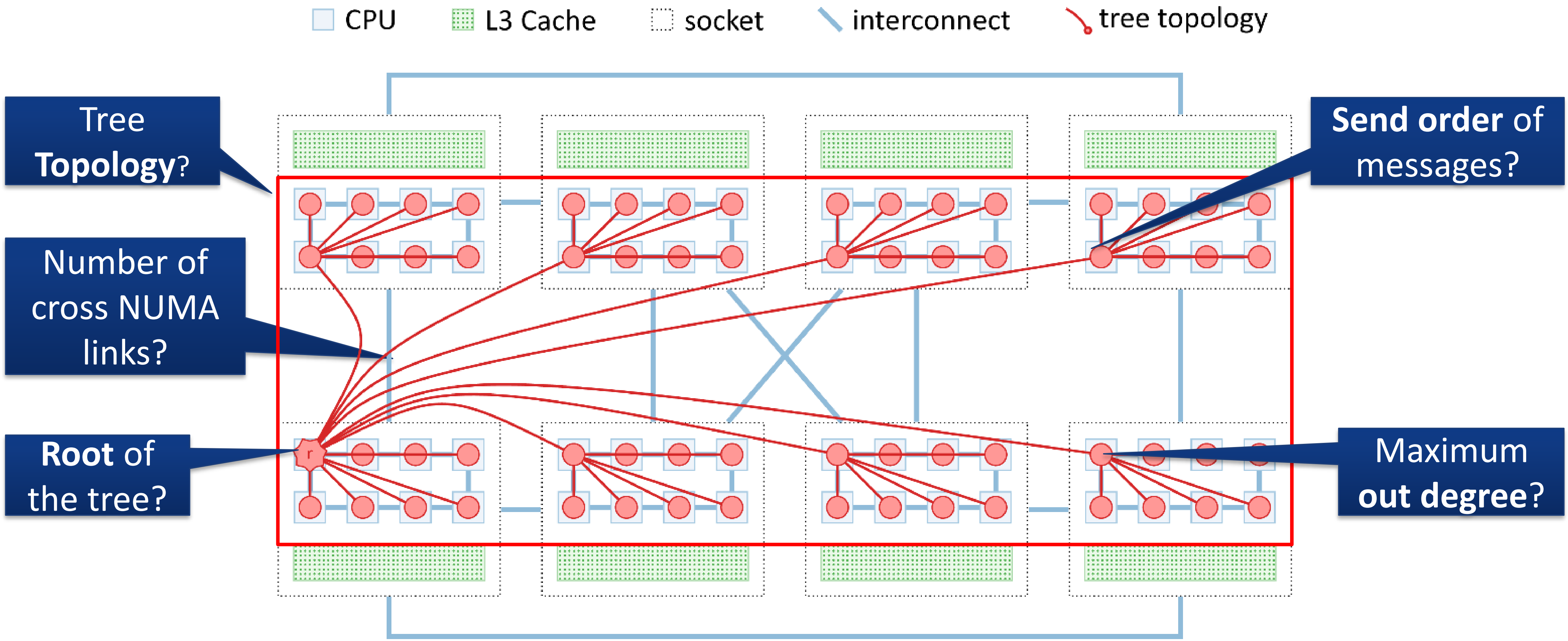
Smelt:
Machine-aware Atomic Broadcast Trees for Multicores

Stefan Kaestle, **Reto Achermann**, Roni Haecki, Moritz Hoffmann, Sabela Ramos, Timothy Roscoe

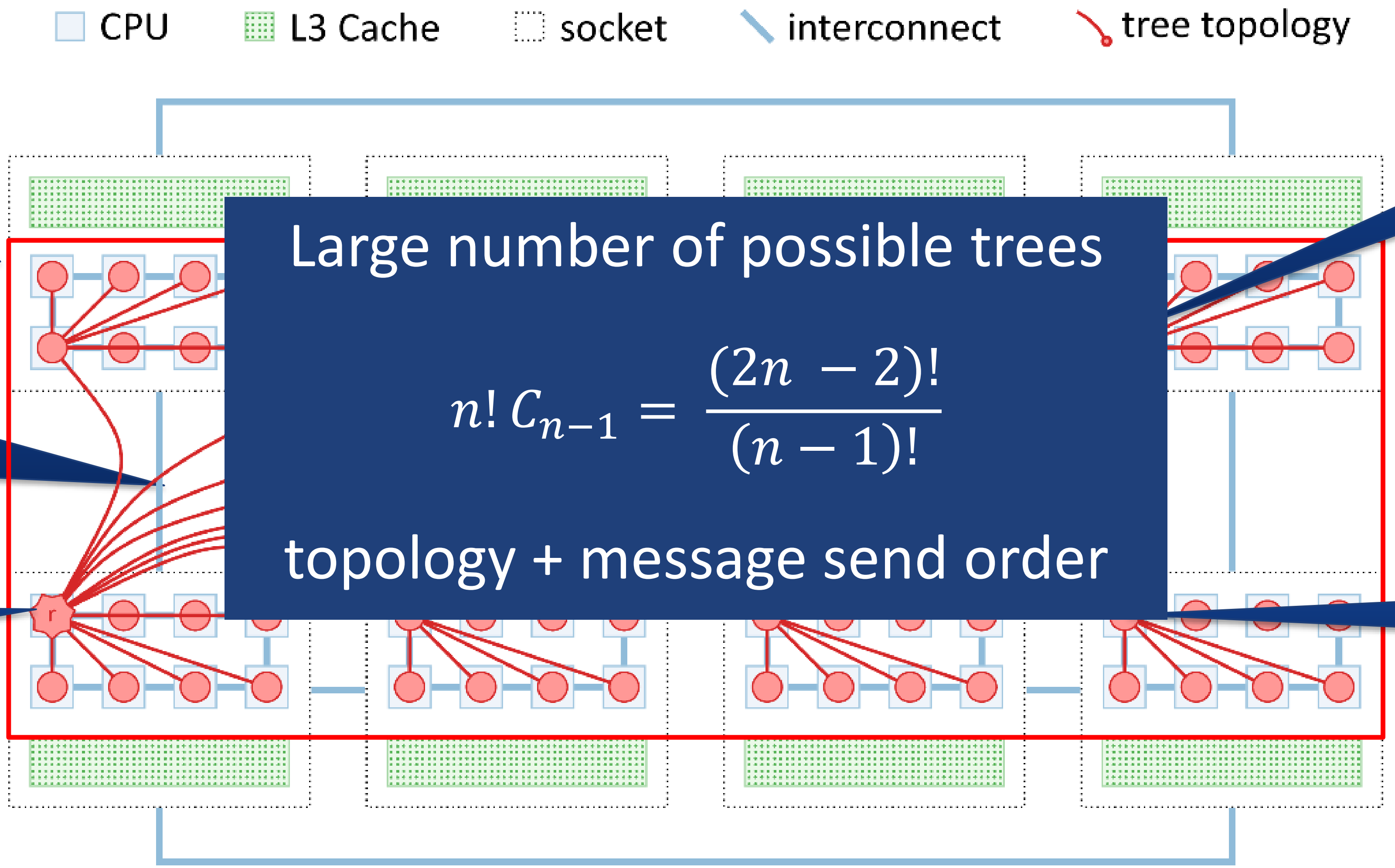
Systems Group, Department of Computer Science, ETH Zurich

Systems@ETH zürich

Large number of trees: topologies and send orders



Large number of trees: topologies and send orders



Tree **Topology**?

Number of cross NUMA links?

Root of the tree?

Send order of messages?

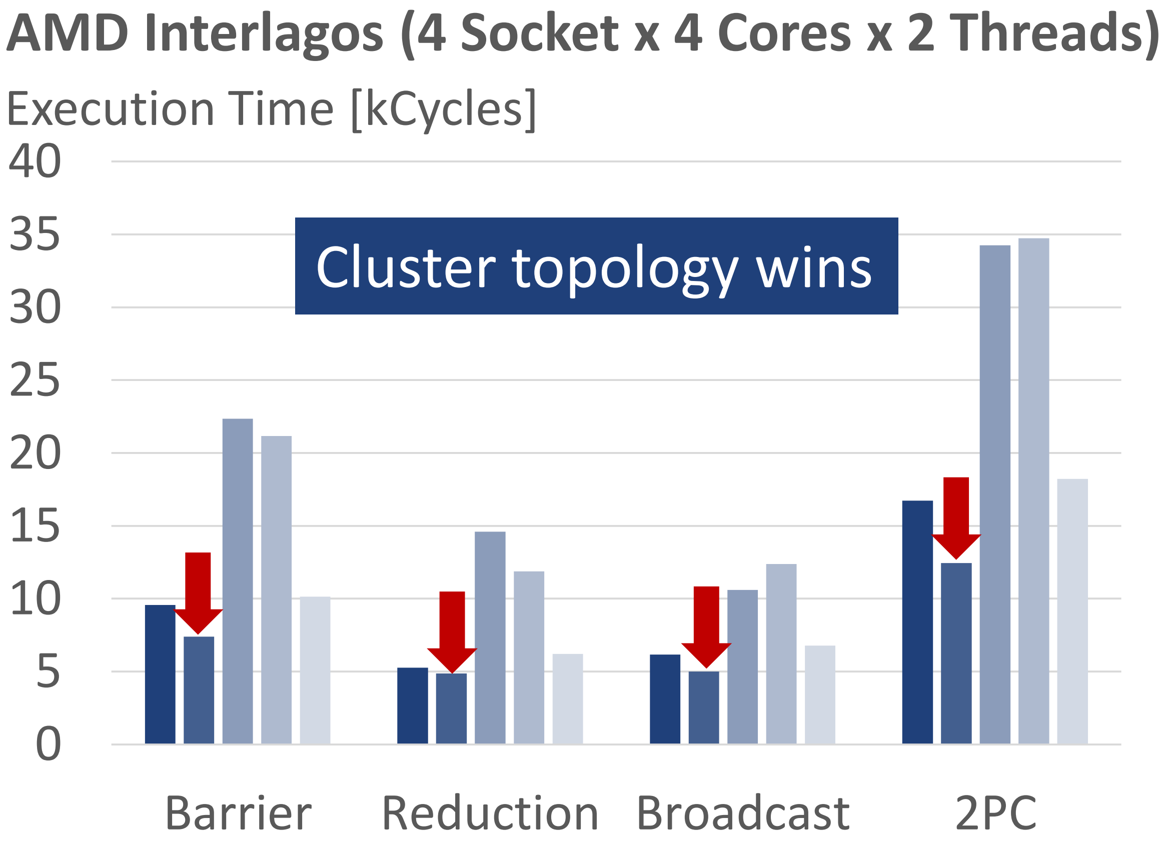
Maximum **out degree**?

