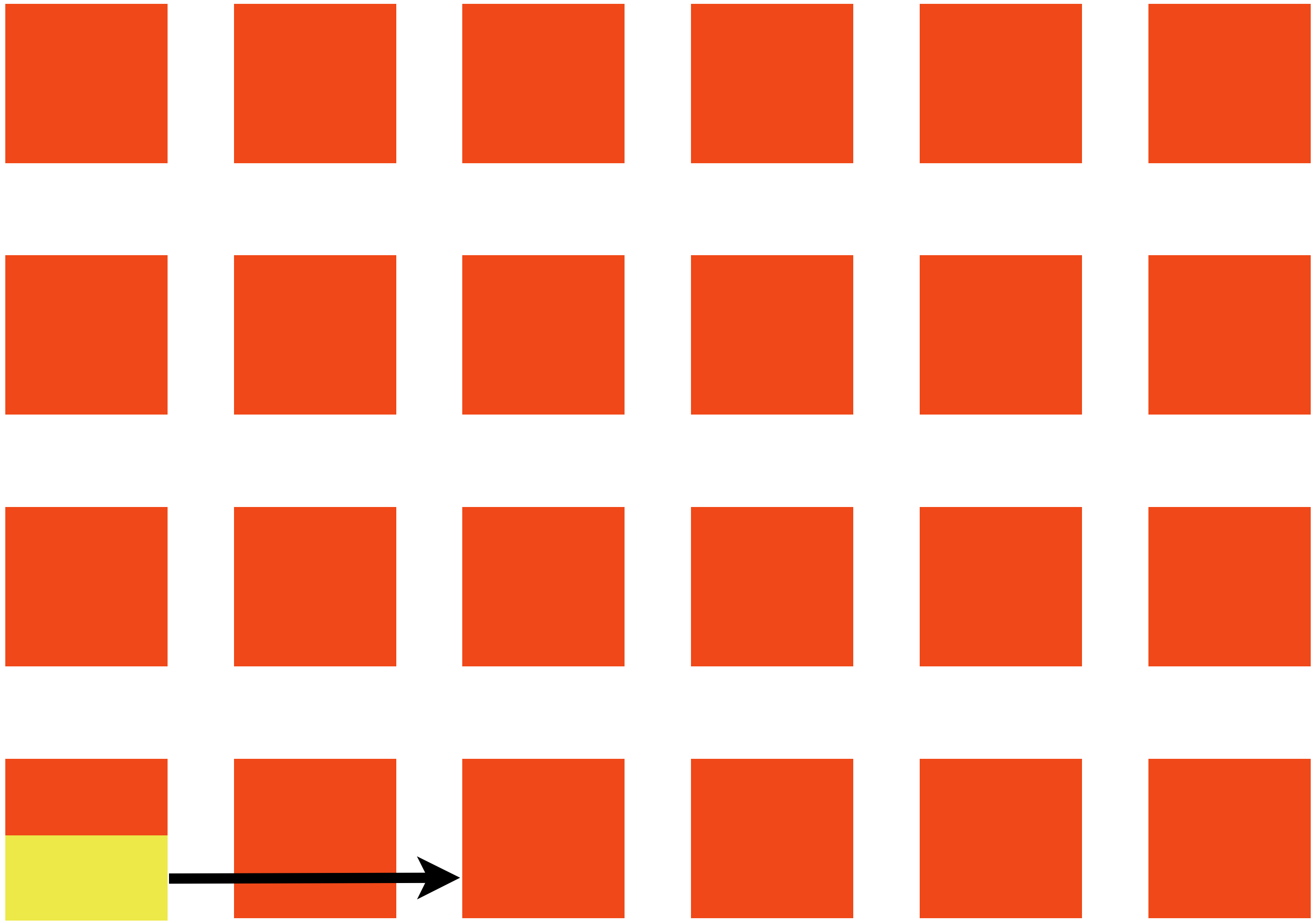
Broadcast algorithm

Naive: *put*-based approach



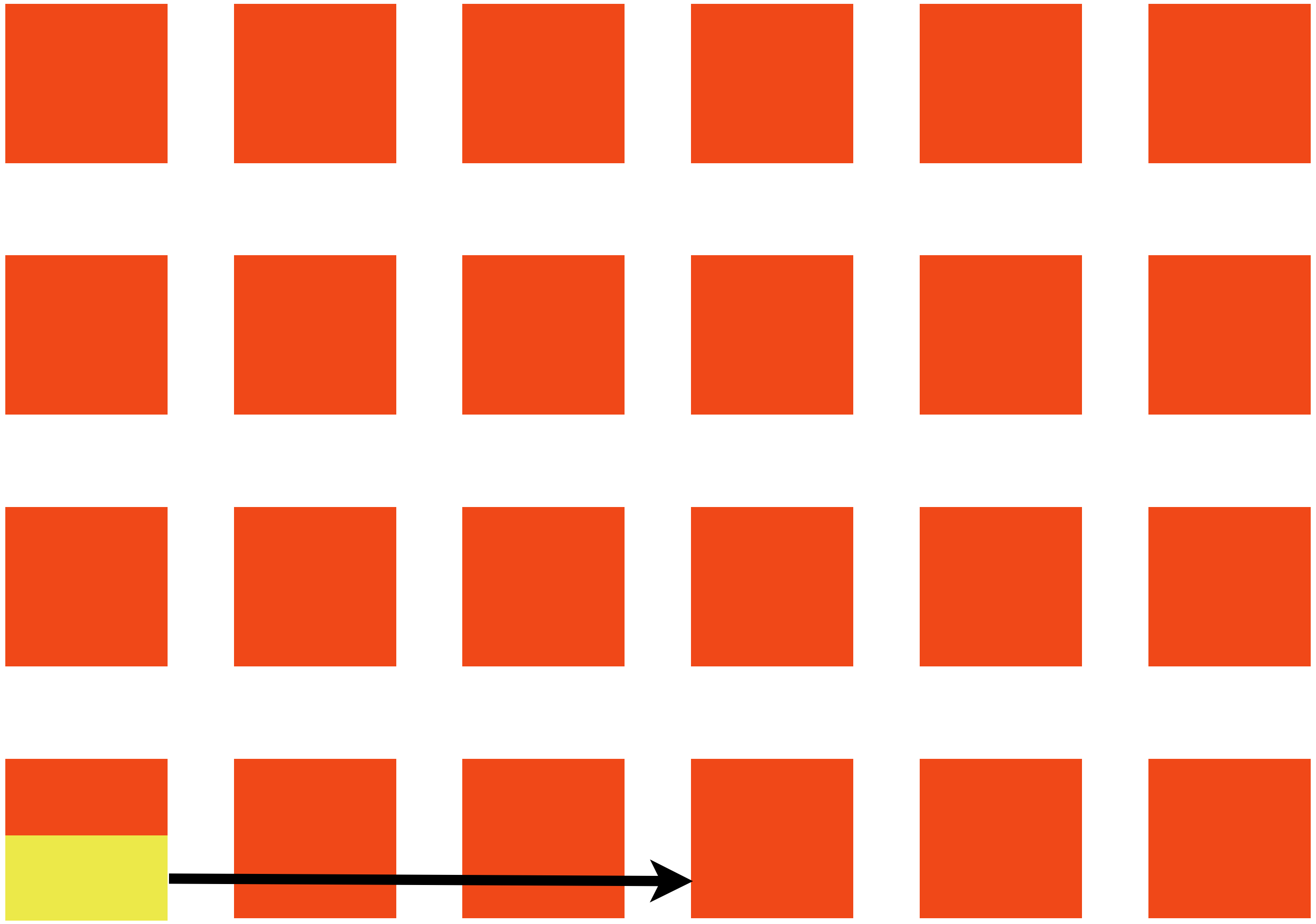
Broadcast algorithm

Core 0 writes to remote MPB's



