Scalable Membarrier for Multicore Systems

Goal: Low Latency on Multicores

- Linux membarrier syscall uses multicast to force barriers

- used by user-space Read-Copy-Update library
- multicast dominates latency on multicore systems

- tree topologies → logorithmic scaling with number of cores
- dynamic receiver groups → maintenance cost for multicast trees

→ in practice: just sequential propagation with linear scaling

Evaluation @ HPI Future SOC Lab

- 4-sockets, 72 physical cores, 144 logical cores (hyperthreads)
- Intel Xeon CPU E7-8890 v3 processor

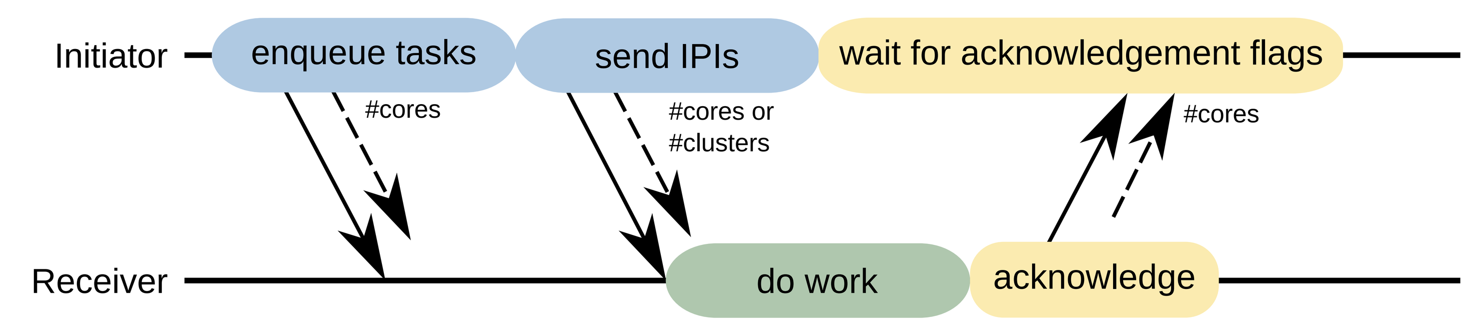
- Ubuntu 16.04
- Linux kernel version 4.16.7 extended with new multicast implementation
- modified kernel is evaluated bare-metal to exclude noise from hypervisors

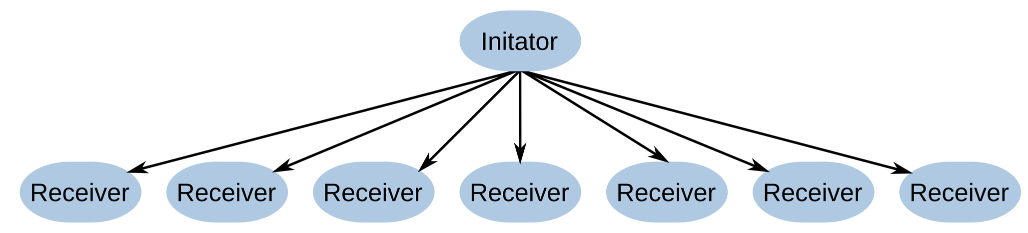
Conclusions

- logarithmic scaling over #cores works
- latency improvement for membarrier syscall
- little impact on throughput
- also usable for "faster" TLB shootdown

- searching for real-world applications:
  - many threads in shared address space
  - multiple threads waiting for completion or waiting anyway