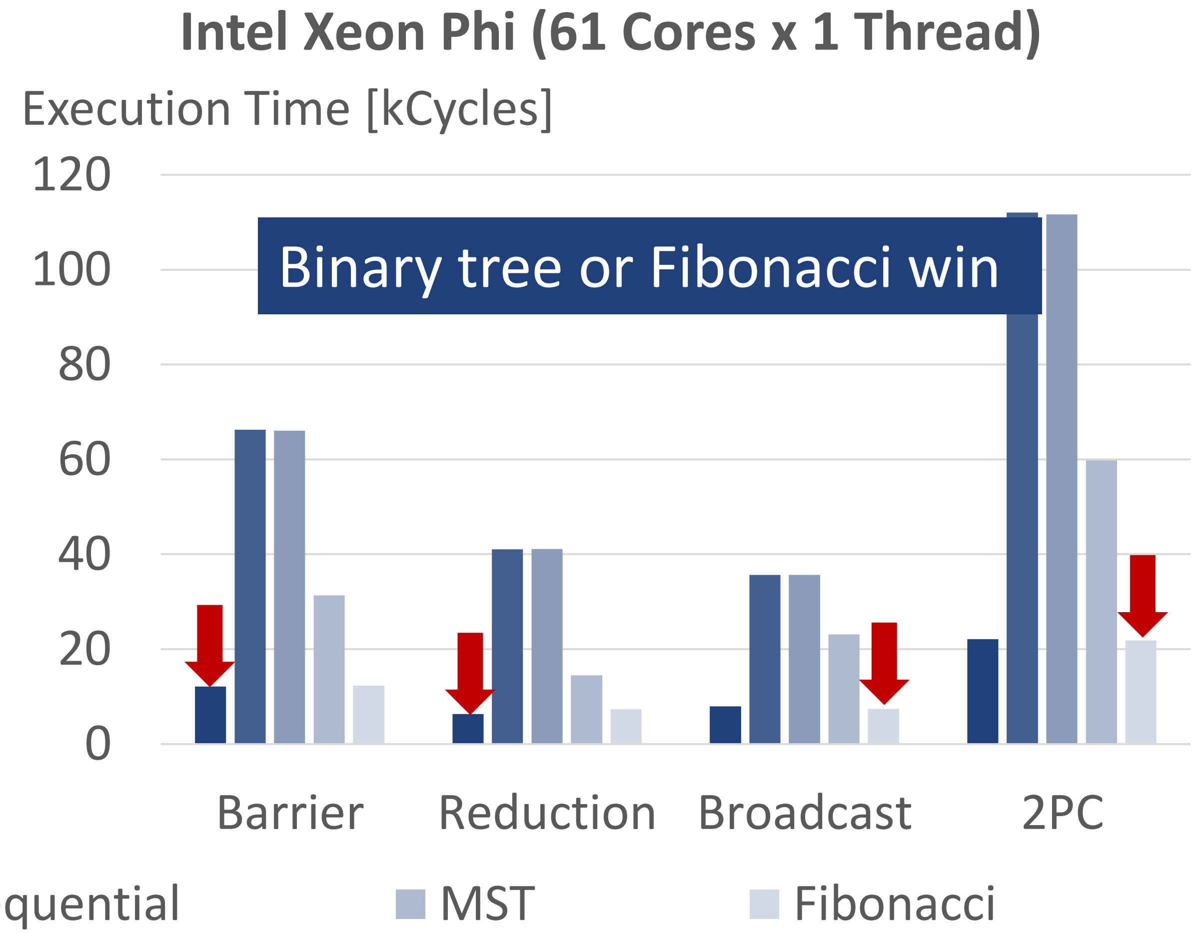
Large number of possible trees

$$n!\,C_{n-1} = \frac{(2n-2)!}{(n-1)!}$$

topology + message send order

There is no globally optimal tree structure

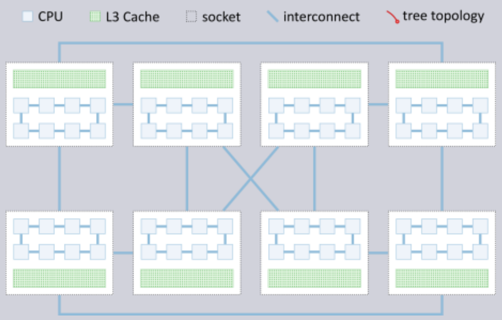




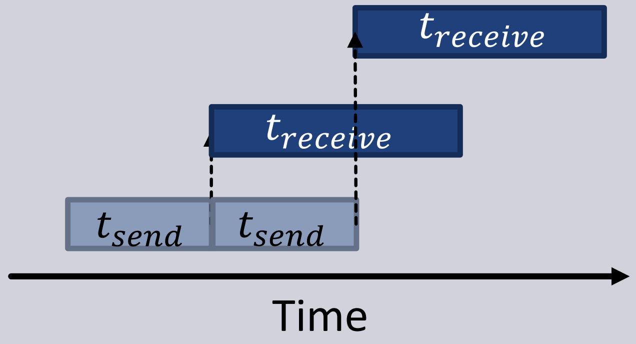
ETH zürich

Smelt: Automatic optimization of broadcast and reduction trees

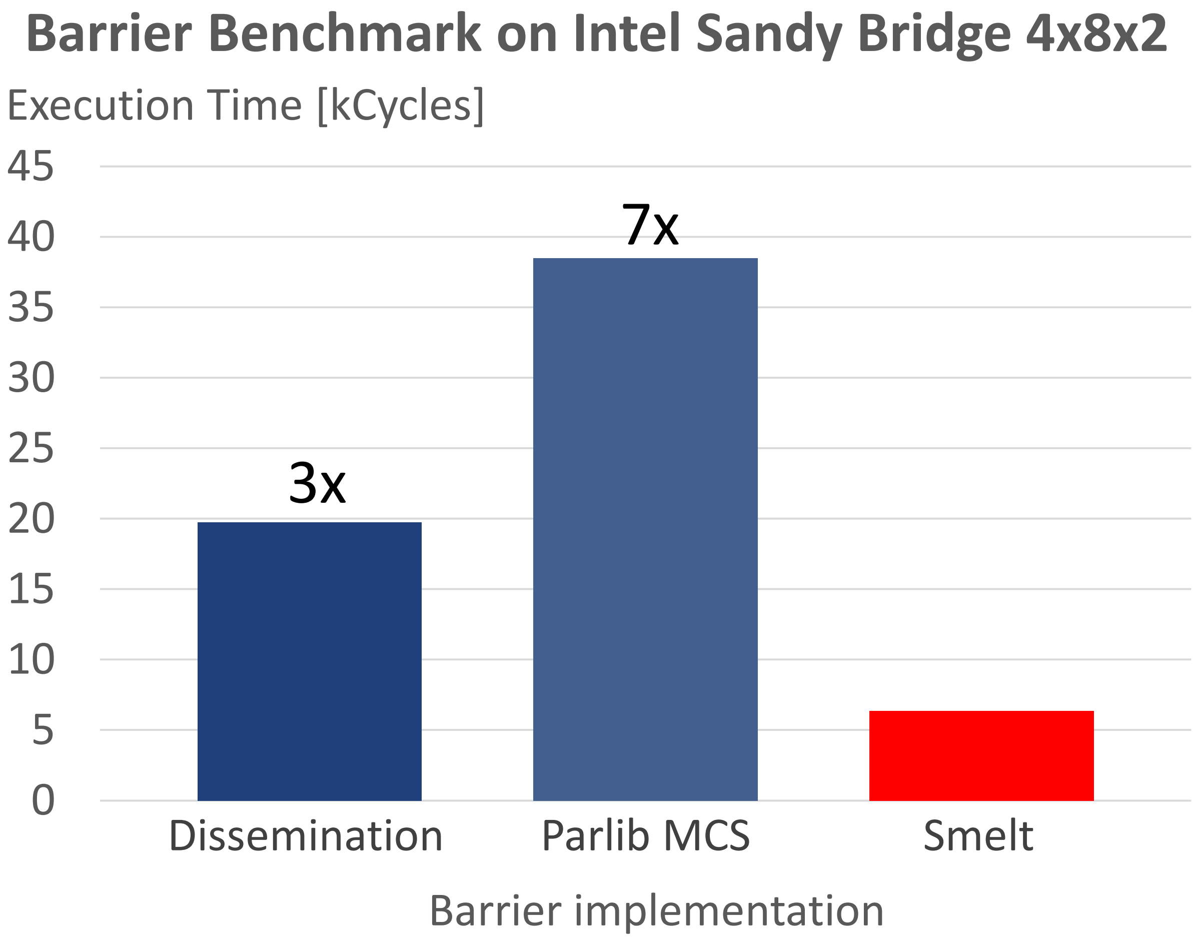
Multicore Model



Tree Generation



Example: Building fast and simple barriers



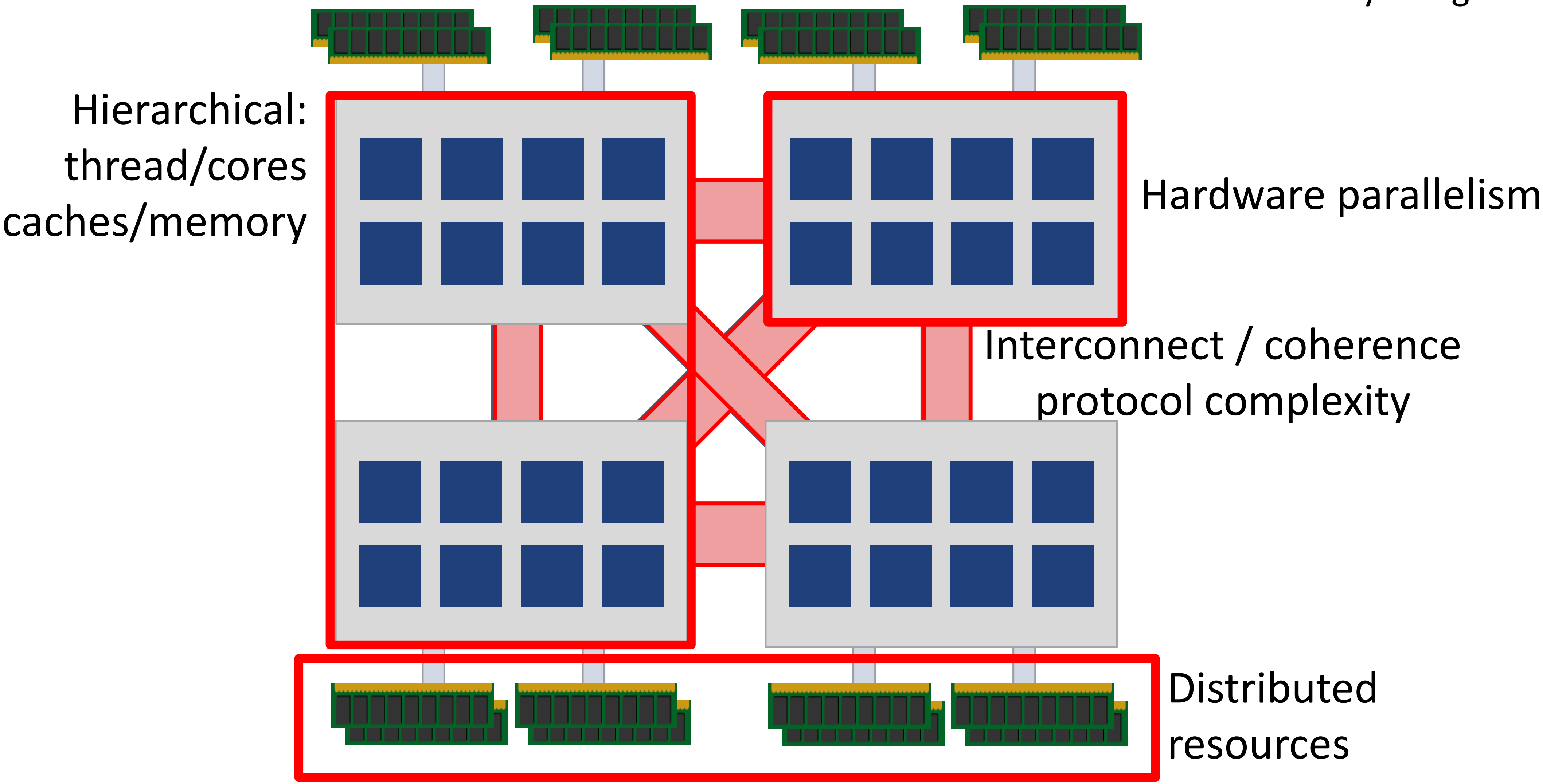
Dramatic improvement through **automatic optimization** of communication patterns

