Pay Migration Tax to Homeland: Anchor-based Scalable Reference Counting for Multicores

Seokyong Jung, Jongbin Kim, Minsoo Ryu, Sooyong Kang, Hyungsoo Jung

Hanyang University

Reference counting

- It is
  - a general technique to manage the number of references for resources
  - mainly used to reclaim resources in timely manner
- Scalability is the most important challenge in multicore environment

1. REF (increase counter)
2. Use resource
3. UNREF (decrease counter)

Known scalability issues of reference counting in Linux

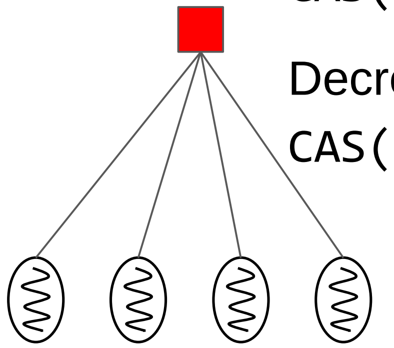
Thread 🟥 Atomic counter ⬜ Counter

Increment:

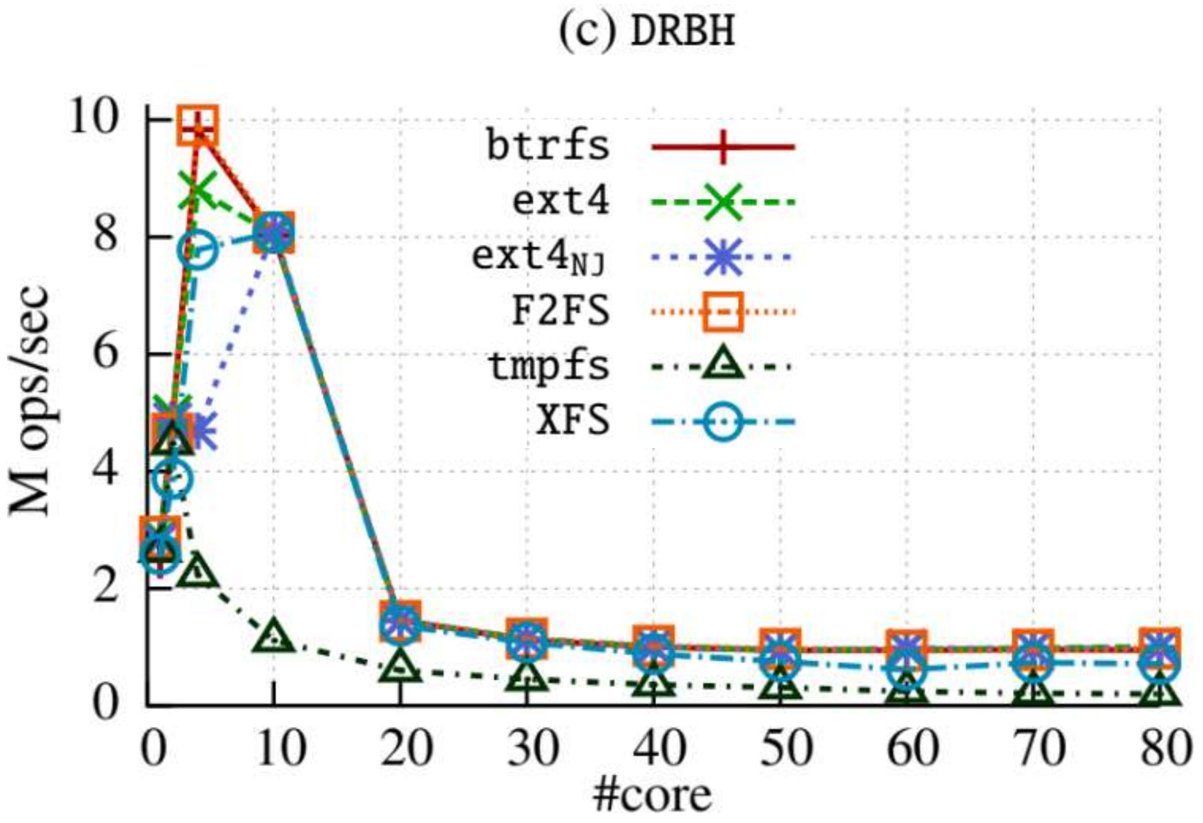
`CAS(counter,v,v+1)`

Decrement:

`CAS(counter,v,v-1)`

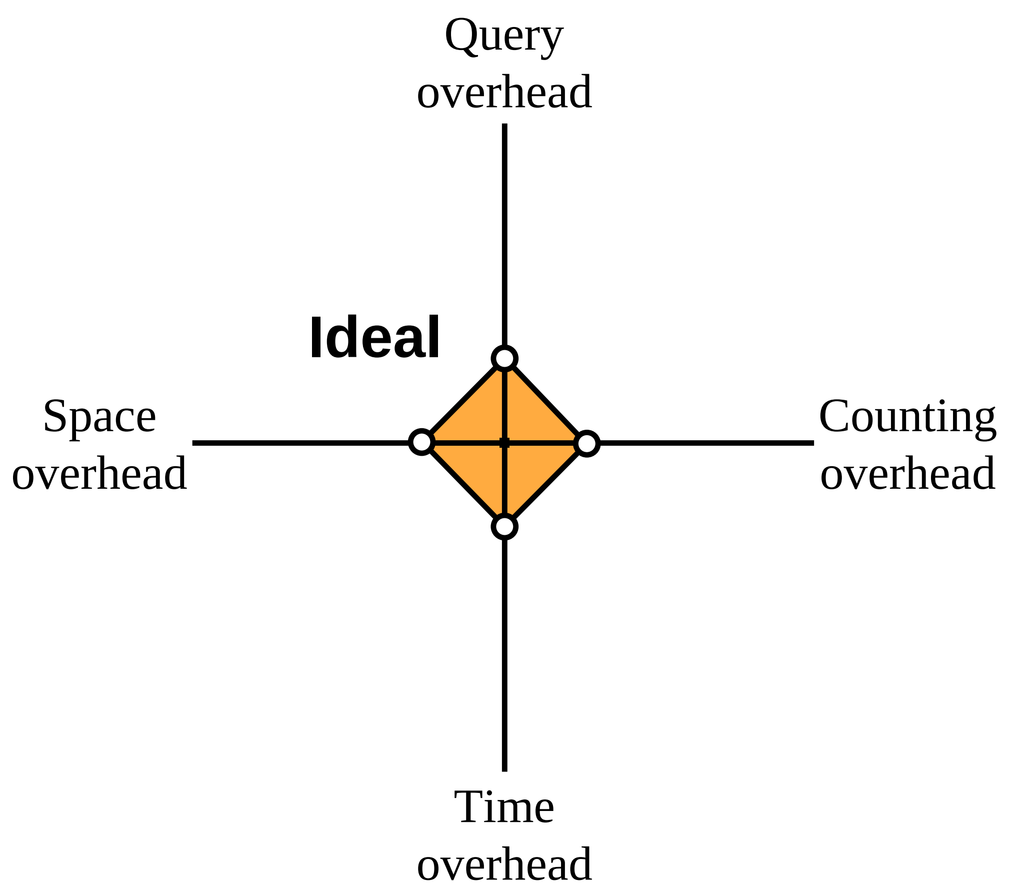


Traditional Counting in Linux



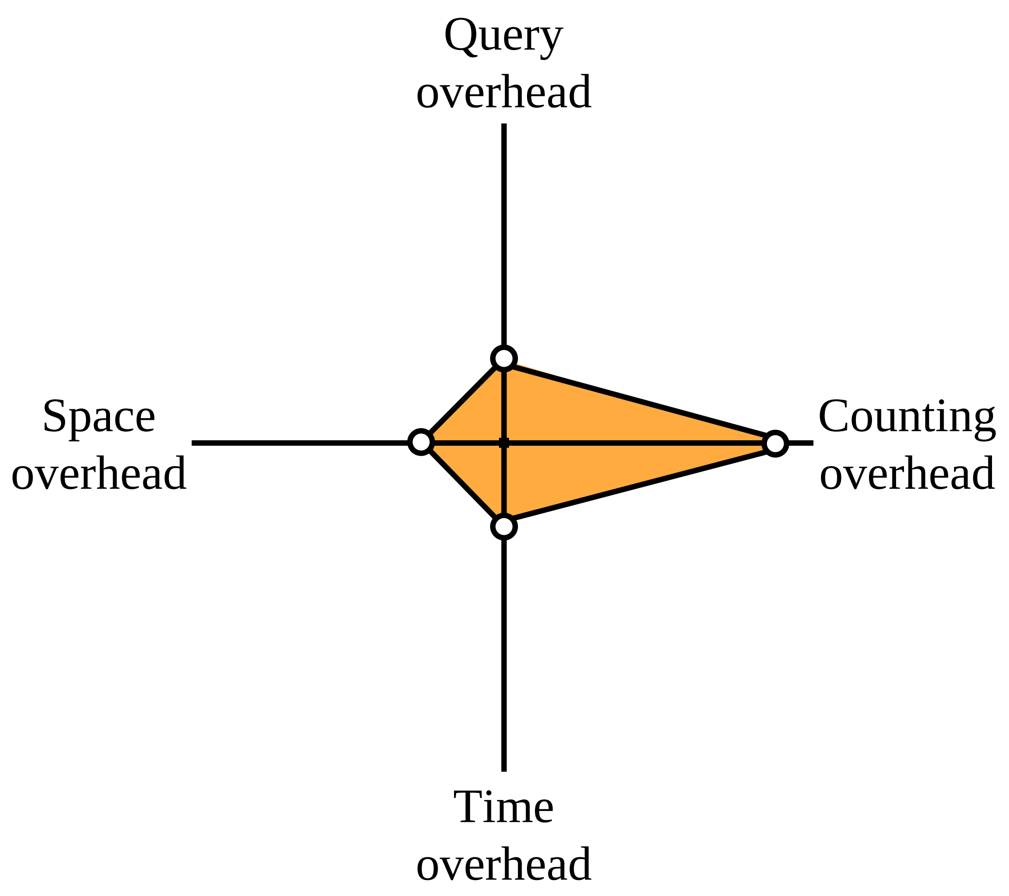
Read throughput for a same page in a single file
(Min et al. ATC'16)

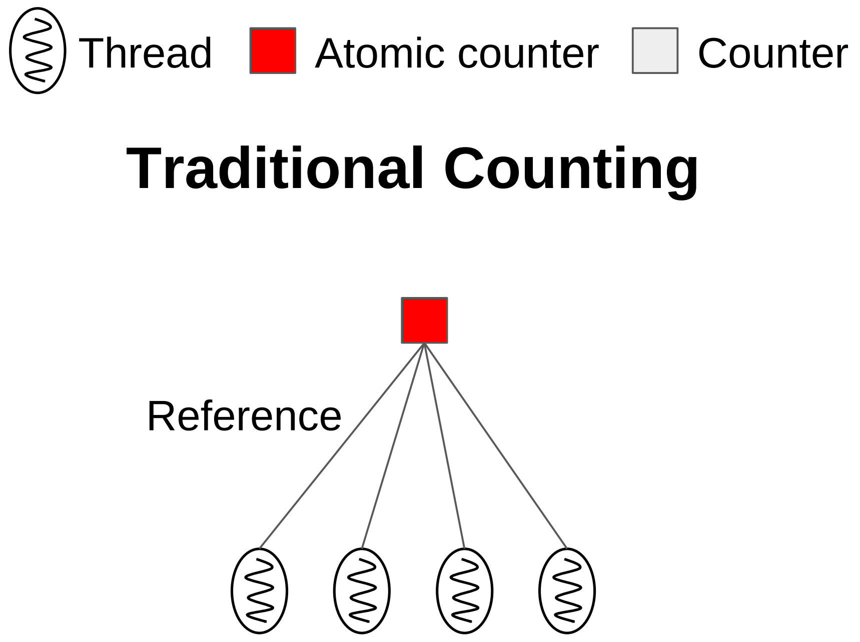
Four performance metrics we established



- Counting Overhead
  - Cost for updating a reference counter
- Query Overhead
  - Cost for checking if a reference counter is zero
- Space Overhead
  - Space required for reference counter itself
- Time Overhead
  - Time for synchronizing between internal structures for reference counting