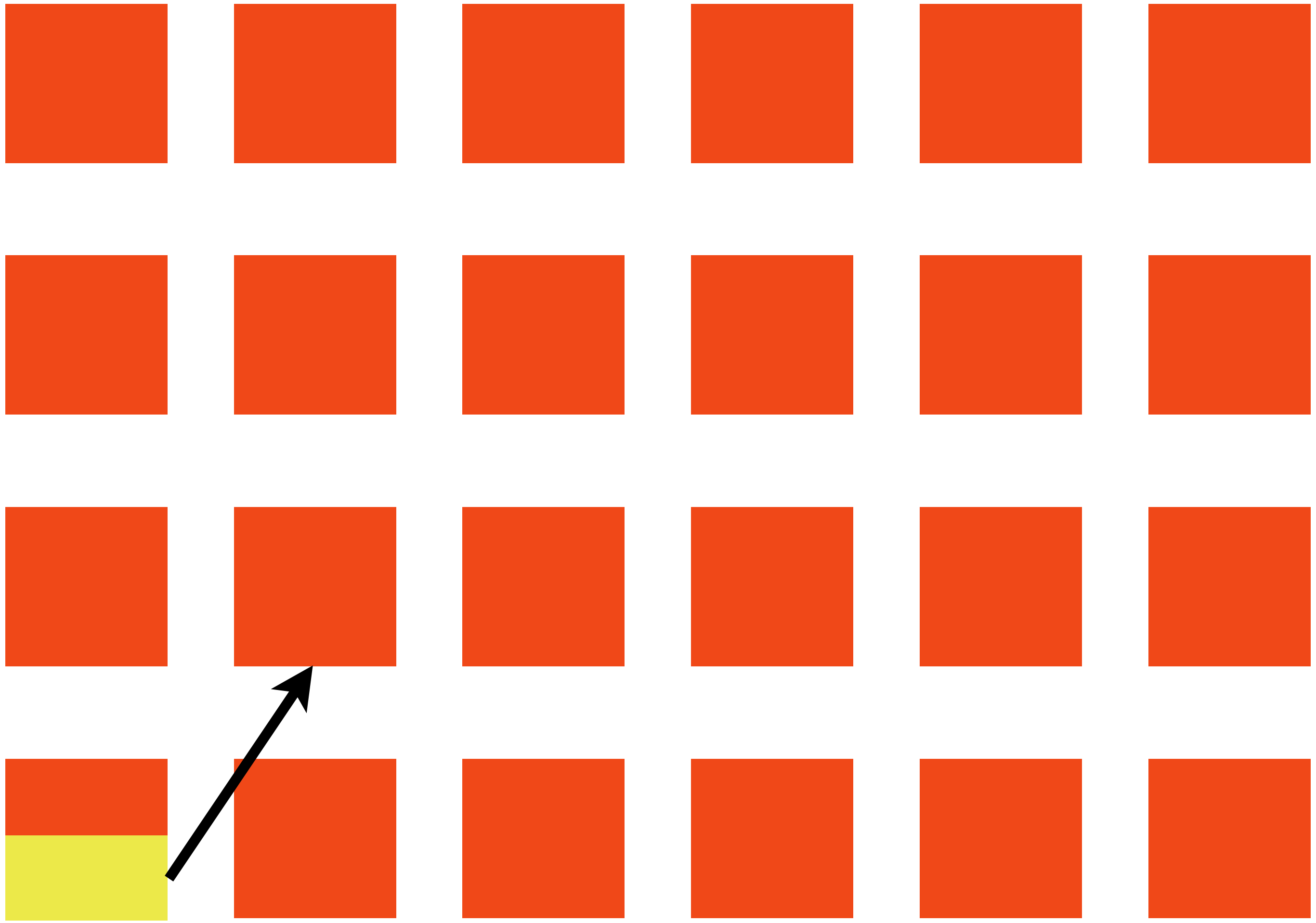
Broadcast algorithm

Core 0 writes to remote MPB's



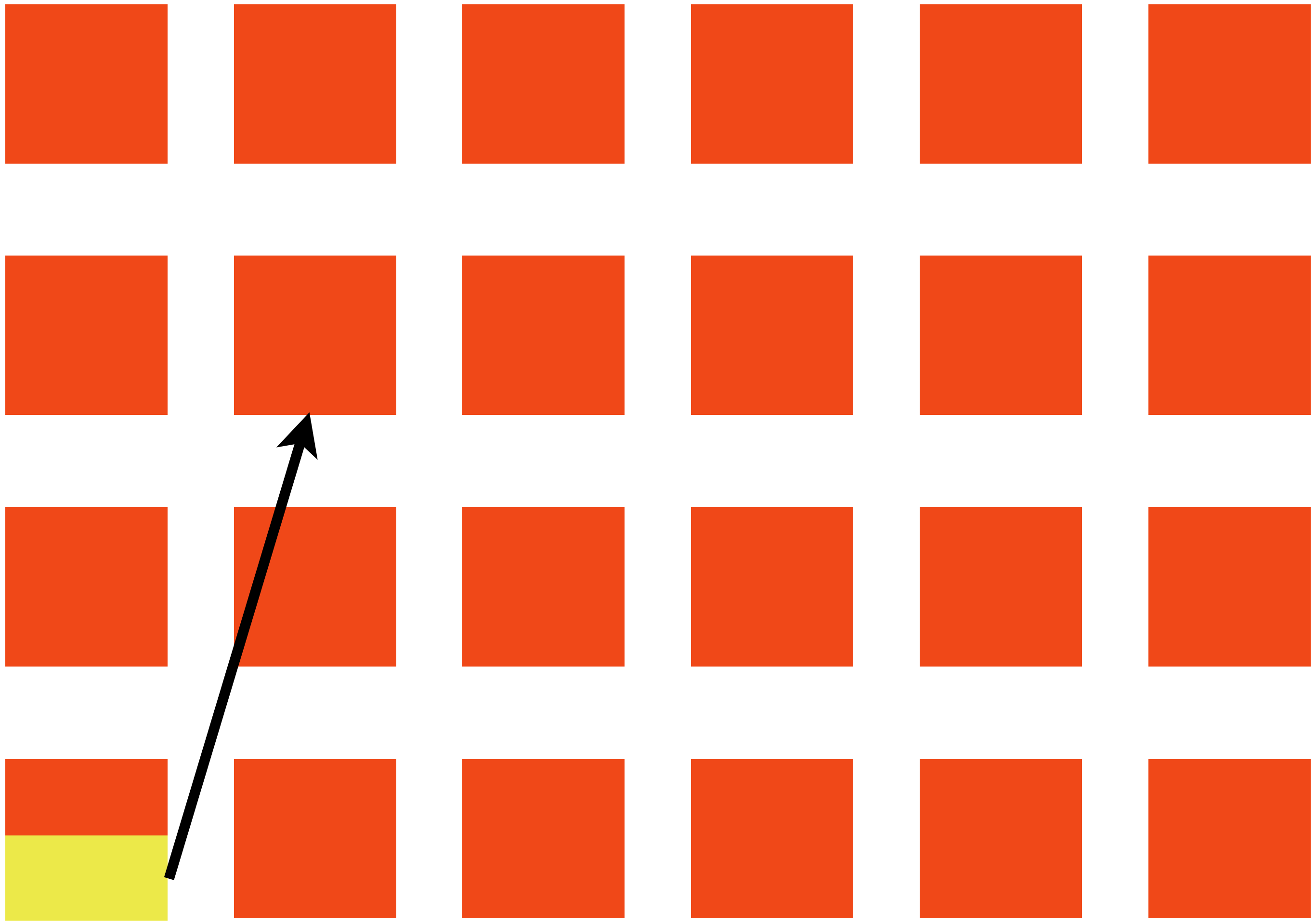
Broadcast algorithm

Core 0 writes to remote MPB's



Broadcast algorithm