Old: Flat Sequential Multicast in Linux Kernel

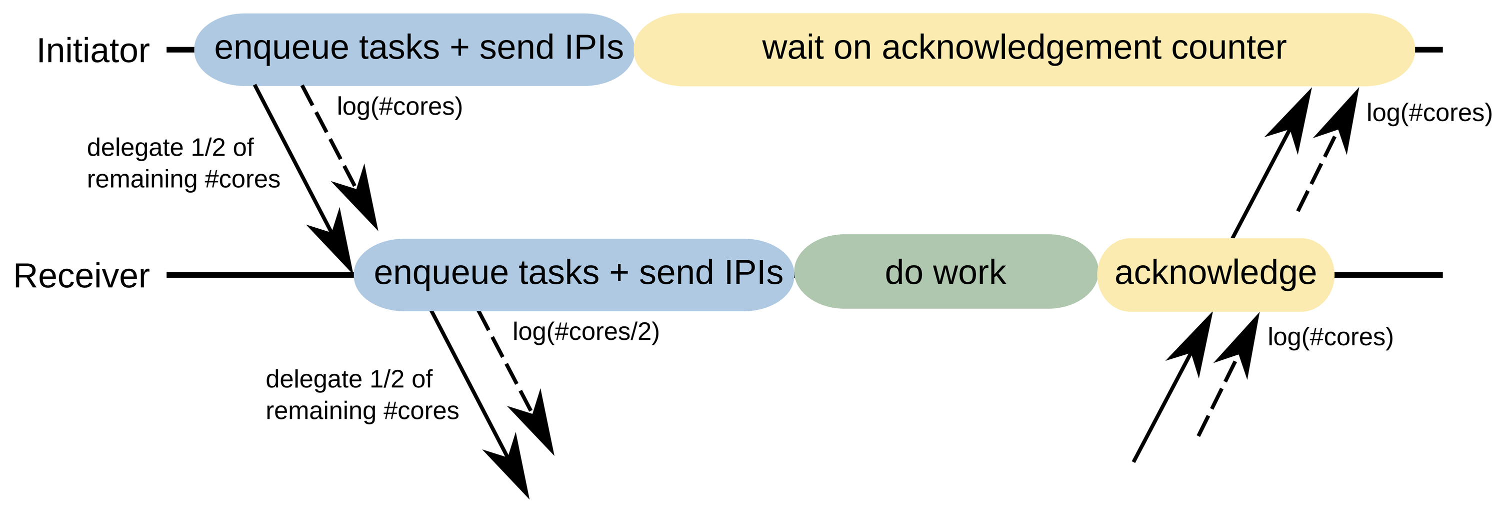


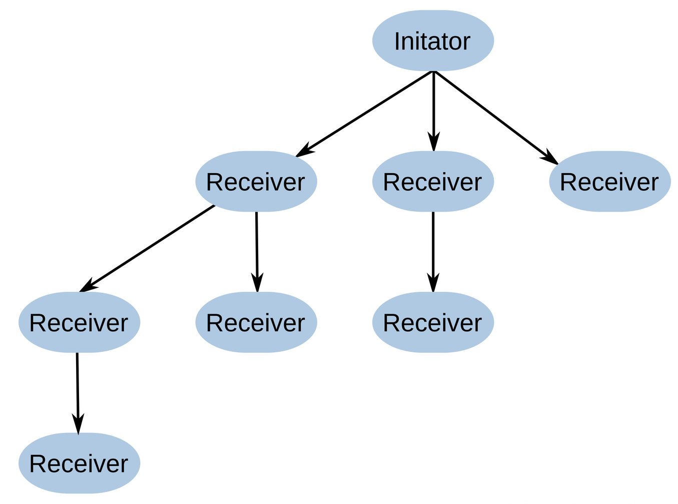


Propagation
- for each receiver:
  - enqueue message
- send IPIs to all receivers