Overhead analysis of prior proposals





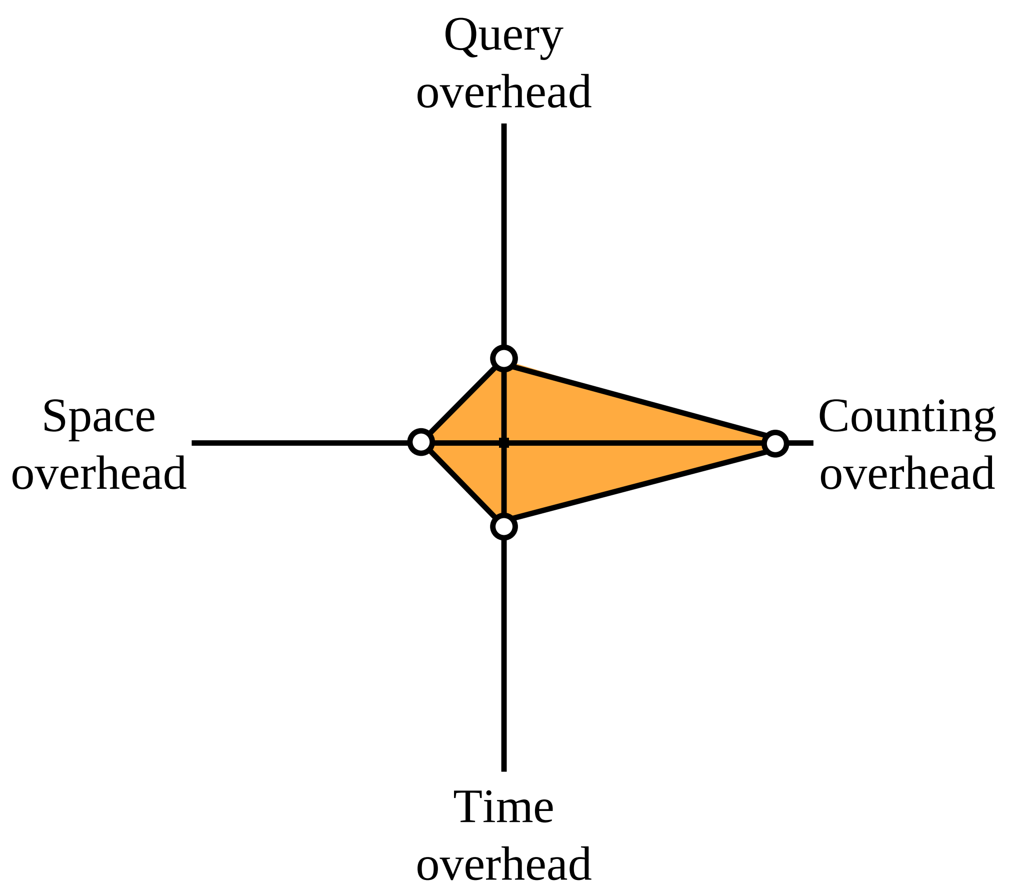
✓ Low query overhead

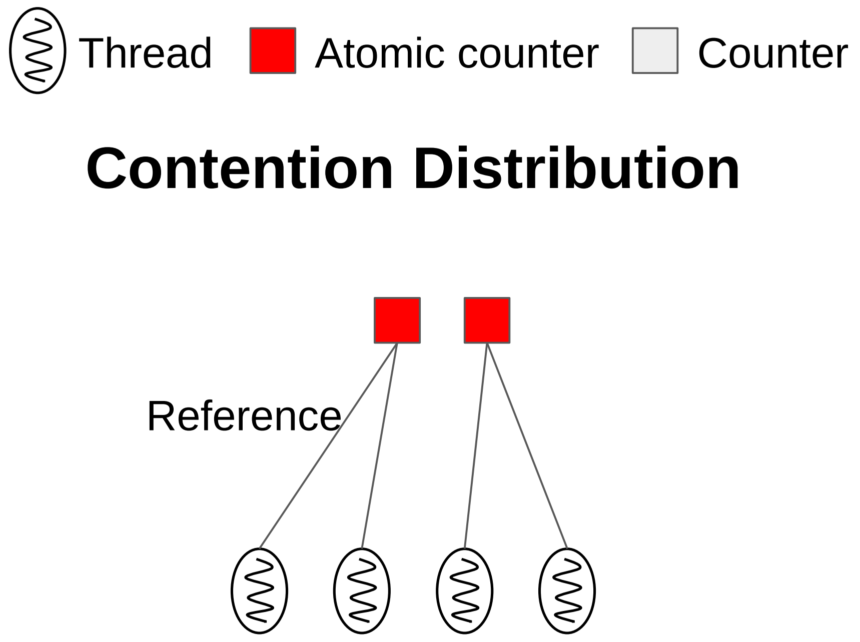
✓ Low space overhead

✓ Low time overhead

✗ High counting overhead

Overhead analysis of prior proposals (cont.)





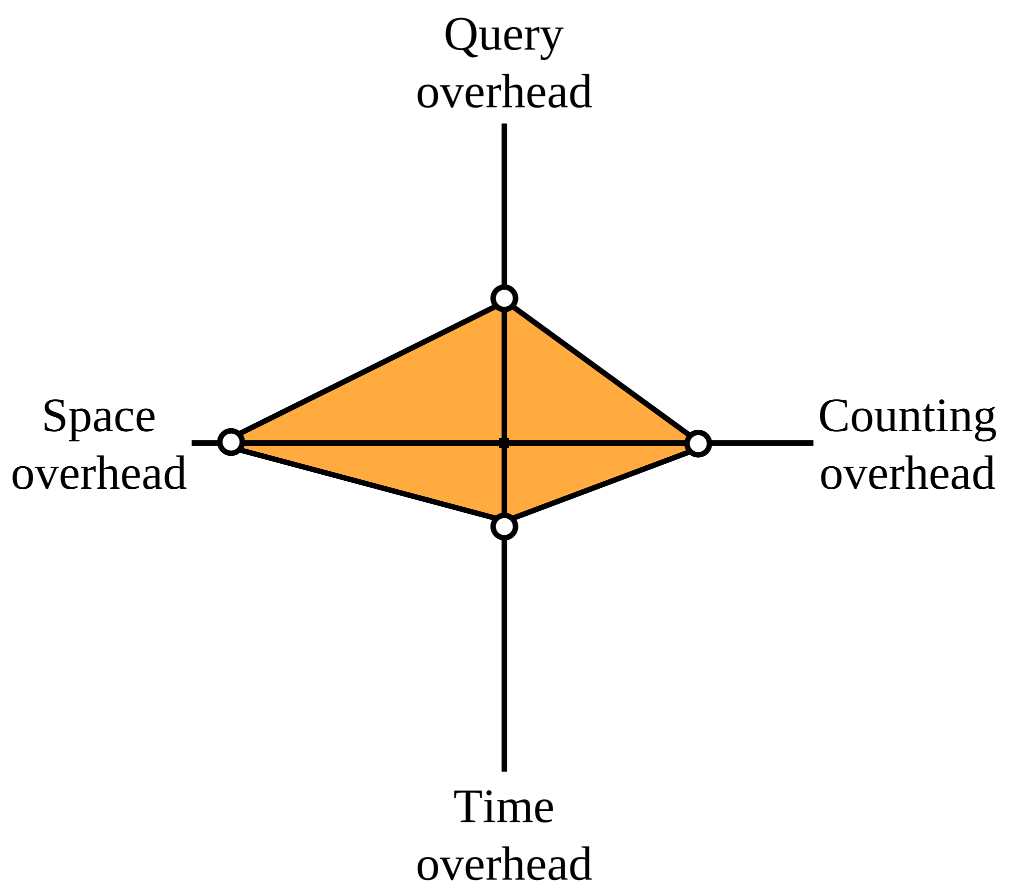
Contention Distribution

✓ Better counting overhead

✗ Worse space overhead

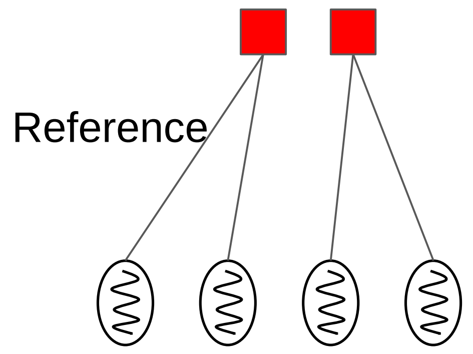
✗ Worse query overhead

Overhead analysis of prior proposals (cont.)



Prior proposals:
SNZI (Ellen et al., SOSP'07)
Carrefour (Dashti et al., SIGPLAN Notices (2013))
Doppel (Nurula et al., OSDI'14)
Dynamic SNZI (Acar et al., PPoPP'17)