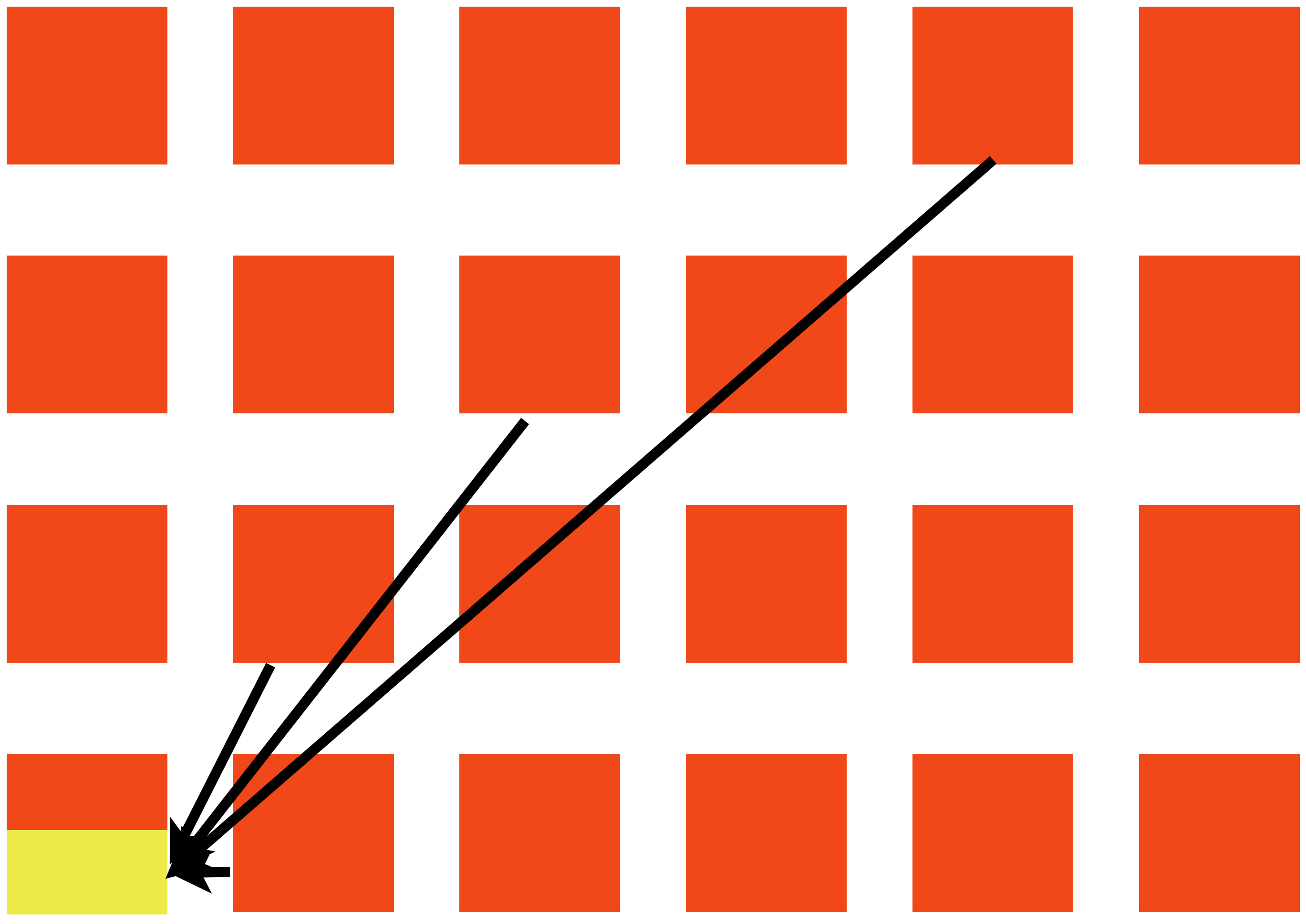
Core 0 writes to remote MPB's



p-1 steps

Broadcast algorithm

Core 0 writes to remote MPB's

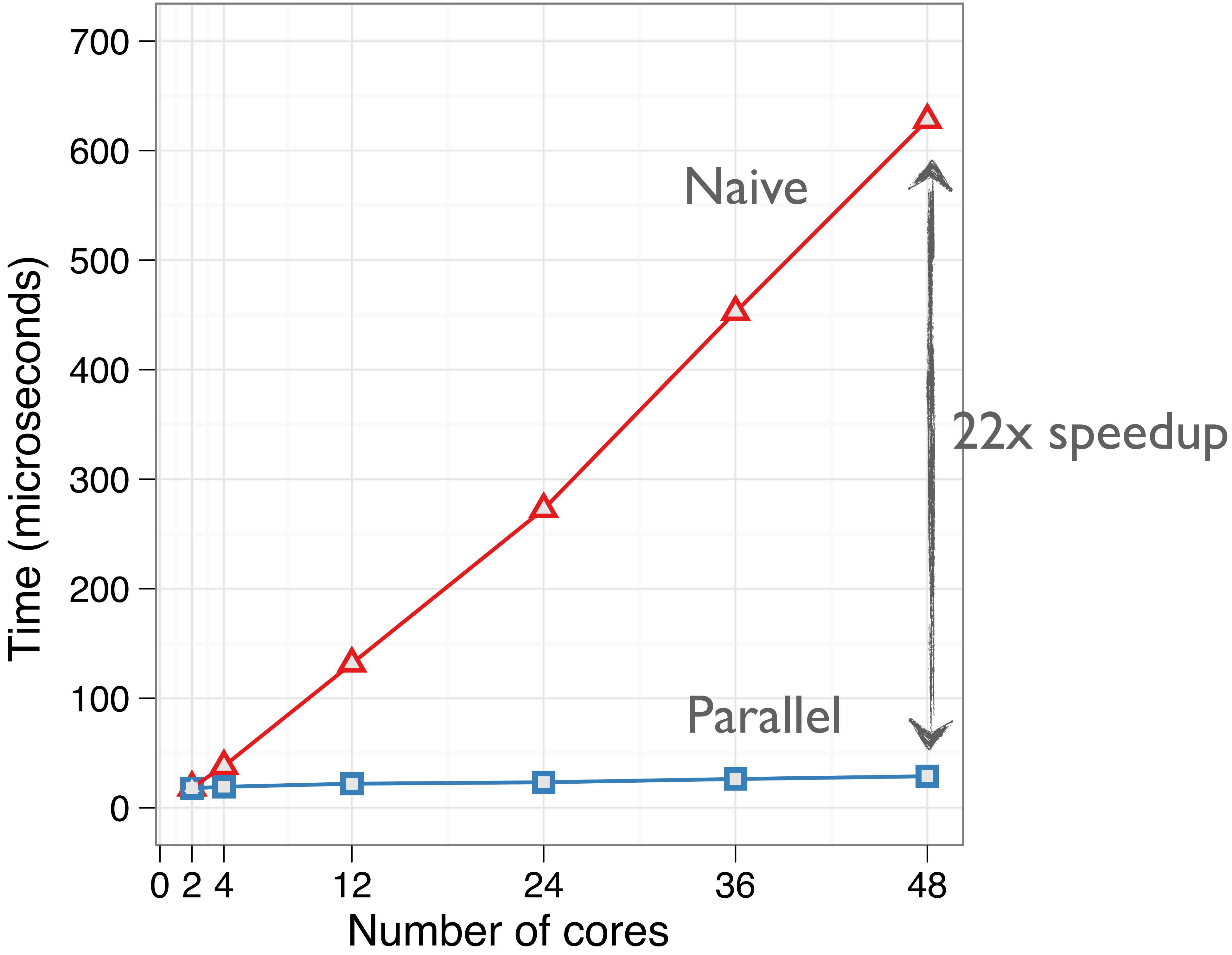


One step, possible contention over the network