Acknowledgement
- for each receiver:
  - wait for flag

New: On-the-fly Binomial Tree Multicast





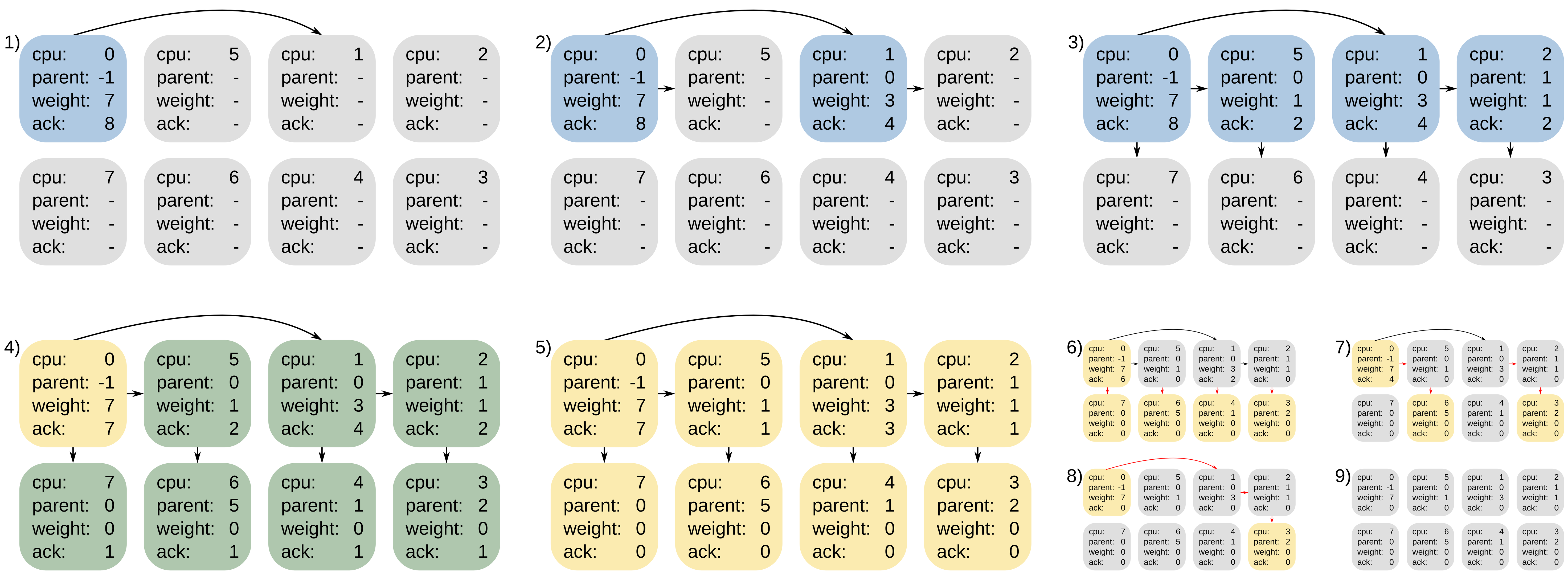
Recursive propagation based on [1]
- until set of receivers is empty:
  - remove one as child
  - send message + IPI to child and delegate half of remaining receivers
- perform the task locally

Acknowledgement based on [2]

use on-the-fly tree for software tree combining:
- decrement counter for pending acks from children
- if zero: propagate to ancestors via shared mem.

[1] Bruck, Herlihy, and Kogan, On the design and implementation of broadcast and global combine operations using the postal model, 1996
[2] Tang and Yew, Software combining algorithms for distributing hot-spot addressing, 1990