Contention Distribution

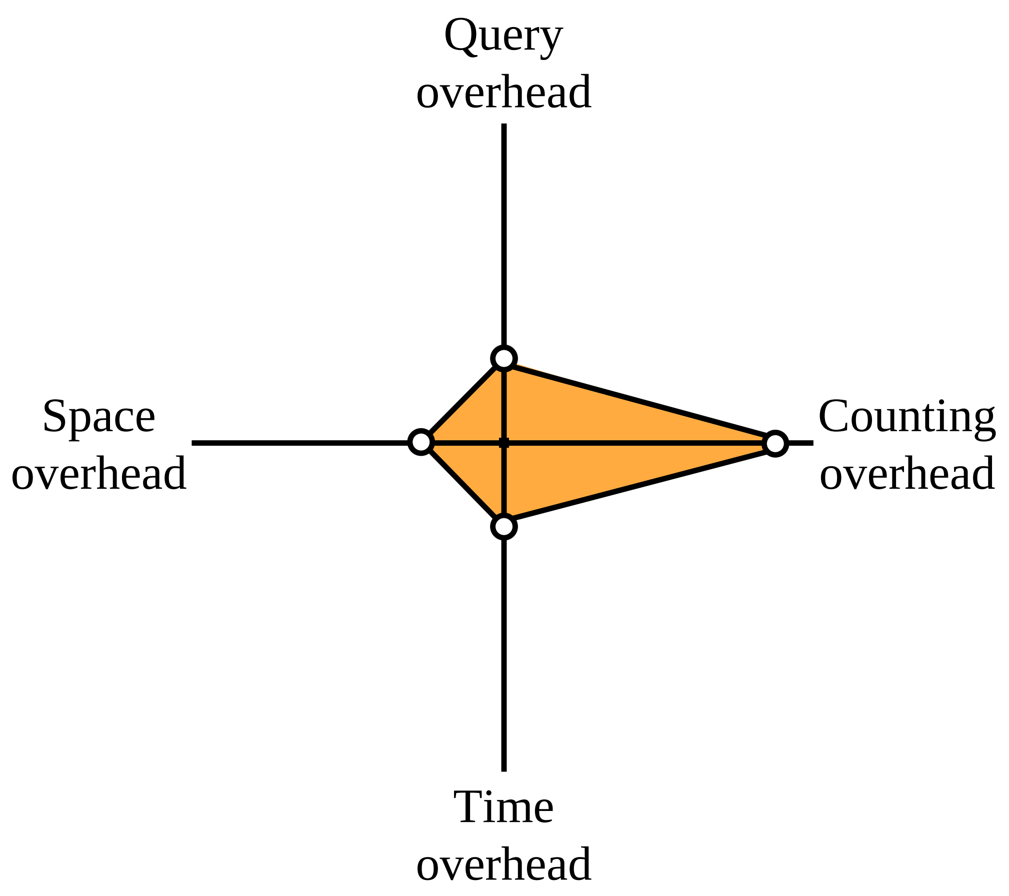


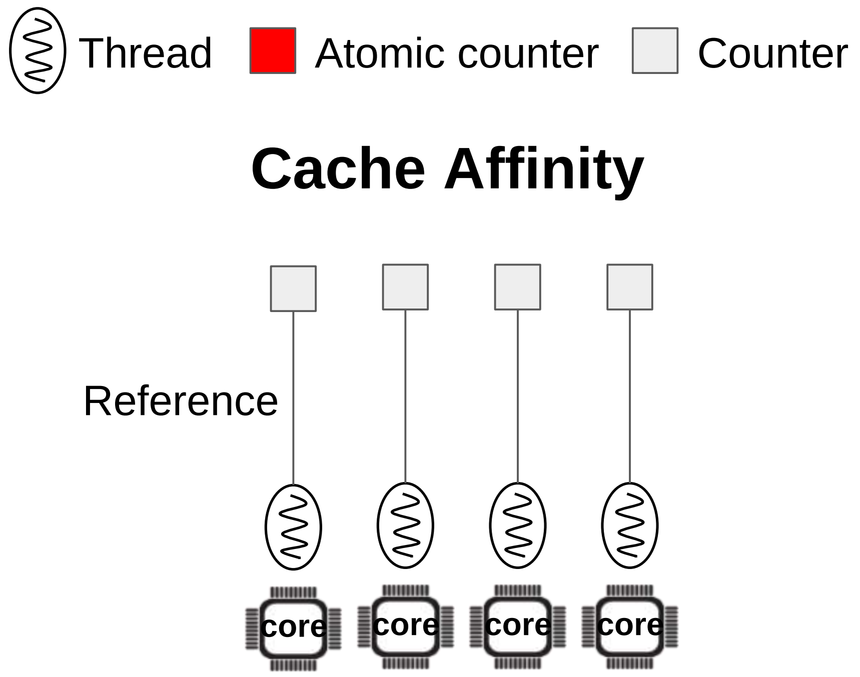
✓ Better counting overhead

✗ Worse space overhead

✗ Worse query overhead

Overhead analysis of prior proposals (cont.)



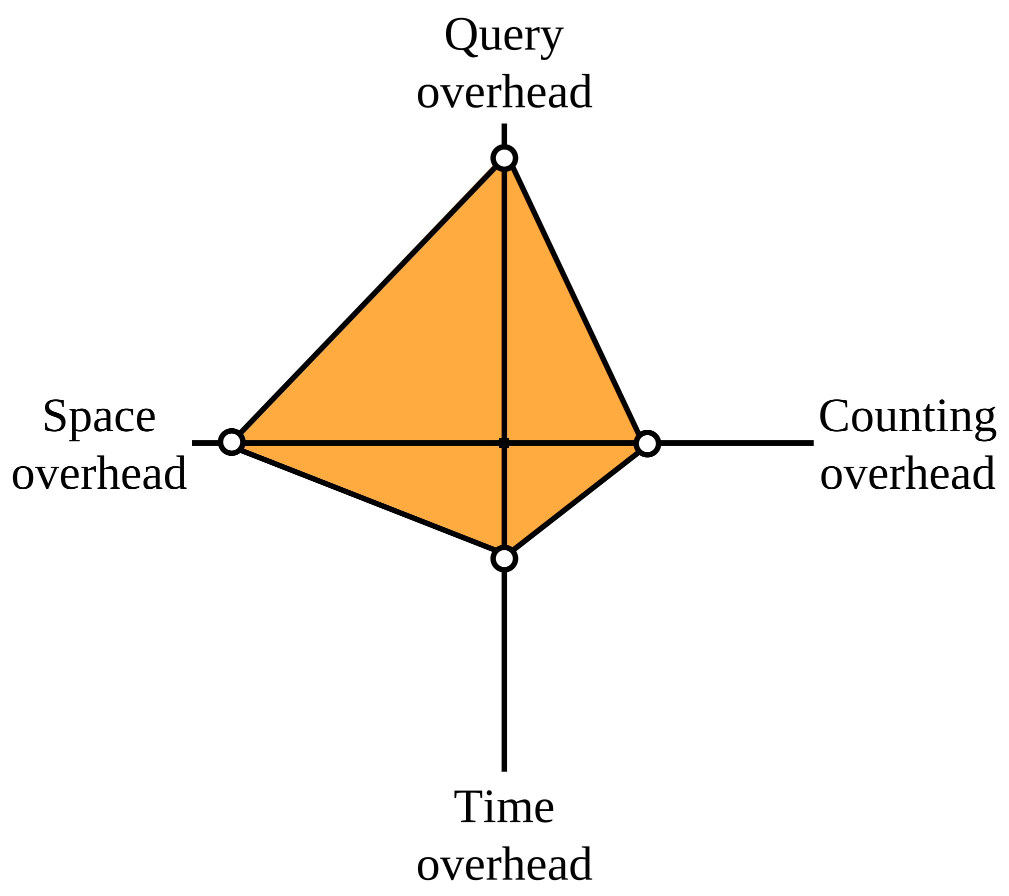


✓ Better counting overhead

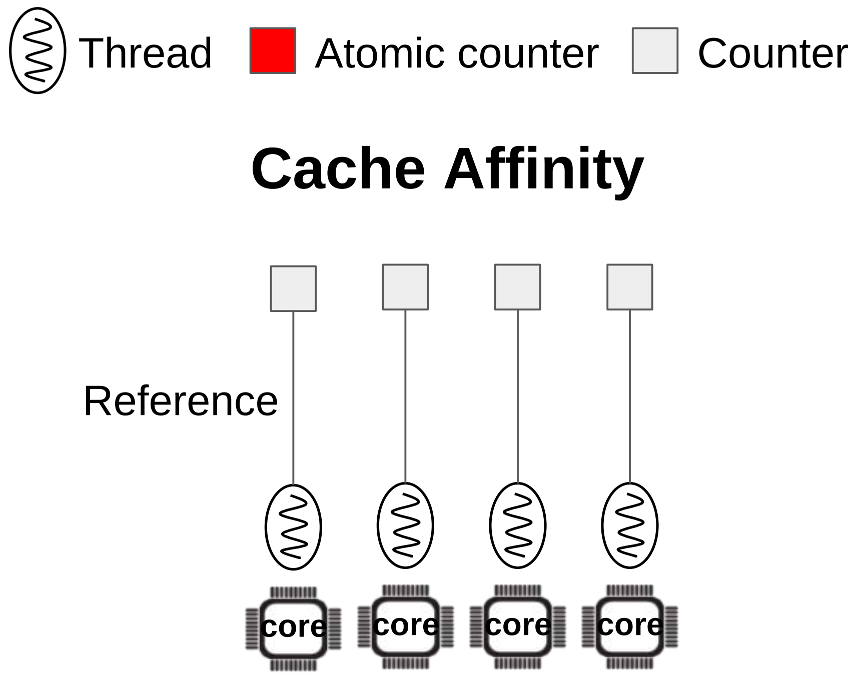
✗ Worse space overhead

✗ Worse query overhead

Overhead analysis of prior proposals (cont.)