Parallel broadcast algorithm

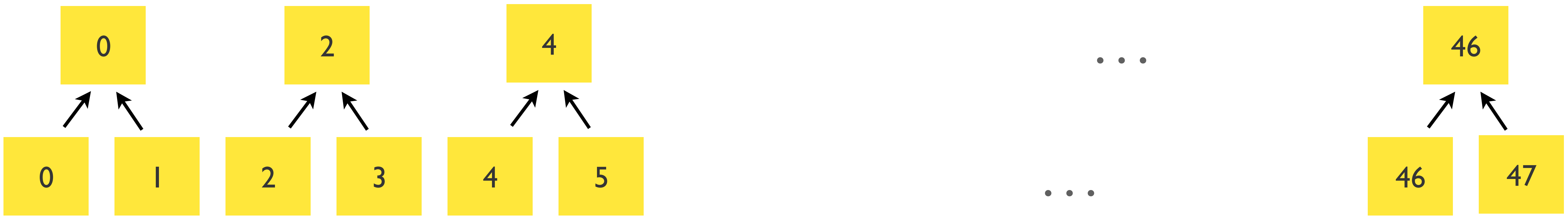
All cores fetch data from core 0’s MPB

Broadcast parallel scaling

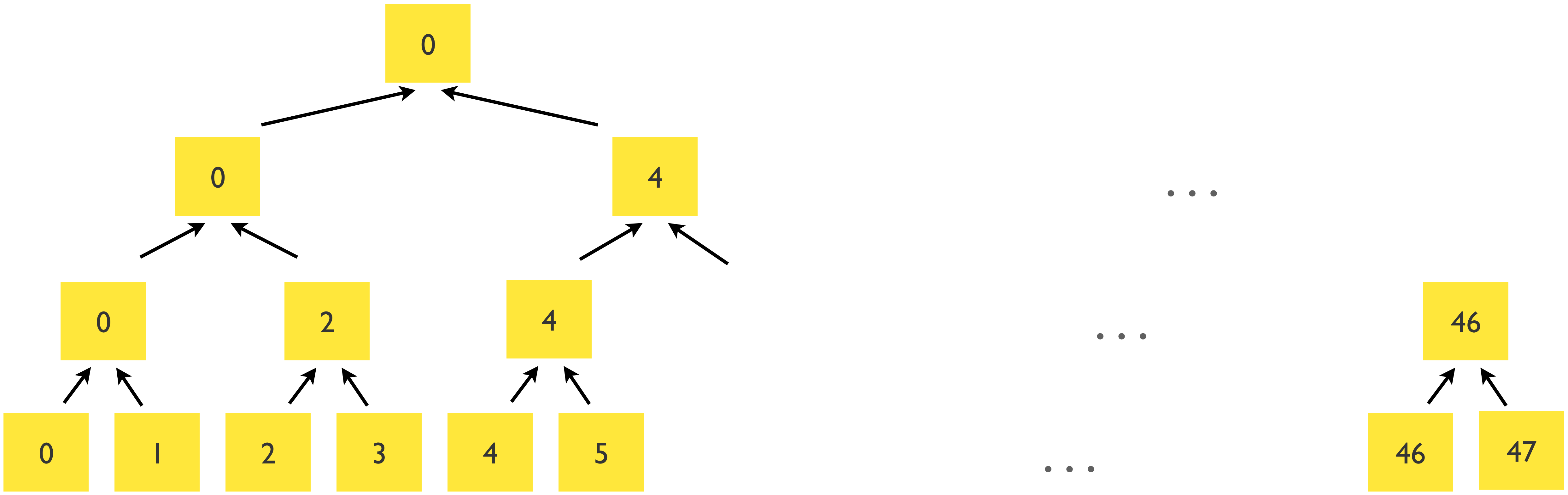