Broadcasts and reductions are central building blocks for parallel programs

Performance	Fault-Tolerance	Execution Control
Atomic broadcasts	**Agreement protocols, atomic broadcasts**	**Reductions, broadcast, barriers**
Replication for **data locality**	Replication for **failure resilience**	Thread **synchronization,** data gathering
e.g. Shoal, Carrefour, SMMP OS, FOS	e.g. 1Paxos	e.g. OpenMP

Multicore hardware is complex

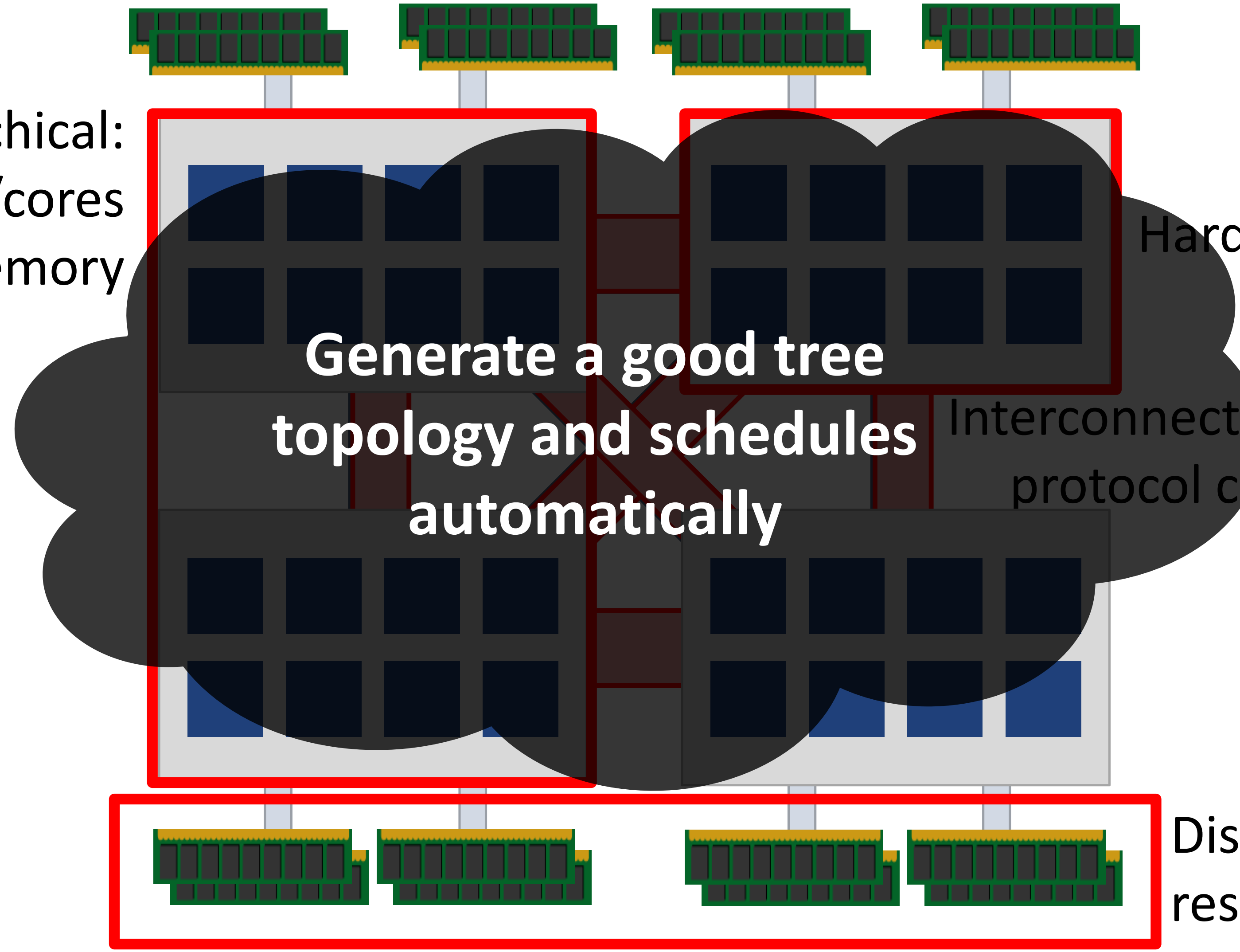
Intel Sandy Bridge 4x8x2



Multicore hardware is complex

Intel Sandy Bridge 4x8x2

Hierarchical:
thread/cores
caches/memory

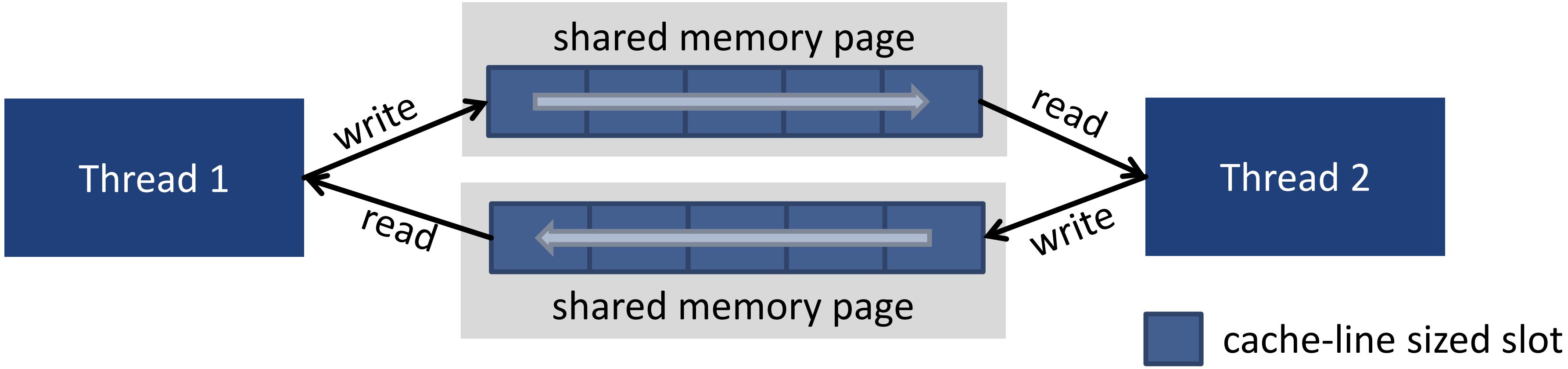


Hardware parallelism

Interconnect / coherence
protocol complexity

Distributed
resources

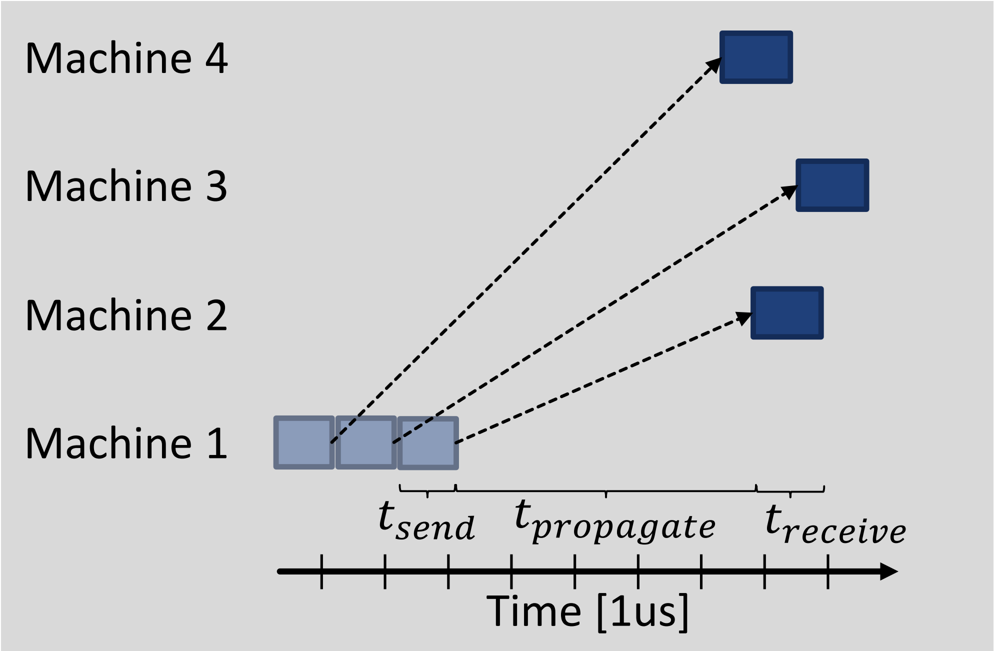
Smelt is based on peer-to-peer message passing



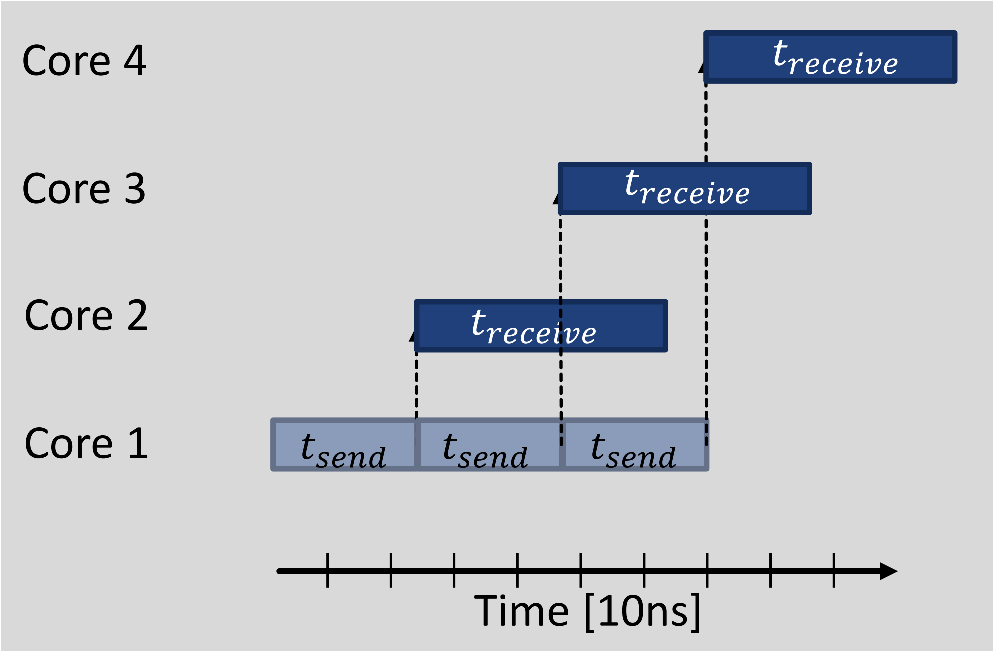
- Works well for our approach.
- Clear concept: Enables **reasoning** about send and receive costs