Prior proposals:
`percpu_counter` structure in Linux (2006)
Sloppy counter (Boyd-Wickizer et al., OSDI'10)
`percpu_ref` structure in Linux (2013)

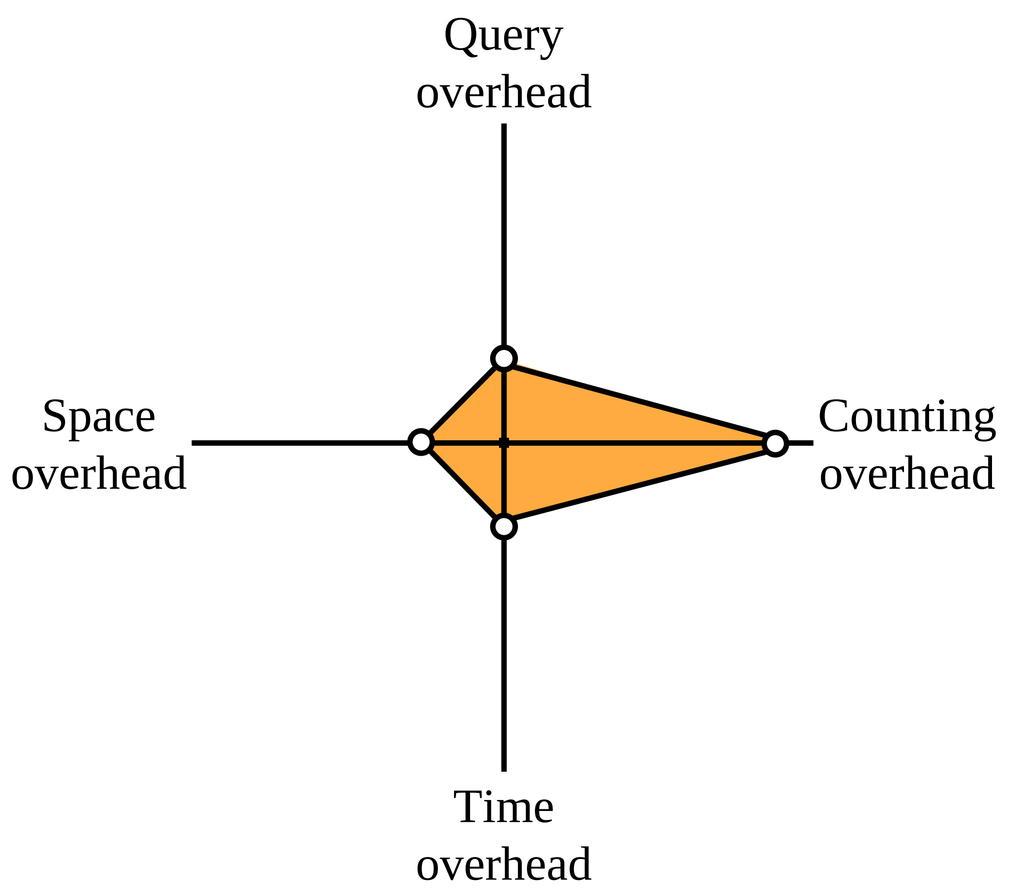


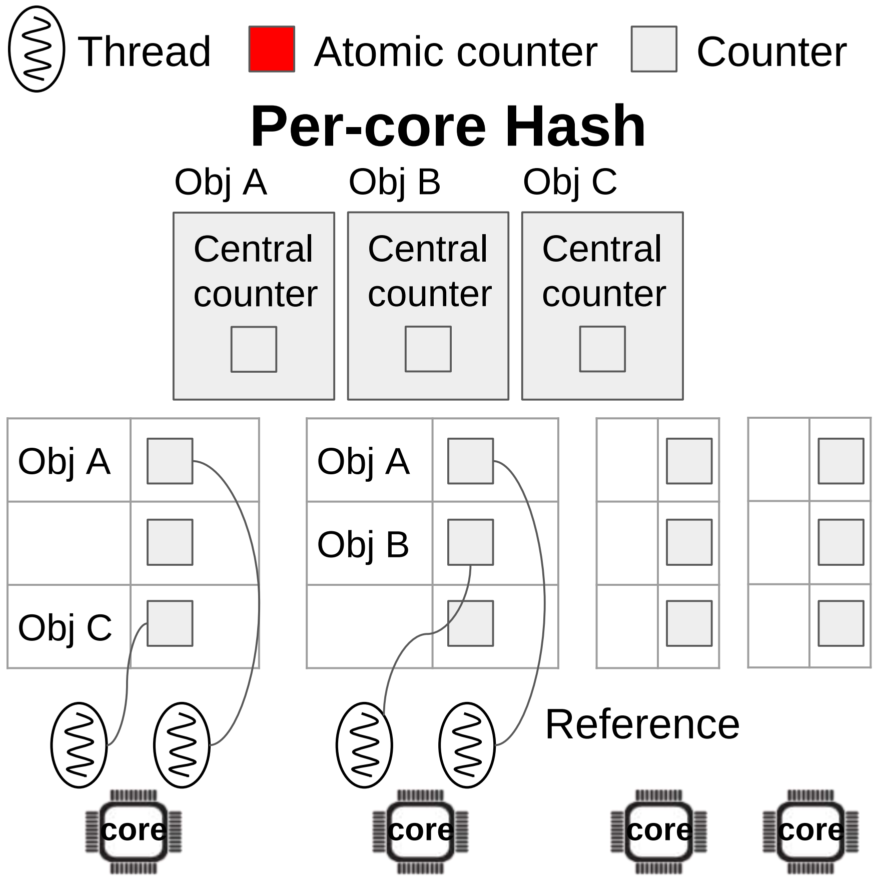
✓ Better counting overhead

✗ Worse space overhead

✗ Worse query overhead

Overhead analysis of prior proposals (cont.)