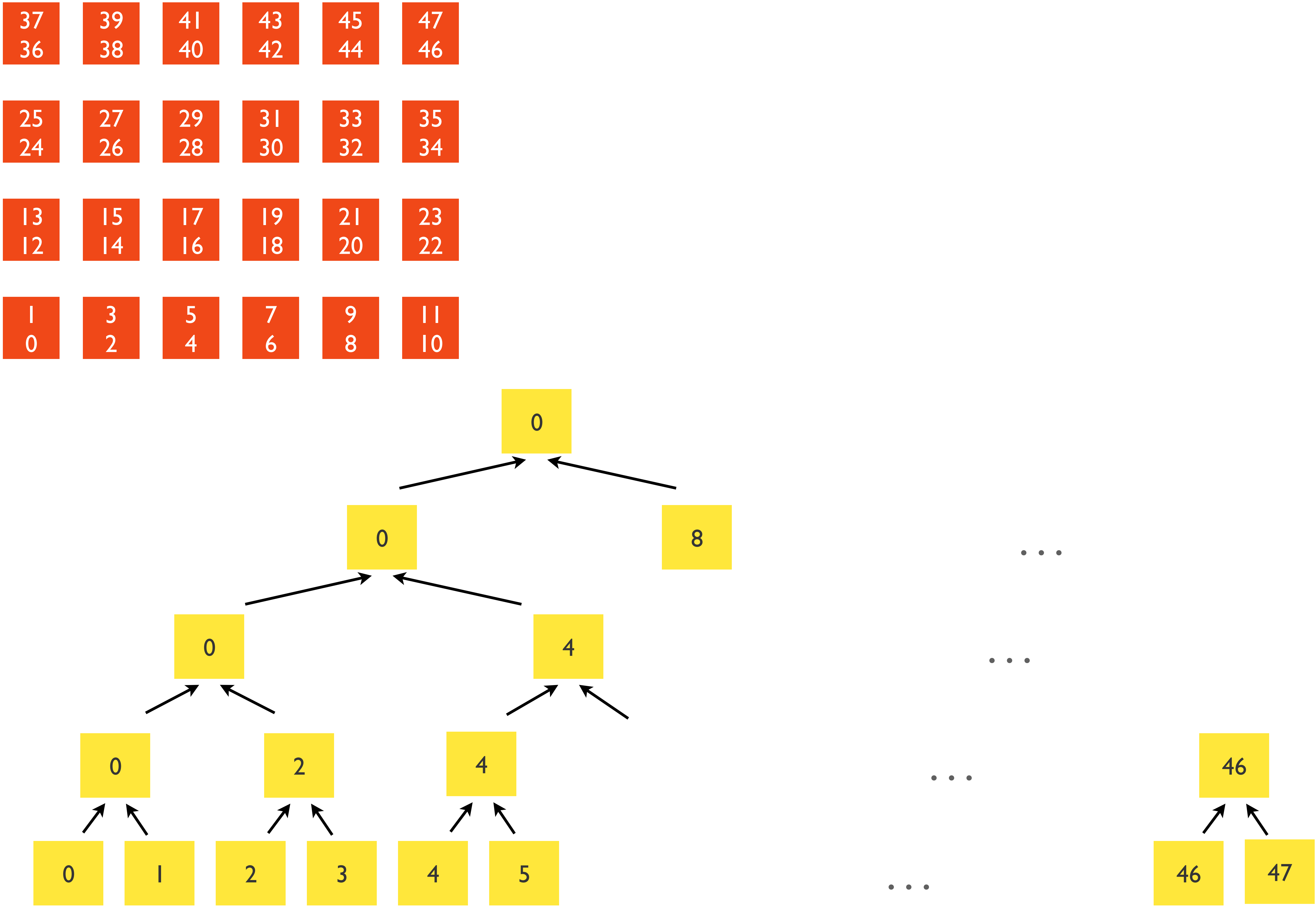
Network contention doesn't seem to hurt broadcast performance

37 36	39 38	41 40	43 42	45 44	47 46
25 24	27 26	29 28	31 30	33 32	35 34
13 12	15 14	17 16	19 18	21 20	23 22