Example for Tree-based Multicast



Steps

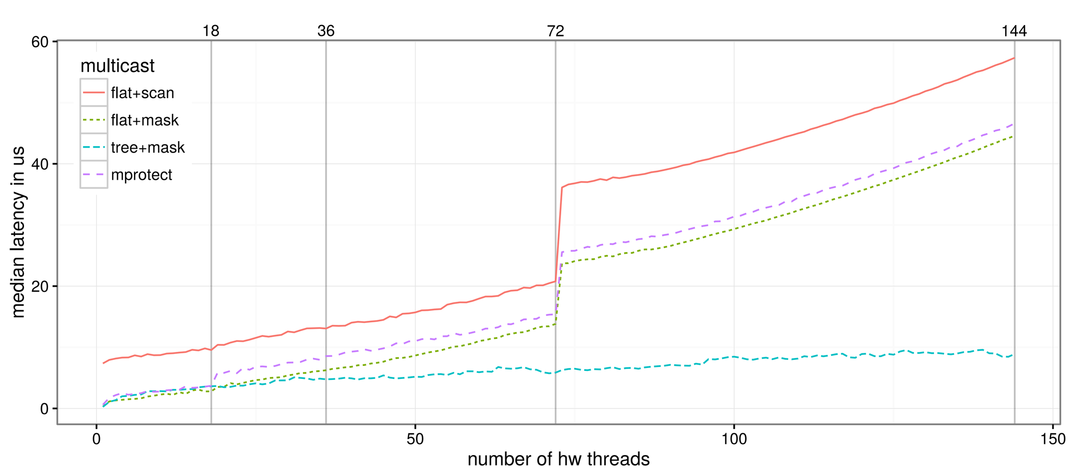
1-3) message is propogated

4) work is done, initiator decrements local ack counter

5) recievers decrement local ack counter

6-9) propagate acknowledgement towards root

Latency vs. Receivers



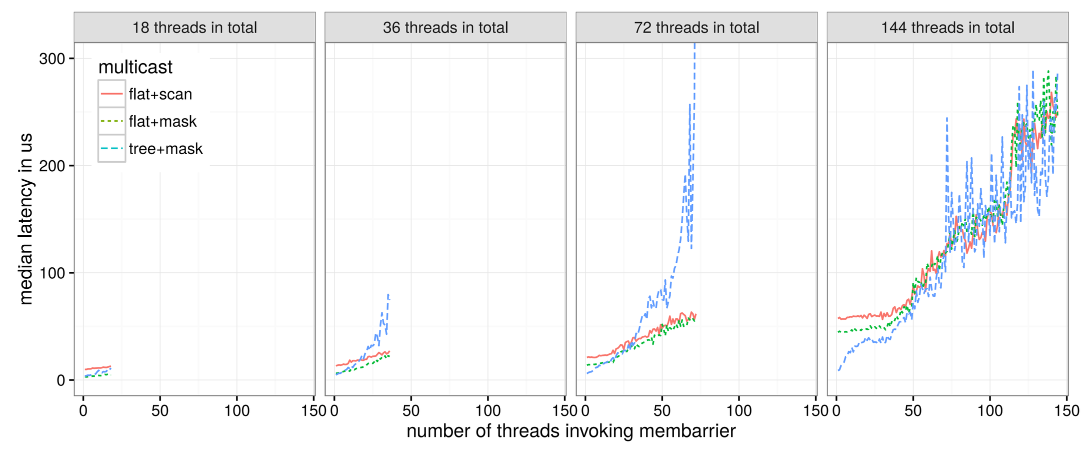
Median latency for growing number of receivers.

Flat variants and mprotect TLB shootdown:
- mostly linear scaling
- drastic latency increase when including 2nd hyperthread

Tree-based multicast:
- roughly logarithmic scaling
- robust agains including 2nd hyperthread

- crossing NUMA domains: no notable effect

Latency vs. Concurrency

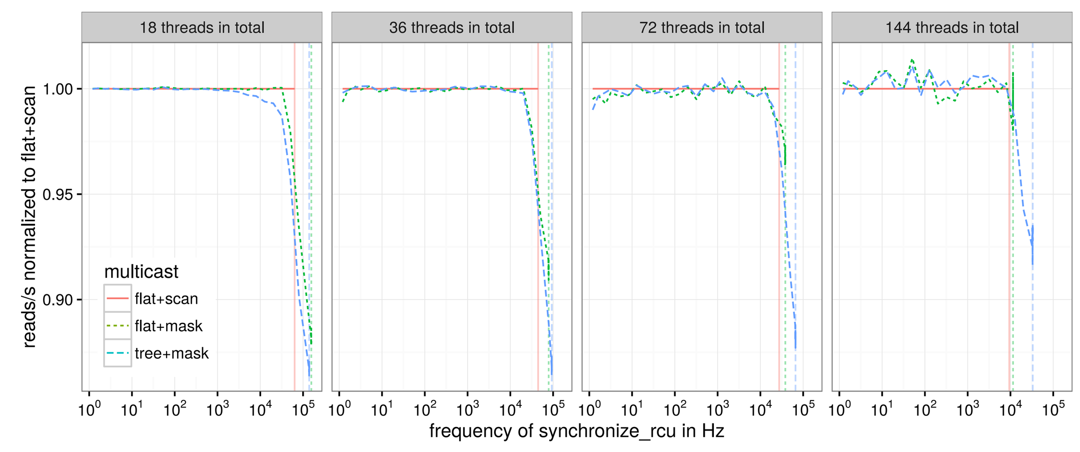


Median latency for growing number of of concurrent initiators.

Flat variants:
- flat variants scale linearly with the number of concurrent multicasts
- only initiator spends time for sending the multicast messages
- concurrent multicasts are perfectly parallelized

Tree-based multicast:
- visible impact and jitter
- part of the effort for propagation is delegated to receivers
- concurrent multicasts compete on these helpers

Impact on uRCU Throughput



uRCU reader throughput speedup with varying frequency of write operations.
Normalized to the throughput of standard implementation.

- major increase in maximal frequency of synchronize_rcu
- minor impact (< 1.25%) on reader throughput up to 10kHz
- decrease < 10% at maximal frequency of standard implementation
- decrease < 15% at maximal frequency for tree-based multicast

Brandenburg University of Technology Cottbus - Senftenberg

Robert Kuban, Randolf Rotta, Jörg Nolte
Distributed Systems & Operating Systems Group
Brandenburg University of Technology Cottbus-Senftenberg, Cottbus,Germany
Email: {robert.kuban, randolf.rotta, joerg.nolte}@b-tu.de

Deutsche Forschungsgemeinschaft
This work was supported by the German Research Foundation (DFG) under project no. 224615364.