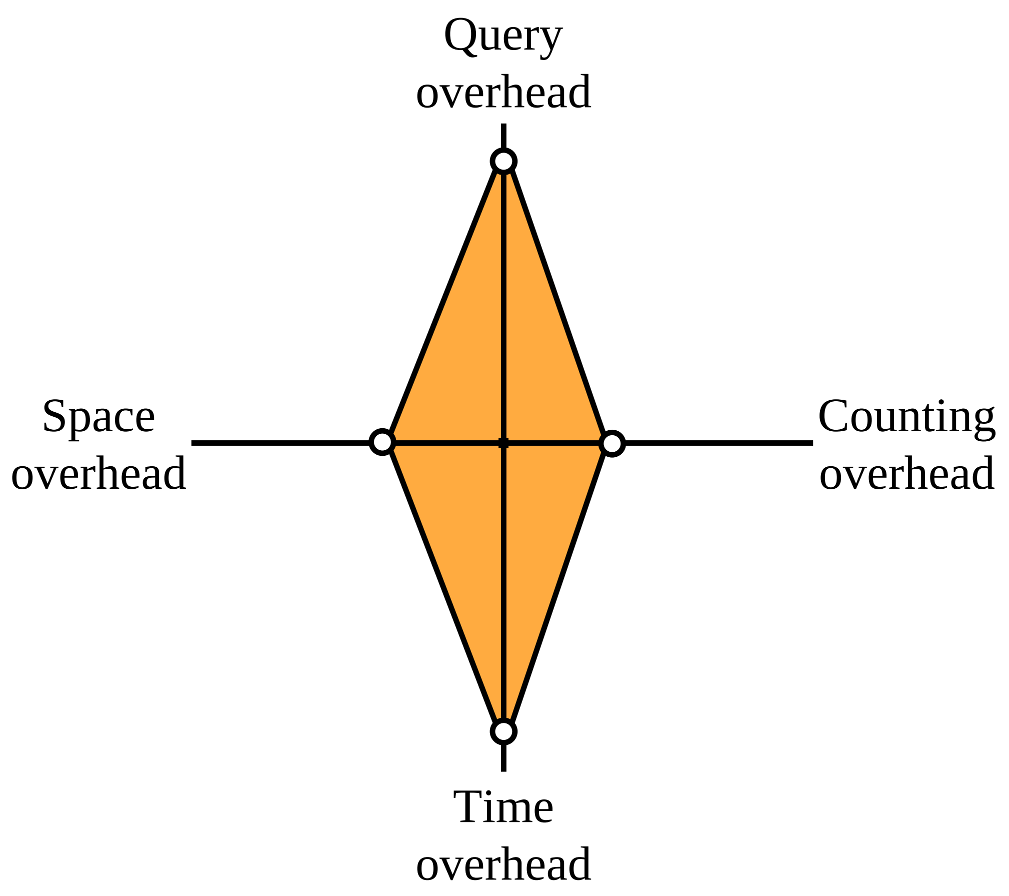


✓ Better counting overhead

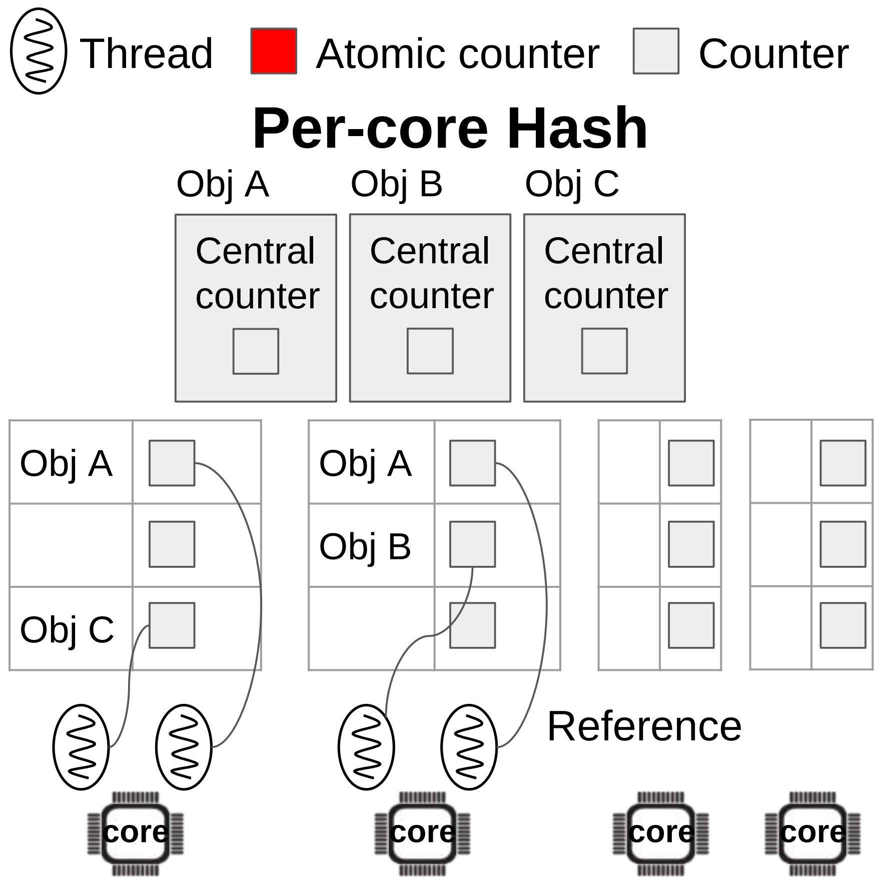
✗ Worse query overhead

✗ Worse time overhead

Overhead analysis of prior proposals (cont.)



Prior proposals:
Reference counting using Linux RCU (Mckenny et al., TR (2013))
RadixVM (Clements et al., EuroSys'13)
OpLog (Boyd-Wickizer, PhD thesis (2014))
ScaleFS (Bhat et al., SOSP'17)