1 0	3 2	5 4	7 6	9 8	11 10



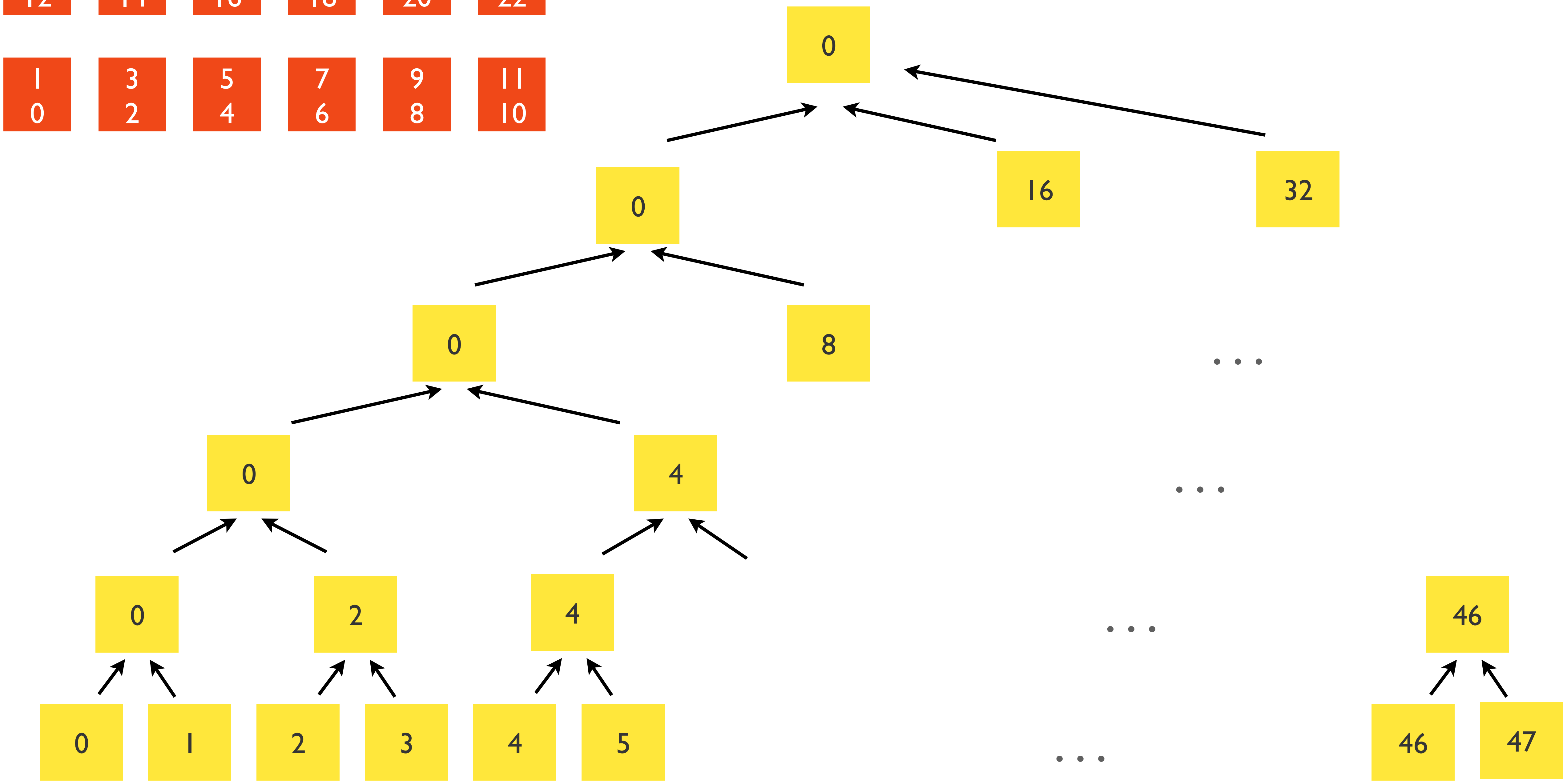
37 36	39 38	41 40	43 42	45 44	47 46
25 24	27 26	29 28	31 30	33 32	35 34
13 12	15 14	17 16	19 18	21 20	23 22
1 0	3 2	5 4	7 6	9 8	11 10