Message-passing on multicores is different

Classical Network

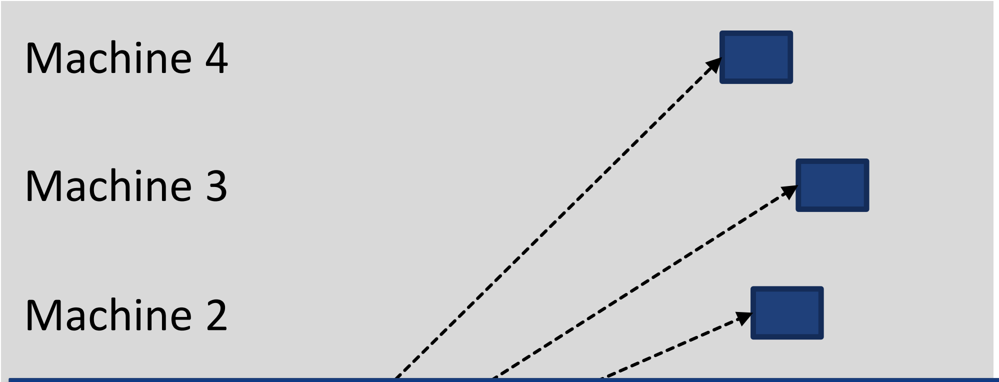


Multicore interconnect

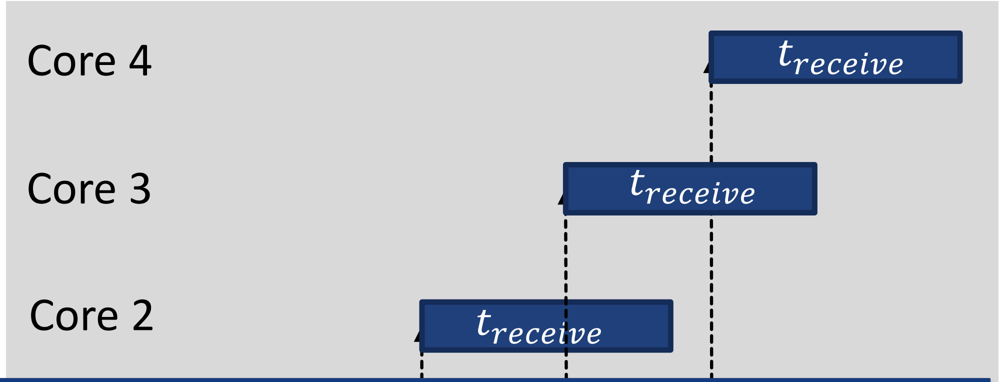


Message-passing on multicores is different

Classical Network



Multicore interconnect



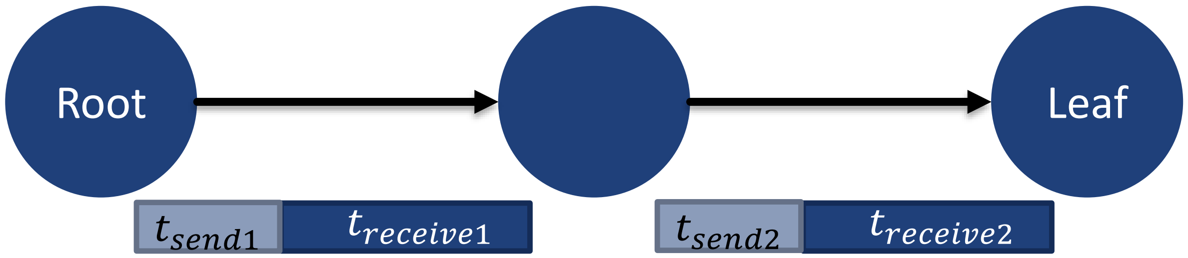
On multicores send and receive times dominate propagation time

Goal: Minimize total time of the broadcast

Minimizing the total time of a broadcast

- $t_{broadcast} = t_{last} - t_{start}$

- Minimize the longest path from the root to the leaves.



$$t_{path} = \sum(t_{send} + t_{receive})$$

We need to know the send and receive cost between any pair of cores

Information obtained from hardware discovery

AMD Interlagos 4x4x2

```
$ lscpu
CPU(s):                 64
Thread(s) per core:     2
Core(s) per socket:     8
Socket(s):              4
NUMA node(s):           8
L1d cache:              16K
L1i cache:              64K
L2 cache:               2048K
L3 cache:               6144K
NUMA node0 CPU(s):      0,4,8,12,16,20,24,28
NUMA node1 CPU(s):      32,36,40,44,48,52,56,60
NUMA node2 CPU(s):      2,6,10,14,18,22,26,30
NUMA node3 CPU(s):      34,38,42,46,50,54,58,62
NUMA node4 CPU(s):      3,7,11,15,19,23,27,31
NUMA node5 CPU(s):      35,39,43,47,51,55,59,63
NUMA node6 CPU(s):      1,5,9,13,17,21,25,29
NUMA node7 CPU(s):      33,37,41,45,49,53,57,61
```

```
$ numactl -hardware
node distances:
node   0   1   2   3   4   5   6   7
  0:  10  16  16  22  16  22  16  22
  1:  16  10  22  16  16  22  22  16
  2:  16  22  10  16  16  16  16  16
  3:  22  16  16  10  16  16  22  22
  4:  16  16  16  16  10  16  16  22
  5:  22  22  16  16  16  10  22  16
  6:  16  22  16  22  16  22  10  16
  7:  22  16  16  22  22  16  16  10
```

Information obtained from hardware discovery

AMD Interlagos 4x4x2

```
$ lscpu
C
Thread(s) per core:         2
C
S
N
L1d cache:                  16K
L
L2 cache:                   2048K
L3 cache:                   6144K
NUMA node0 CPU(s):          0,4,8,12,16,20,24,28
NUMA node1 CPU(s):          32,36,40,44,48,52,56,60
NUMA node2 CPU(s):          2,6,10,14,18,22,26,30
NUMA node3 CPU(s):          34,38,42,46,50,54,58,62
NUMA node4 CPU(s):          3,7,11,15,19,23,27,31
NUMA node5 CPU(s):          35,39,43,47,51,55,59,63
NUMA node6 CPU(s):          1,5,9,13,17,21,25,29
NUMA node7 CPU(s):          33,37,41,45,49,53,57,61
```

NUMA distance: abstract value

Doesn't distinguish between send() and recv()

Symmetric: A→B == B→A

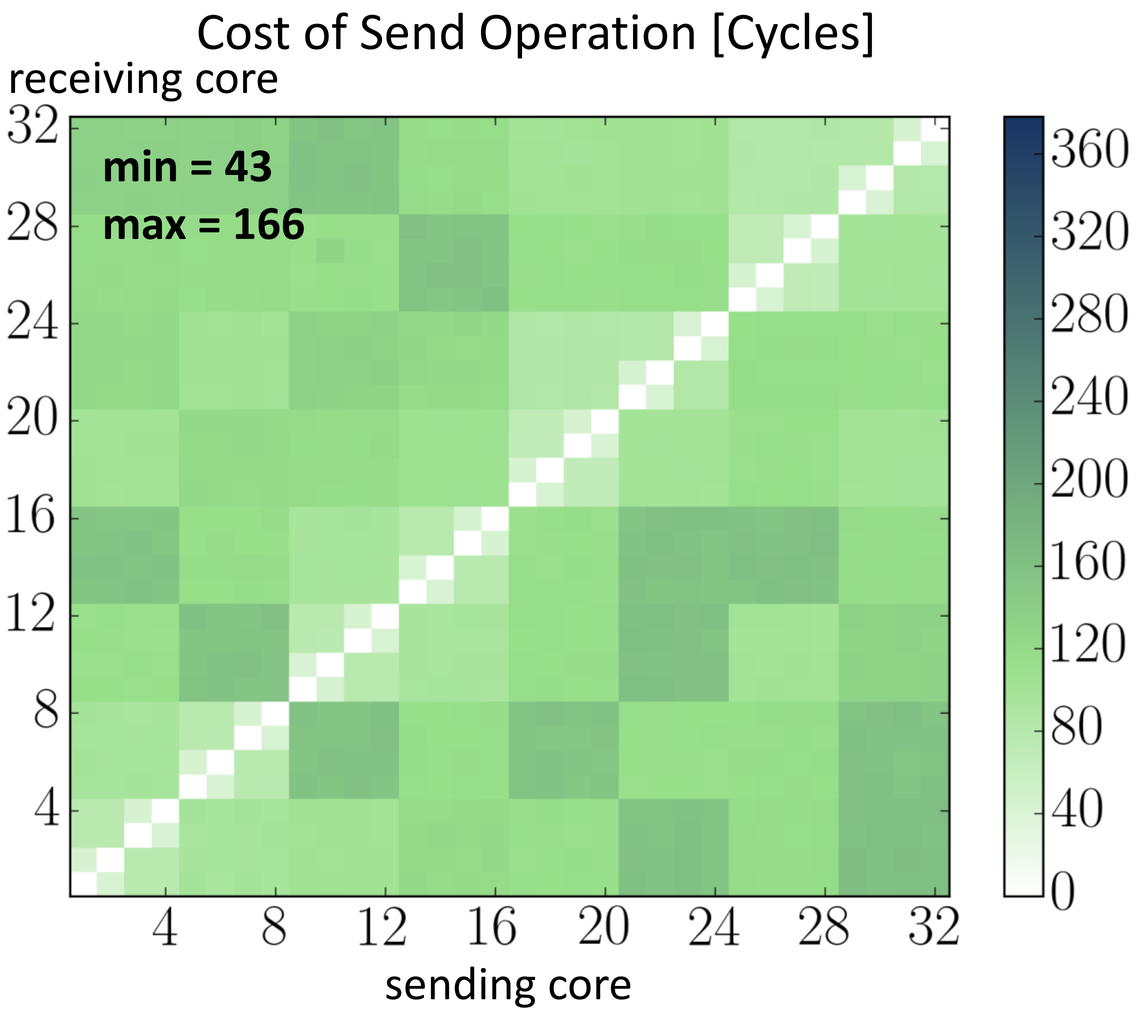
```
$ numactl -hardware
node distances:
```

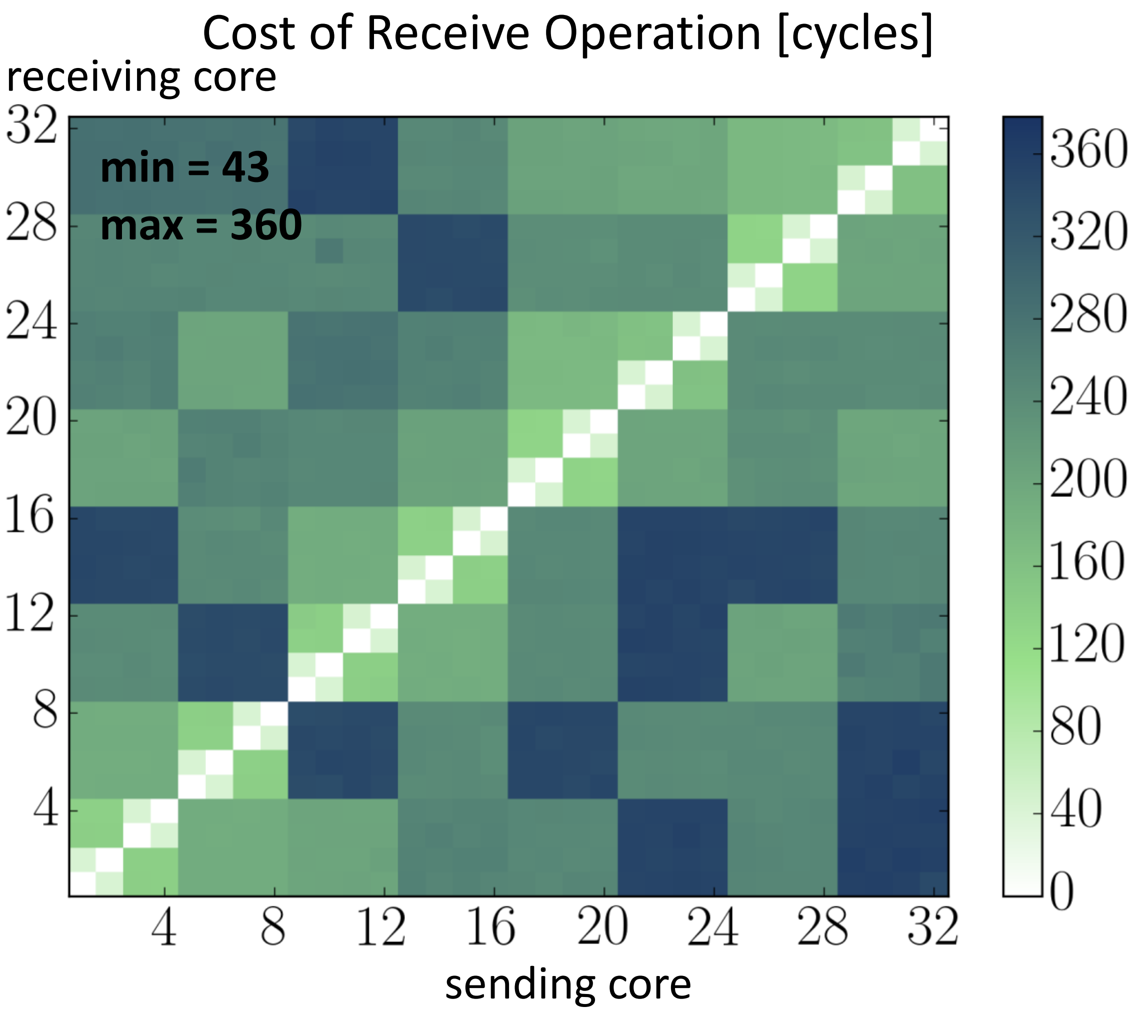
node	0	1	2	3	4	5	6	7
0:	10	16	16	22	16	22	16	22
1:	16	10	22	16	16	22	22	16
2:	16	22	10	16	16	16	16	16
3:	22	16	16	10	16	16	22	22
4:	16	16	16	16	10	16	16	22
5:	22	22	16	16	16	10	22	16
6:	16	22	16	22	16	22	10	16
7:	22	16	16	22	22	16	16	10