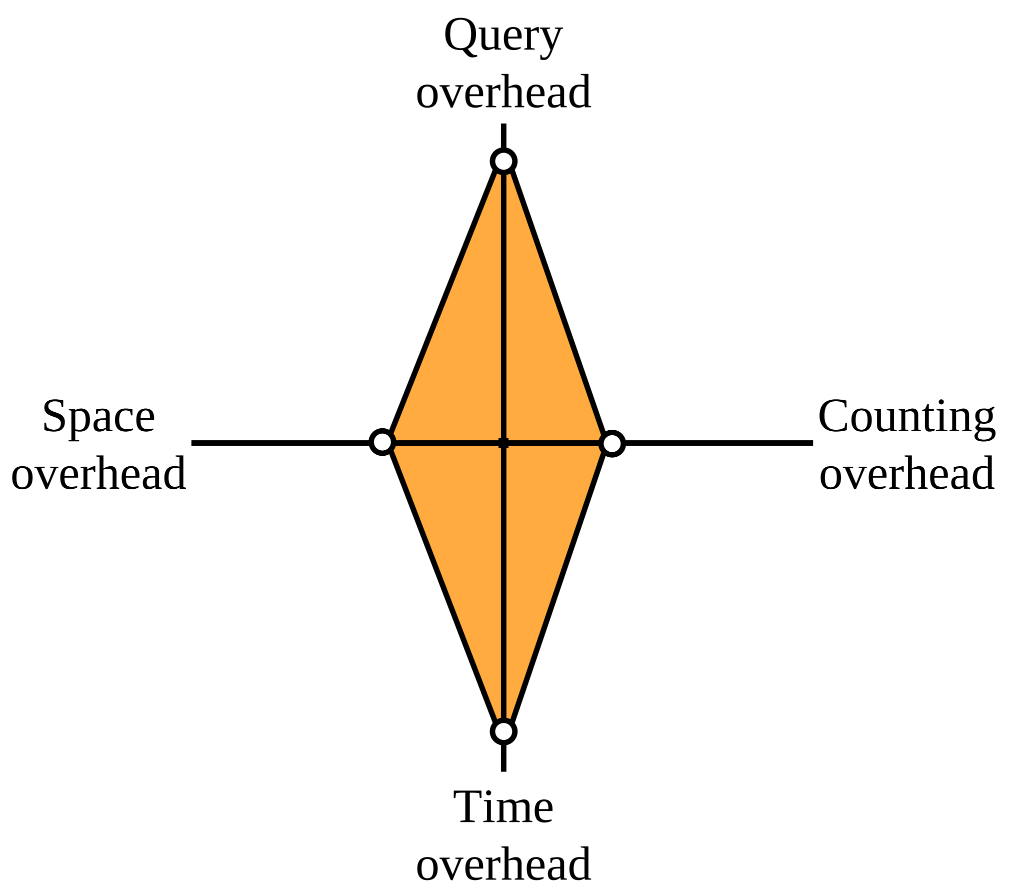


✓ Better counting overhead

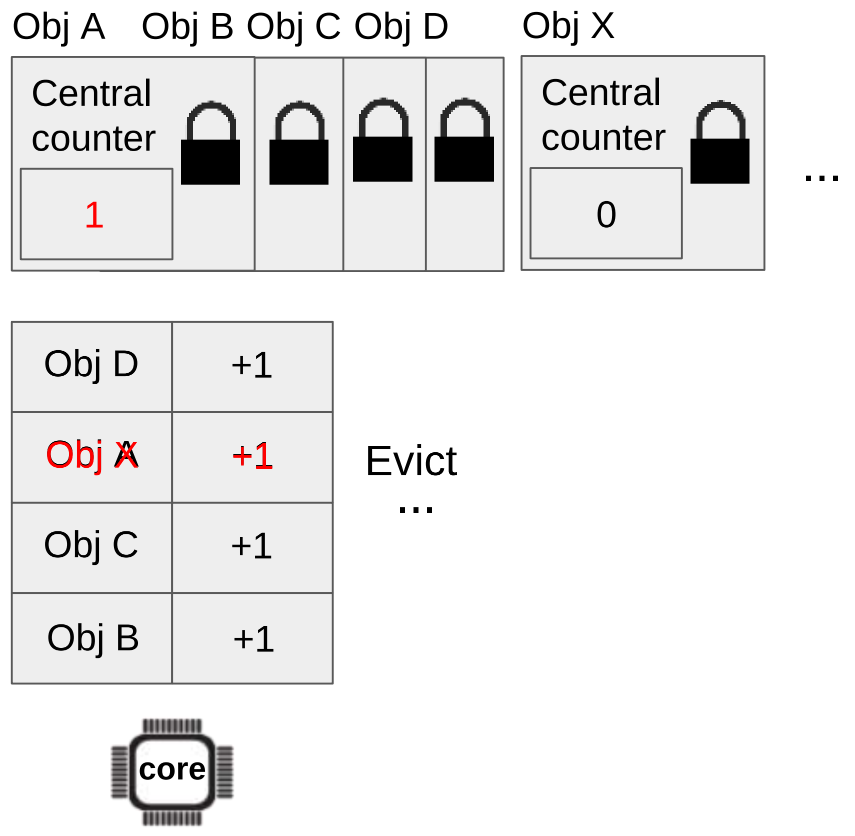
✕ Worse query overhead

✕ Worse time overhead

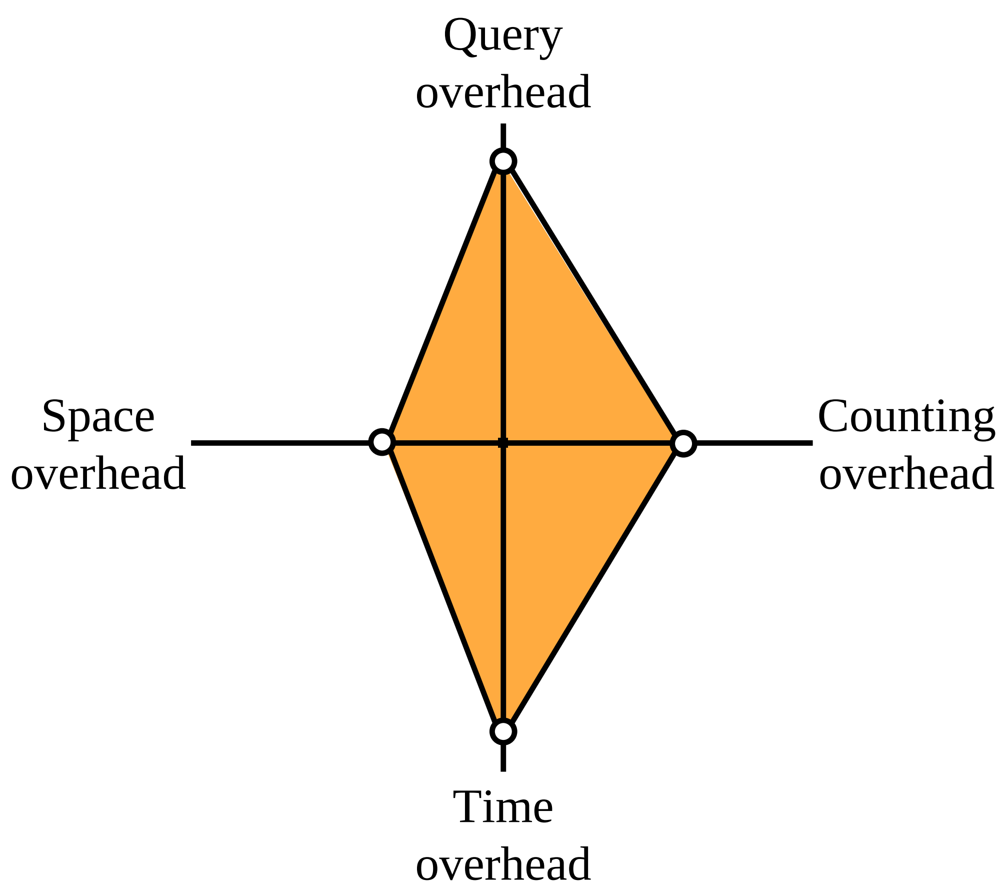
Overhead analysis of prior proposals (cont.)