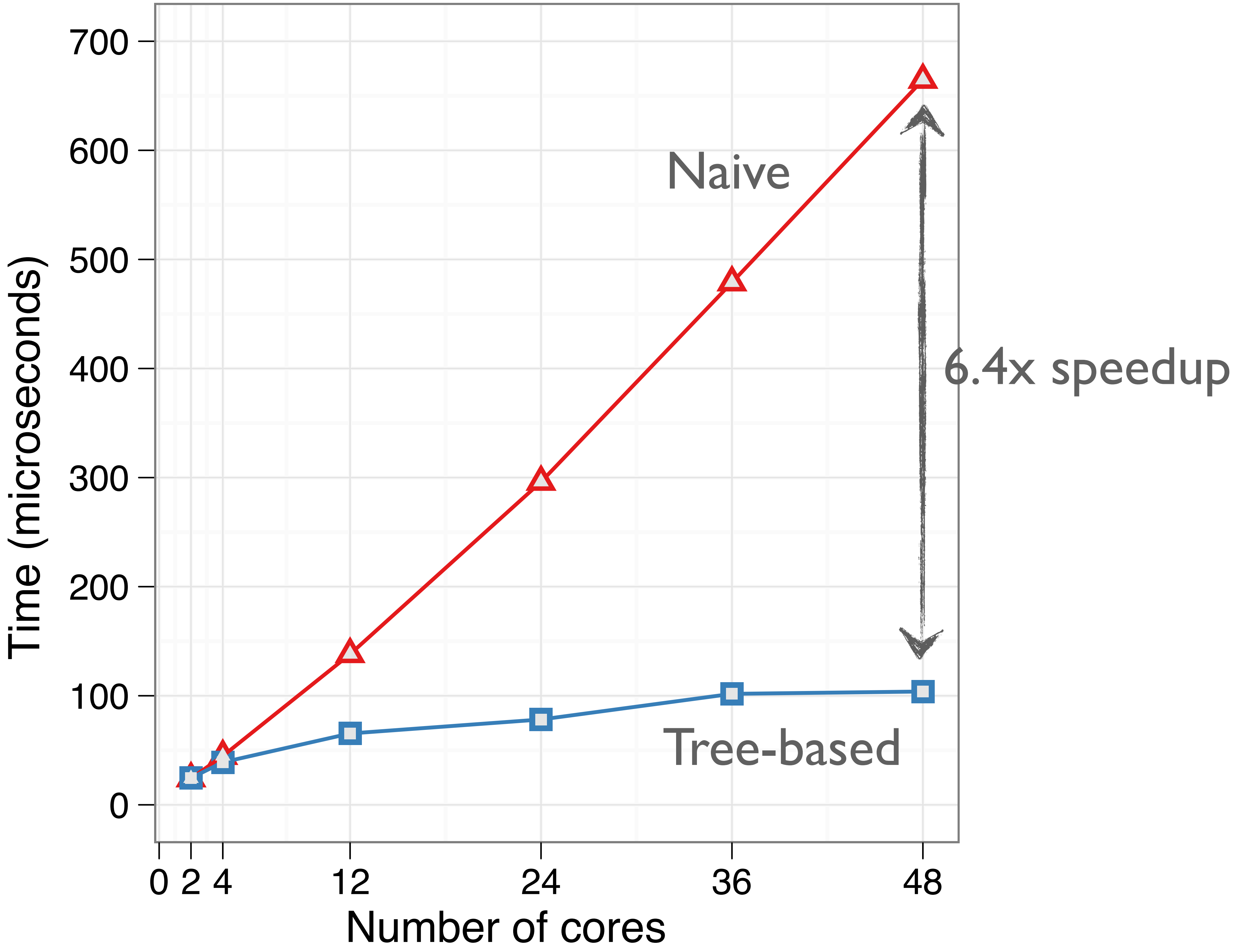

37
36
39
38
41
40
43
42
45
44
47
46
25
24
27
26
29
28
31
30
33
32
35
34
13
12
15
14
17
16
19
18
21
20
23
22
1
0
3
2
5
4
7
6
9
8
11
10
0
0
8
0
4
0
2
4
46
0
1
2
3
4
5
46
47
. . .
. . .
. . .
. . .