2 4 2 4

Complement with microbenchmarks: pairwise send and receive

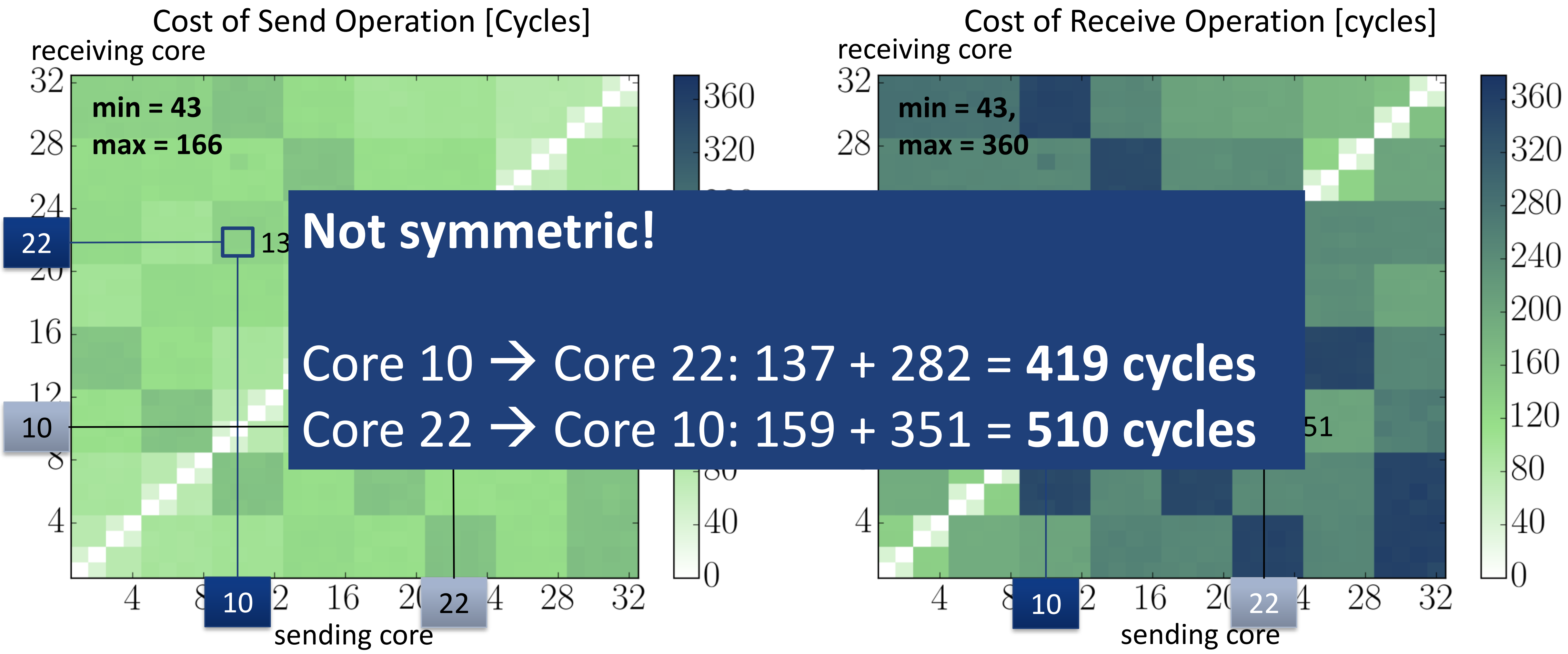
AMD Interlagos 4x4x2





Complement with microbenchmarks: pairwise send and receive

AMD Interlagos 4x4x2



Not symmetric!

Core 10 → Core 22: 137 + 282 = **419 cycles**

Core 22 → Core 10: 159 + 351 = **510 cycles**

Smelt

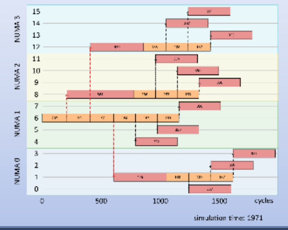
Using Smelt for group communication

```
#include <smelt/smelt.h>

void main() {

   smelt_init();


   smelt_topology_create();


   smelt_broadcast(msg);

}
```

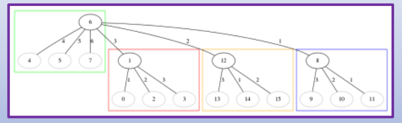
Smelt runtime

```
smelt_topology_create() {
```