37 36	39 38	41 40	43 42	45 44	47 46
25 24	27 26	29 28	31 30	33 32	35 34
13 12	15 14	17 16	19 18	21 20	23 22
1 0	3 2	5 4	7 6	9 8	11 10

log p steps



Reduce parallel scaling



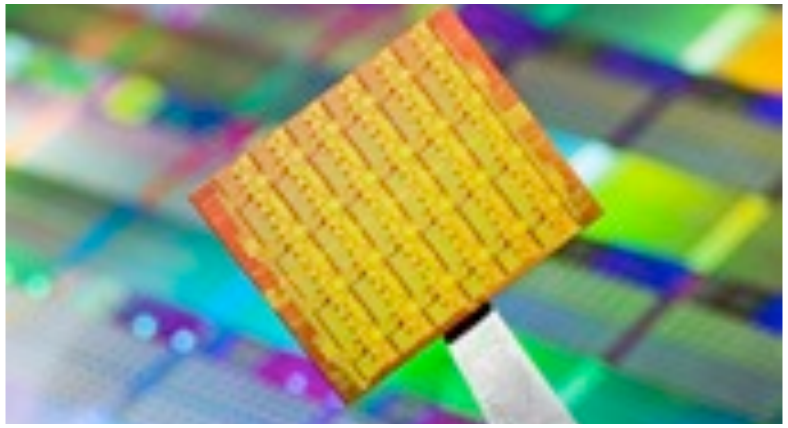
Parallel reduce scales as log p.

Conclusions

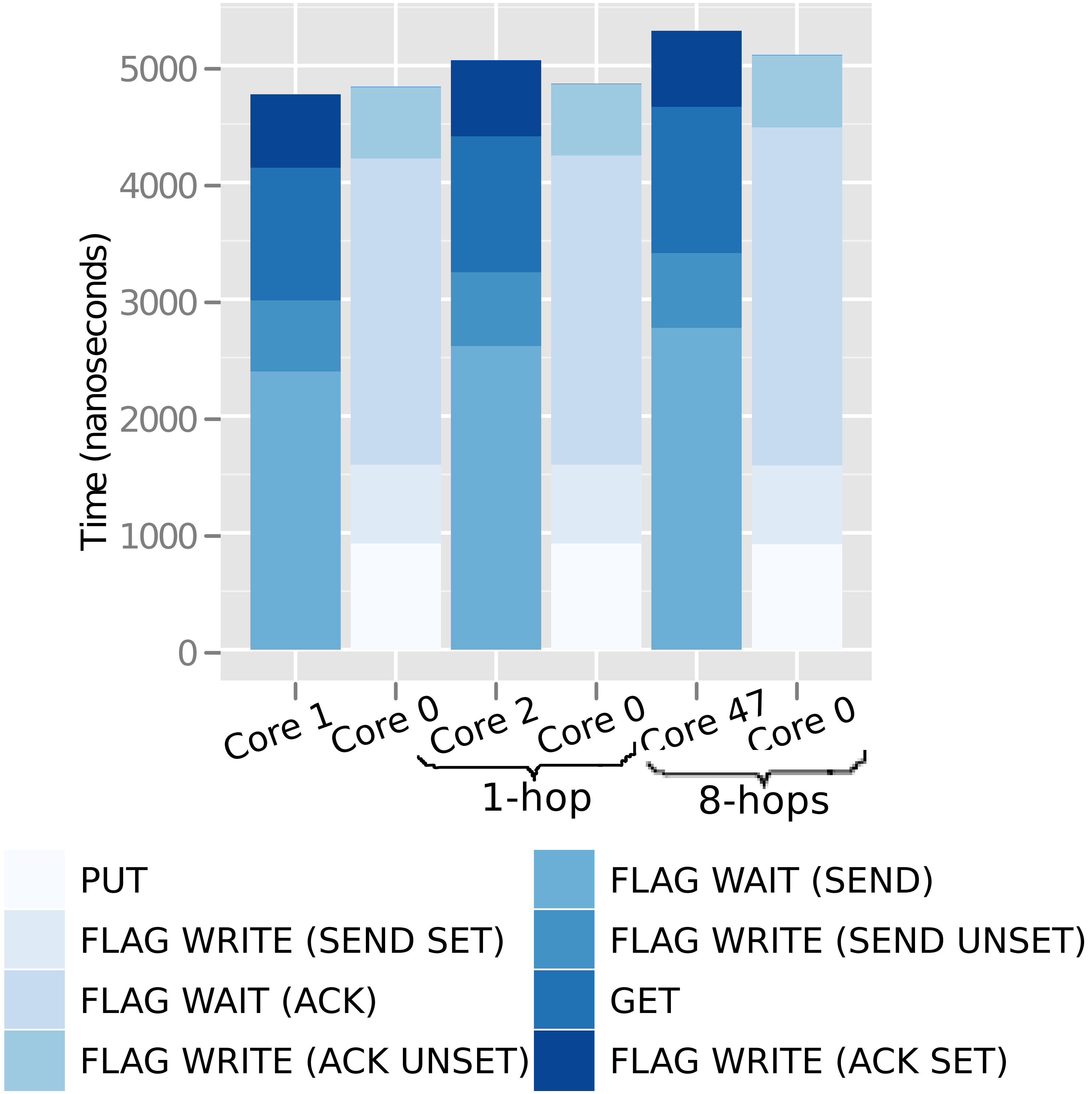
- We demonstrate significant speedup for collective communication by efficiently utilizing the message-passing buffer
- Possible programming models for the SCC?
  - MPI-like, hide MPB complexity hidden from the user
  - Programmer uses collectives (which can be tuned)
- Future work
  - Study other collective patterns
  - Project real application scalability on the SCC
  - Power studies

References

- Intel Communities: Many-core Applications Research Community
  - http://communities.intel.com/community/marc
- Howard et al., A 48-core IA-32 Message Passing Processor with DVFS in 45 nm CMOS, Proc. ISSCC 2010.
- Mattson et al., The 48-core SCC processor: The Programmer's View, Proc. SC 2010.



Questions? aparna@cc.gatech.edu


Time (nanoseconds)
0
1000
2000
3000
4000
5000
Core 1
Core 0
Core 2
Core 0
Core 47
Core 0
1-hop
8-hops
PUT
FLAG WRITE (SEND SET)
FLAG WAIT (ACK)
FLAG WRITE (ACK UNSET)
FLAG WAIT (SEND)
FLAG WRITE (SEND UNSET)
GET
FLAG WRITE (ACK SET)

Backup