Smelt Algorithm



Tree topology



```
}
```

Program

Hardware Discovery

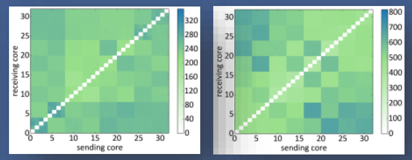
Cores

NUMA nodes

```
lstopo; /proc/cpu
```

Measurements

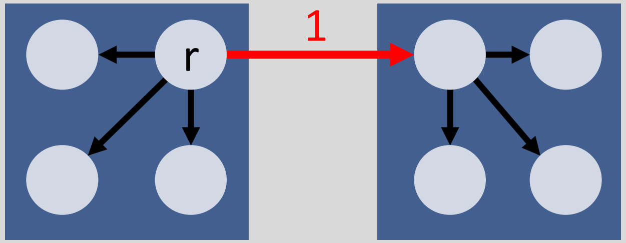
Micro-benchmarks



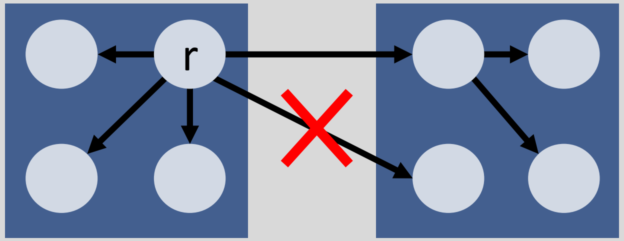
Multicore Model

Smelt's tree generator heuristics

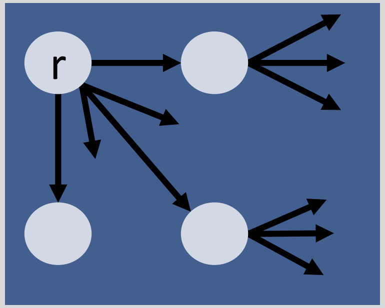
Remote Cores First



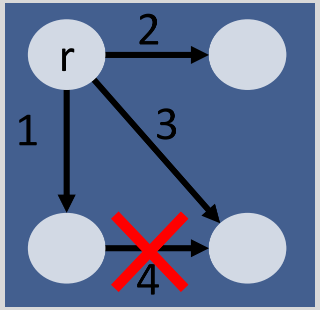
Avoid Expensive Communication



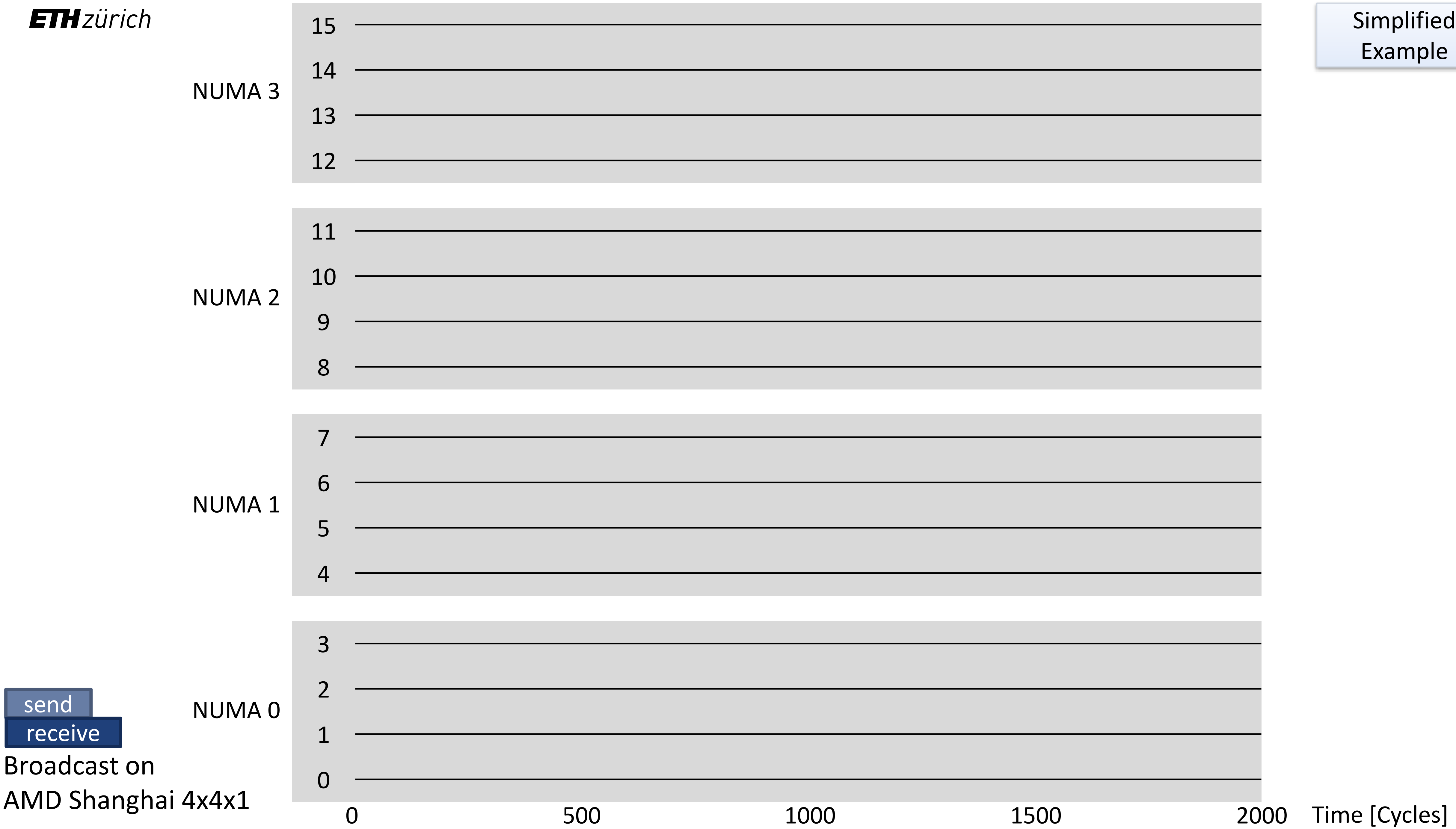
Maximize Parallelism



No redundancy



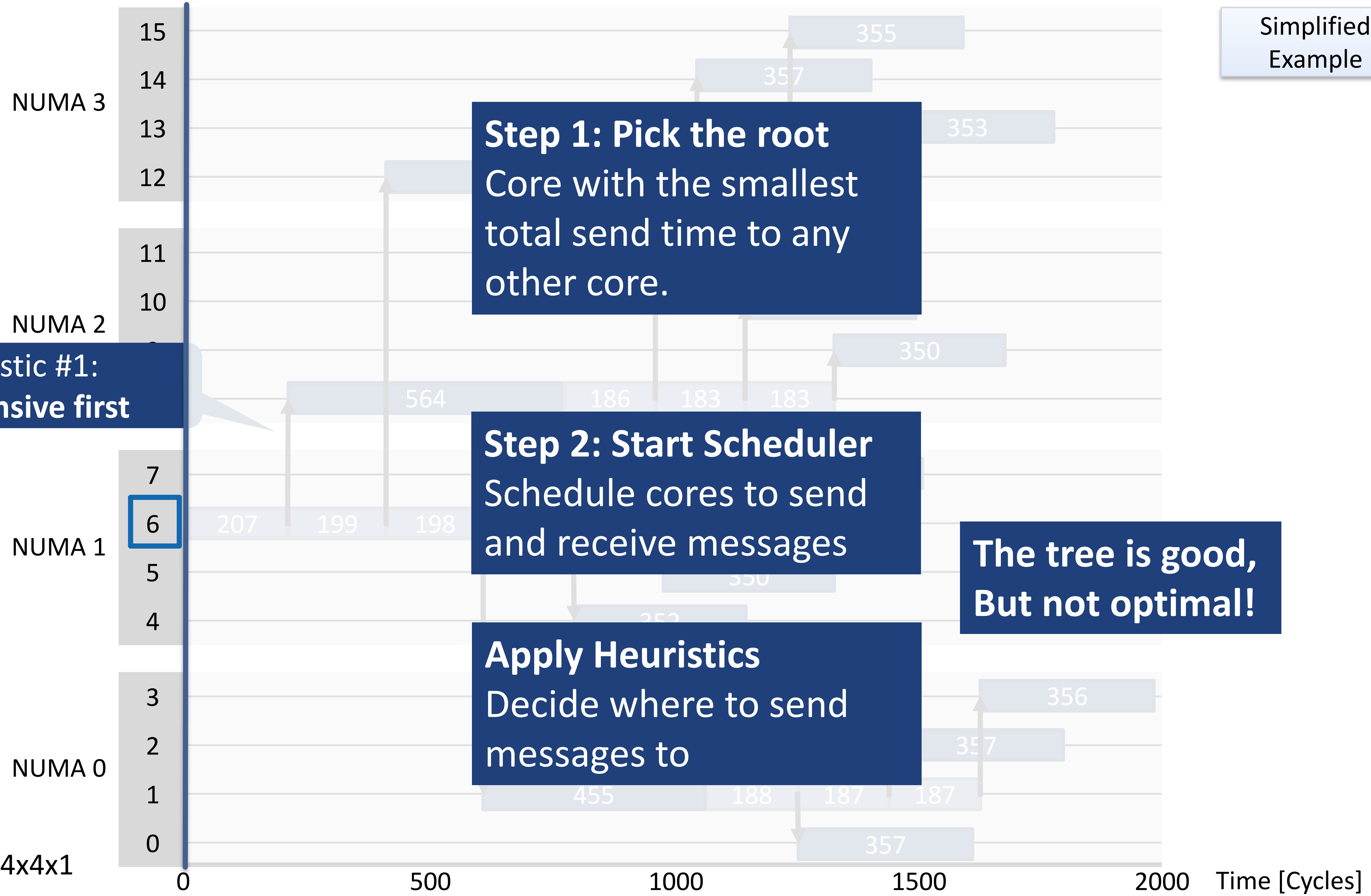
ETH zürich



Broadcast on
AMD Shanghai 4x4x1

ETH zürich

Simplified Example



Step 1: Pick the root
Core with the smallest total send time to any other core.

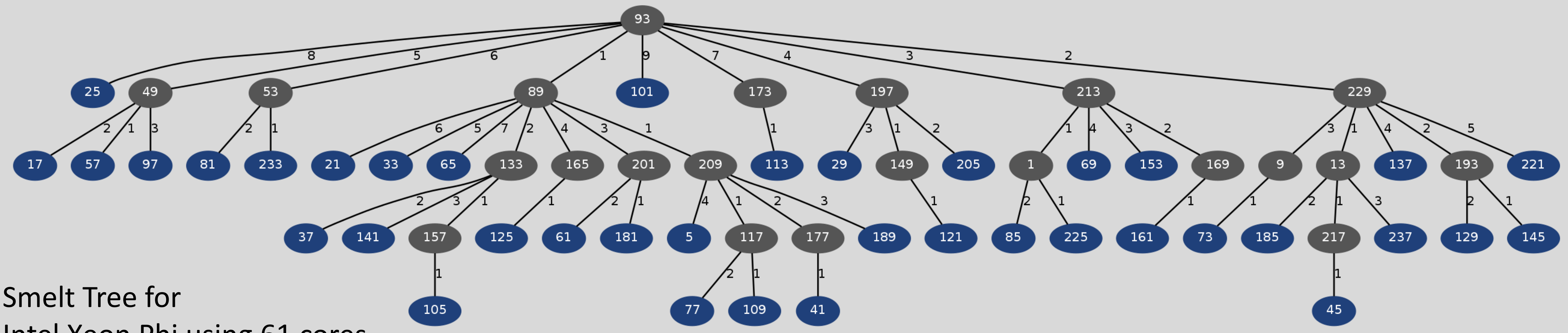
Step 2: Start Scheduler
Schedule cores to send and receive messages

Apply Heuristics
Decide where to send messages to

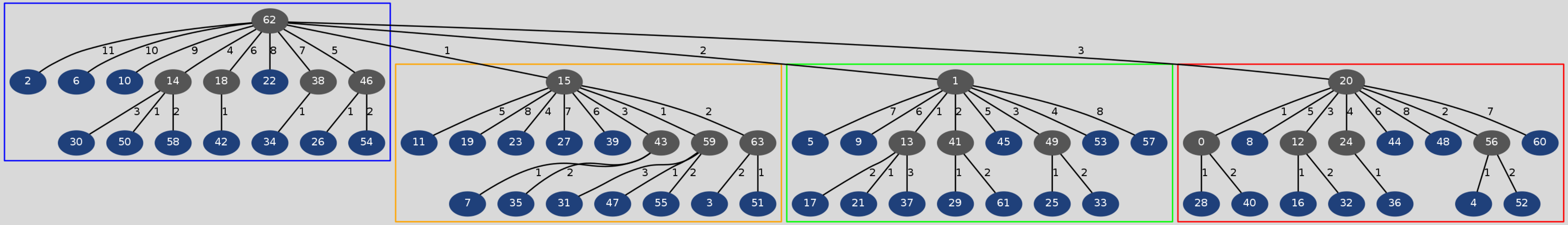
Heuristic #1:
Expensive first

The tree is good, But not optimal!

Broadcast on AMD Shanghai 4x4x1



Smelt Tree for
Intel Xeon Phi using 61 cores



Smelt Tree for
Intel Sandy Bridge 4 Sockets x 8 Cores x 2 Threads

Evaluation Testbed

Intel

Architecture	Sockets	Cores / Socket	Threads / Core
Ivy Bridge	2	10	2
Nehalem	4	8	2
Knights Corner	1	61	4
Sandy Bridge	4	8	2
Sandy Bridge	2	10	2
Bloomfield	2	4	2

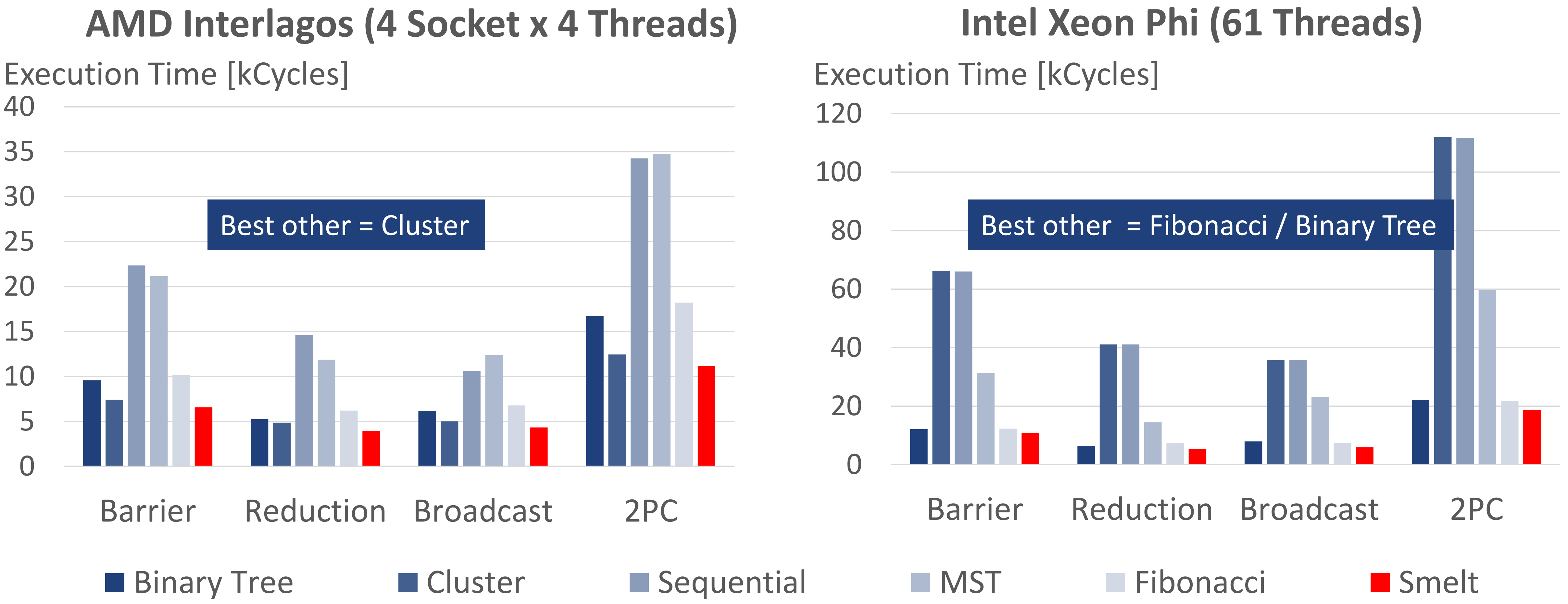
AMD

Architecture	Sockets	Cores / Socket	Threads / Core
Magny Cours	4	12	1
Barcelona	8	4	1
Shanghai	4	4	1
Interlagos	4	4	2
Istanbul	4	6	1

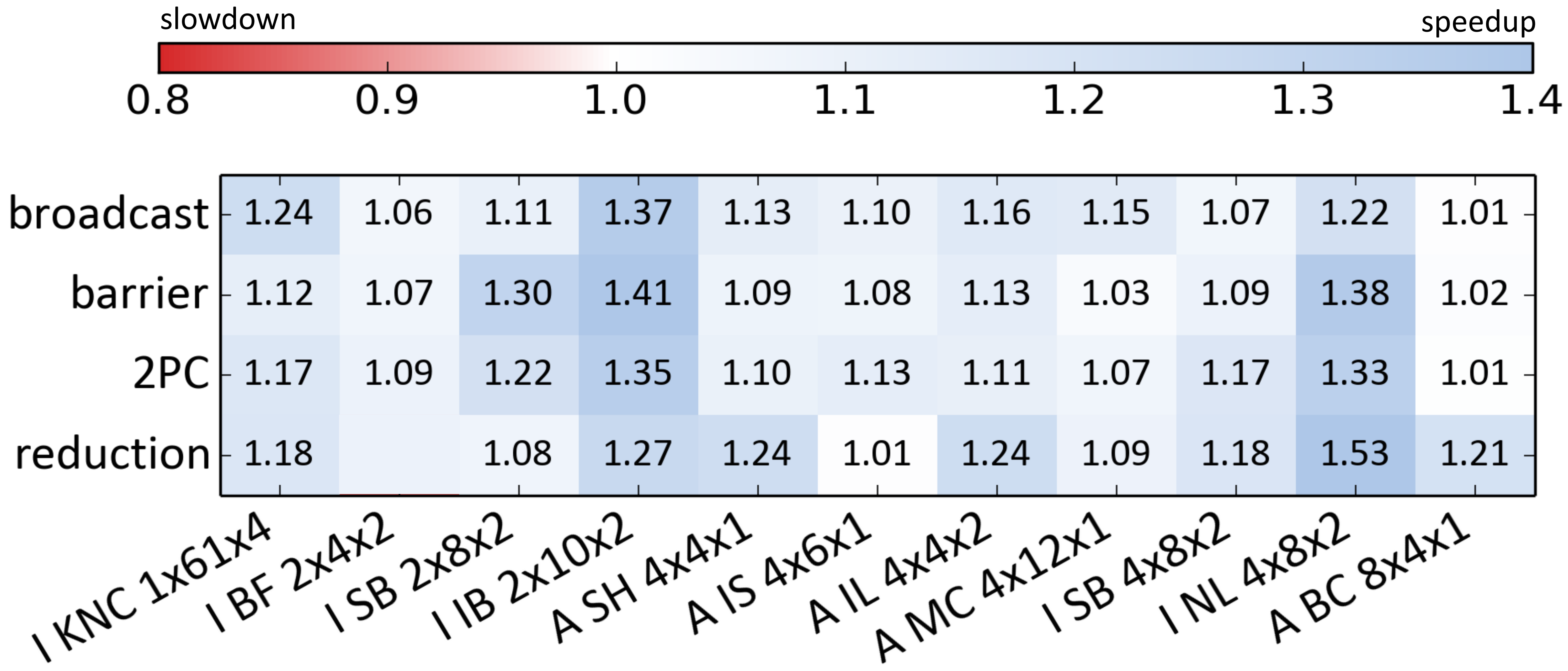
Full set of results online.

http://machinedb.systems.ethz.ch

Smelt produces good trees across architectures

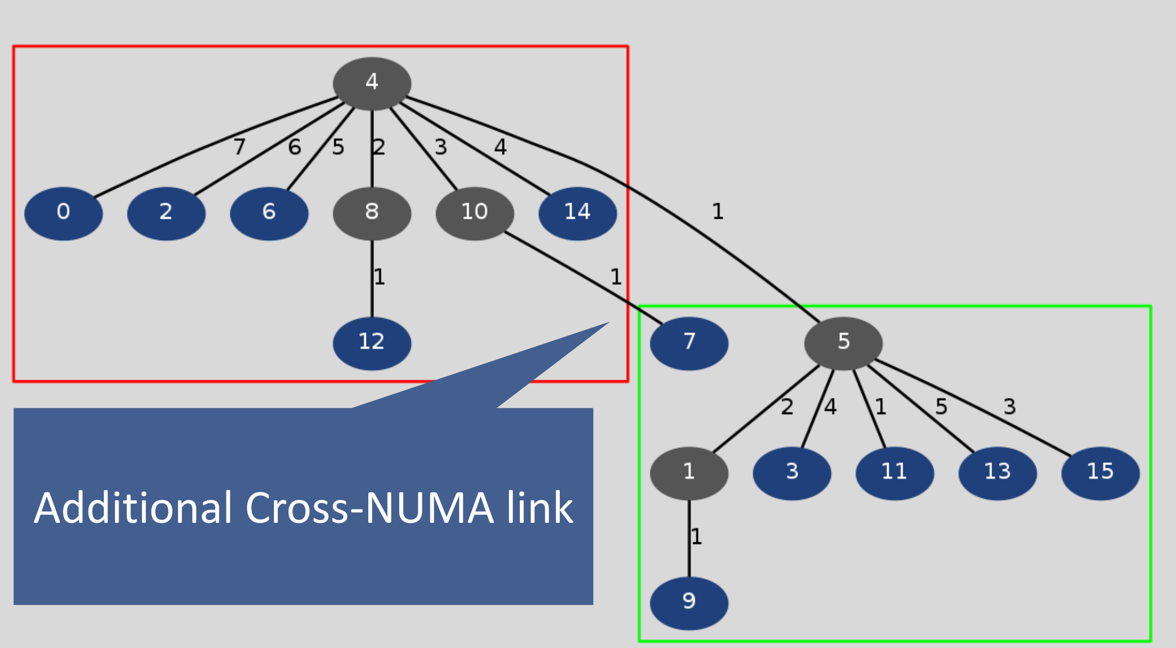


Smelt produces good trees across architectures

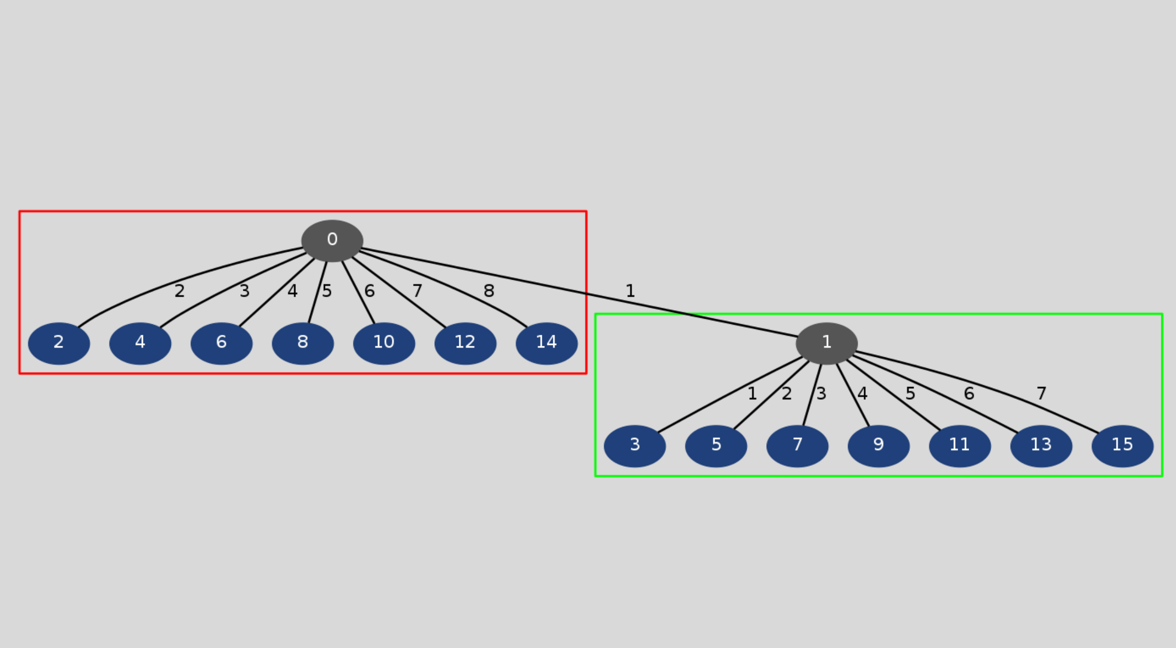


Manufacturer Codename Sockets x Cores x Threads

Fast broadcast trees are good for reductions in most cases

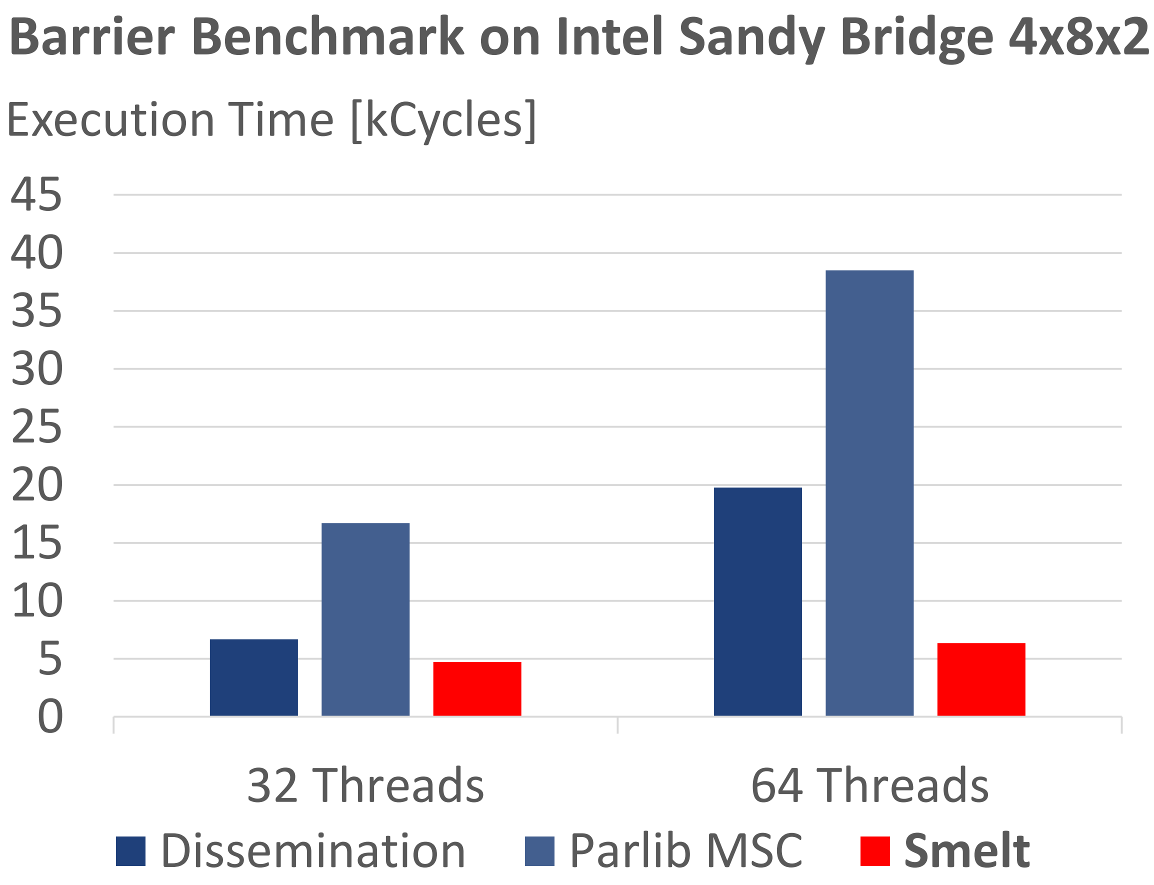


Smelt Tree on Intel Bloomfield 2x4x2



Cluster Topology on Intel Bloomfield 2x4x2

Smelt provides simple and fast barriers



Barriers based on reduction and broadcast

```
void smelt_barrier(void) {
    smelt_reduce();
    smelt_broadcast();
}
```

Simple barrier implementation

OpenMP: EPCC OpenMP Benchmark Collection

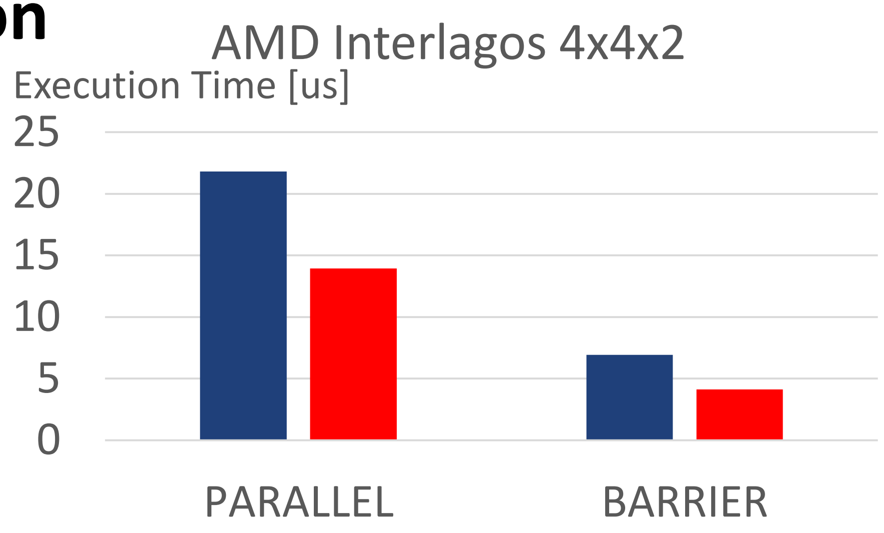
```
/* epcc openmp barrier benchmark */
void testbar() {
    int j;
    #pragma omp parallel private(j)
    {
        for (j = 0; j < innerreps; j++) {
            delay(delaylength);
            #pragma omp barrier
        }
    }
}
```

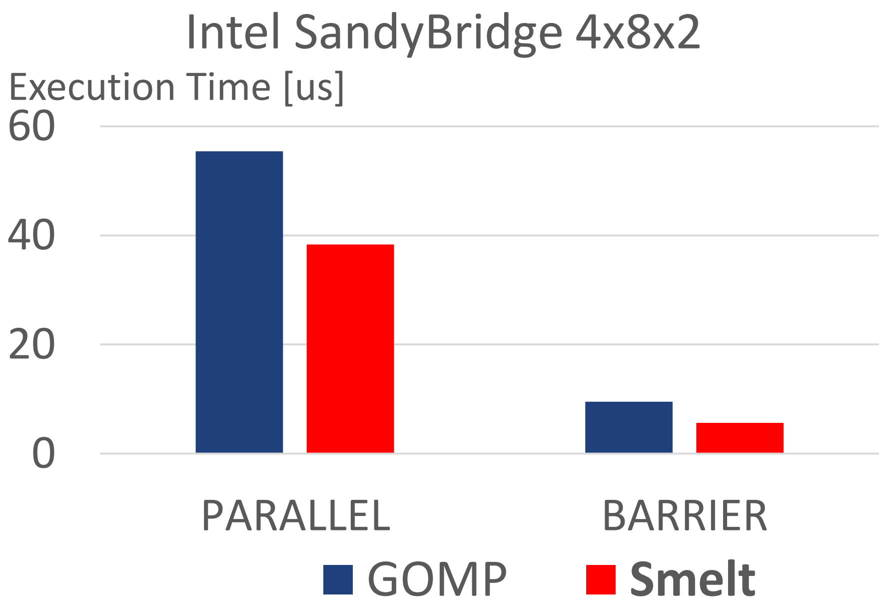
Explicit barrier

Implicit barrier at the end of parallel block

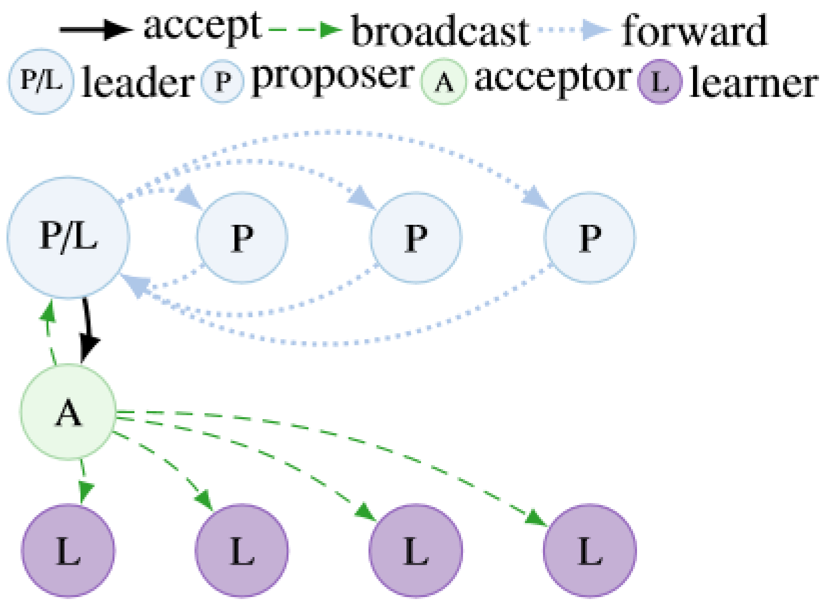
Replaced GOMP barrier with Smelt

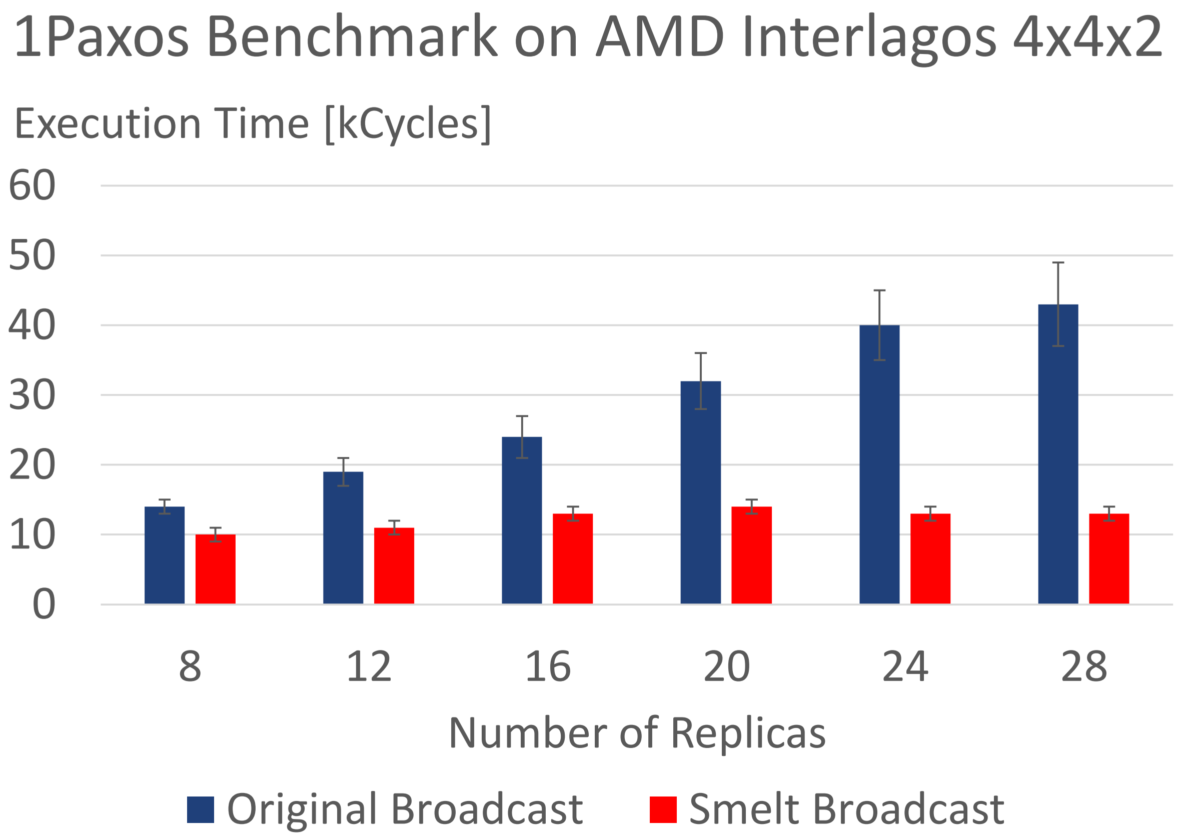
→ Remaining results on the website





Agreement Protocols: 1Paxos



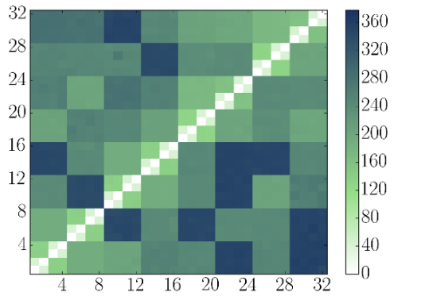


4 clients to generate load

N replicas executing 1Paxos

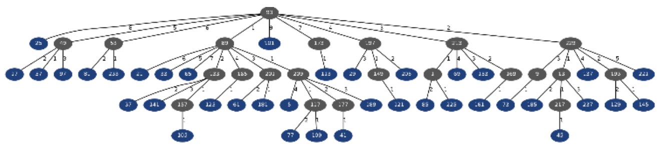
Summary

Talk to us at the first poster session

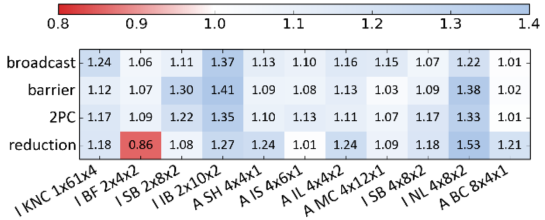


- Broadcasts and reductions are **central building blocks**

- **No globally optimal** tree topology



- Information from hardware discovery is **not sufficient**



	I KNC 1x61x4	I BF 2x4x2	I SB 2x8x2	I IB 2x10x2	A SH 4x4x1	A IS 4x6x1	A IL 4x4x2	A MC 4x12x1	I SB 4x8x2	I NL 4x8x2	A BC 8x4x1
broadcast	1.24	1.06	1.11	1.37	1.13	1.10	1.16	1.15	1.07	1.22	1.01
barrier	1.12	1.07	1.30	1.41	1.09	1.08	1.13	1.03	1.09	1.38	1.02
2PC	1.17	1.09	1.22	1.35	1.10	1.13	1.11	1.07	1.17	1.33	1.01
reduction	1.18	0.86	1.08	1.27	1.24	1.01	1.24	1.09	1.18	1.53	1.21

- Smelt’s produces **good trees**

machinedb.systems.ethz.ch

github.com/libsmelt