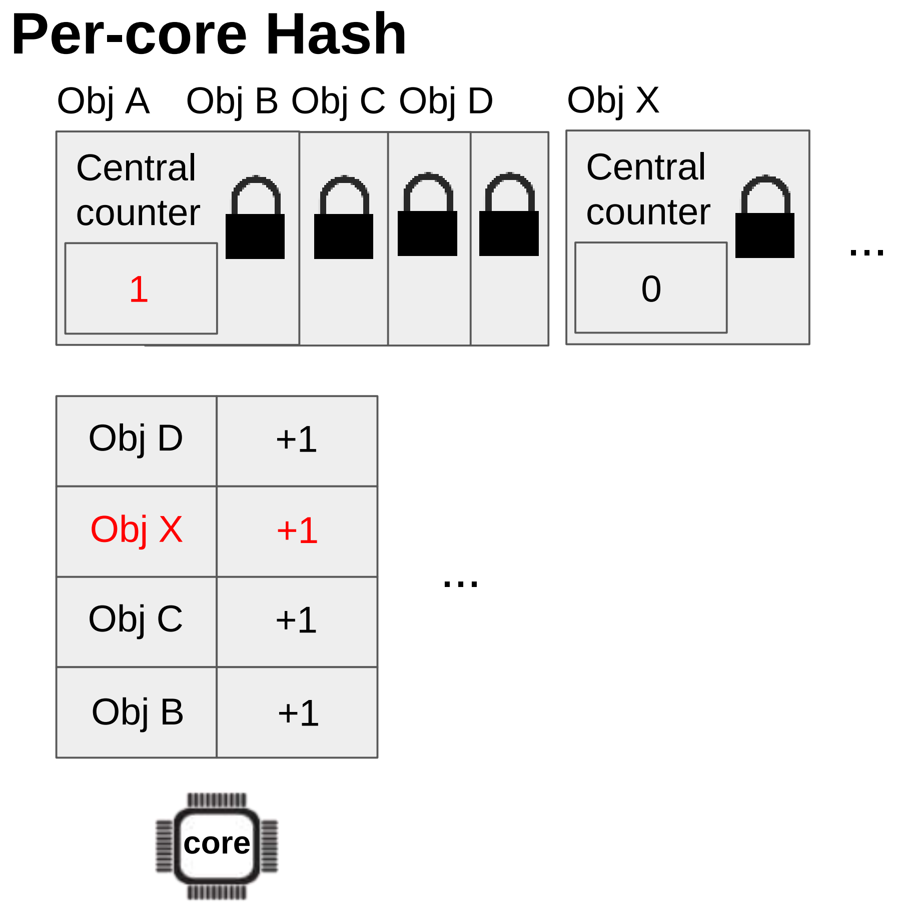


Per-core Hash

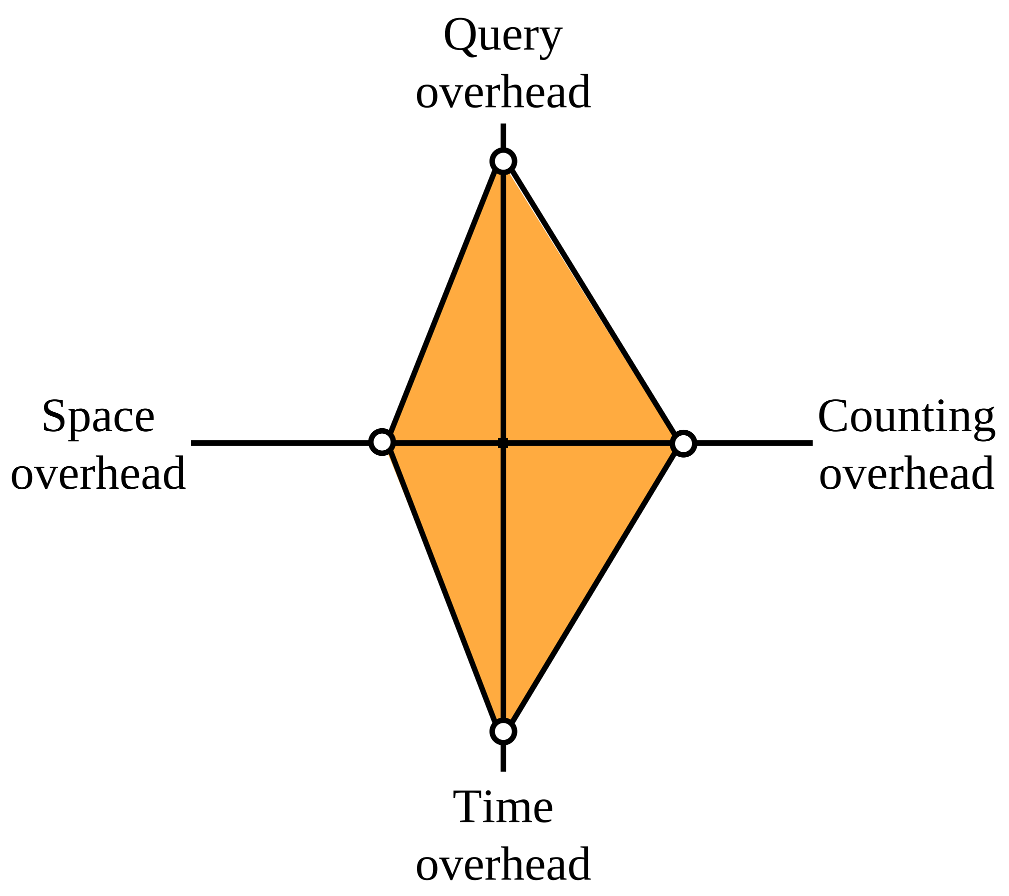


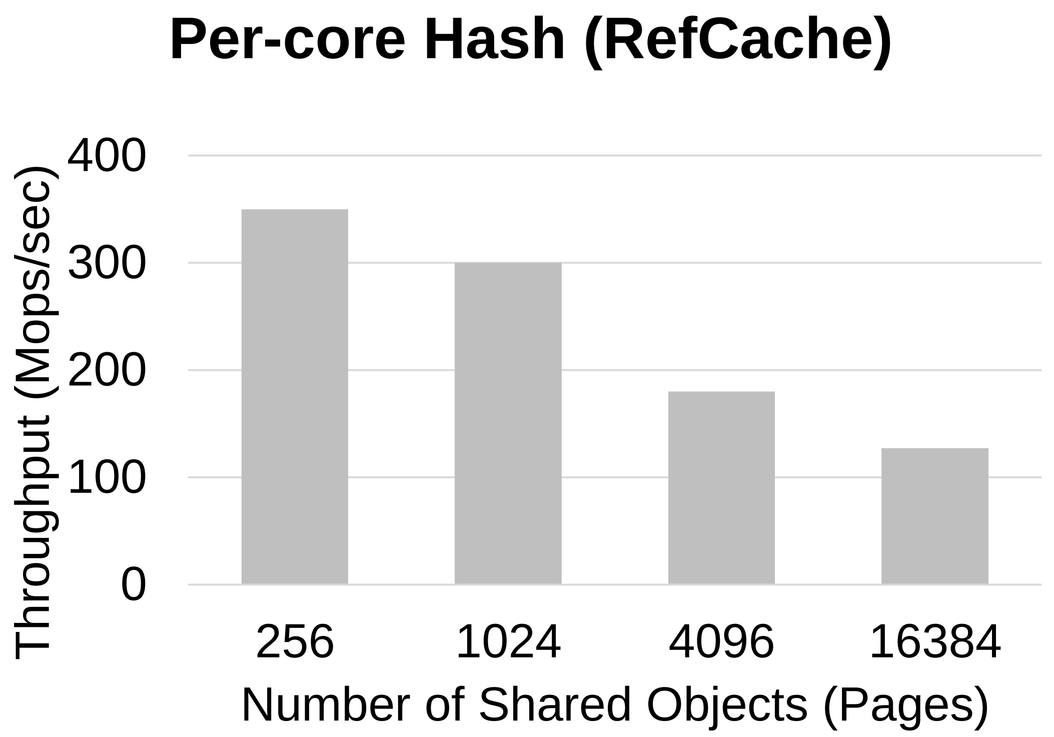
Overhead analysis of prior proposals (cont.)



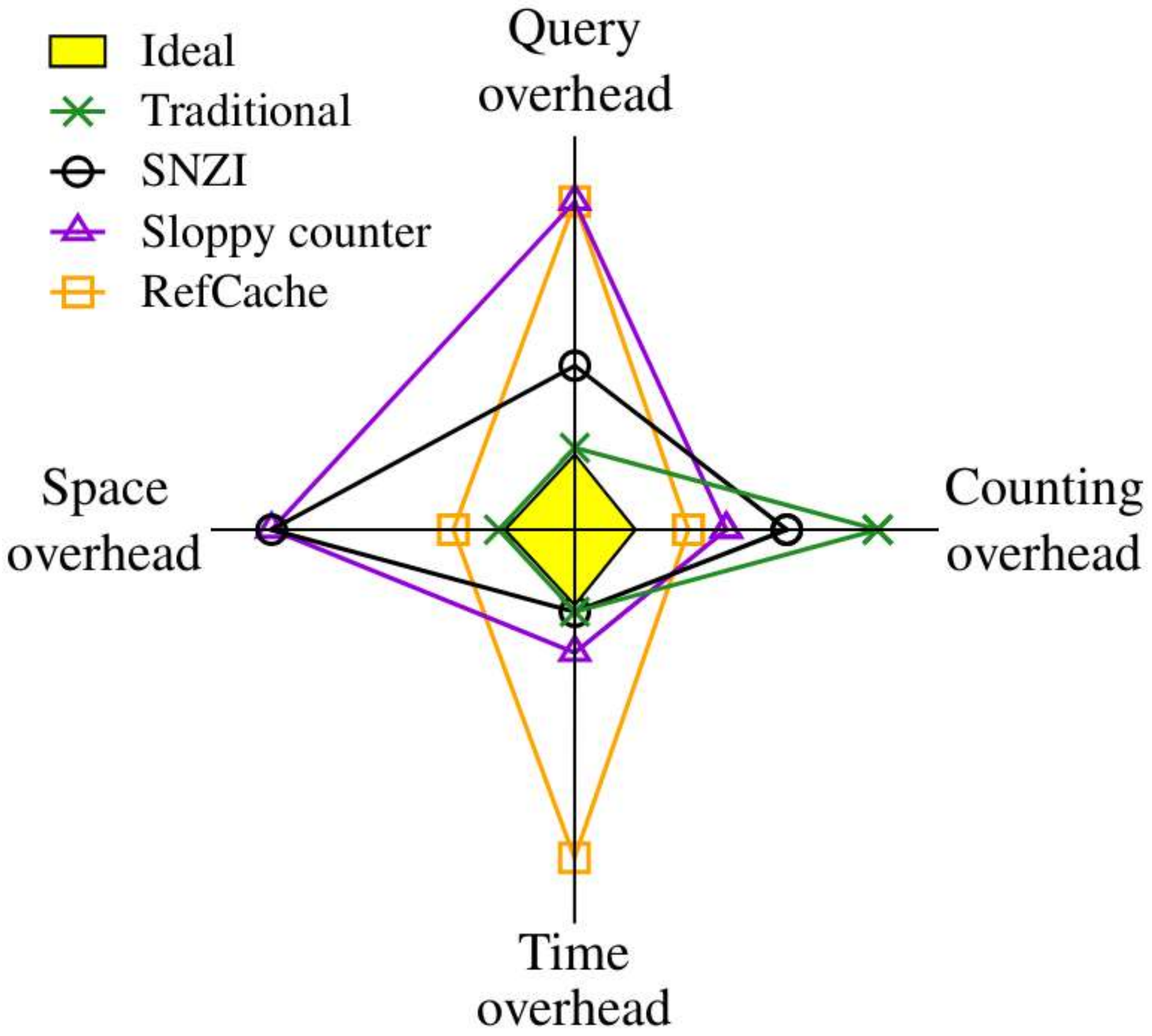


Overhead analysis of prior proposals (cont.)

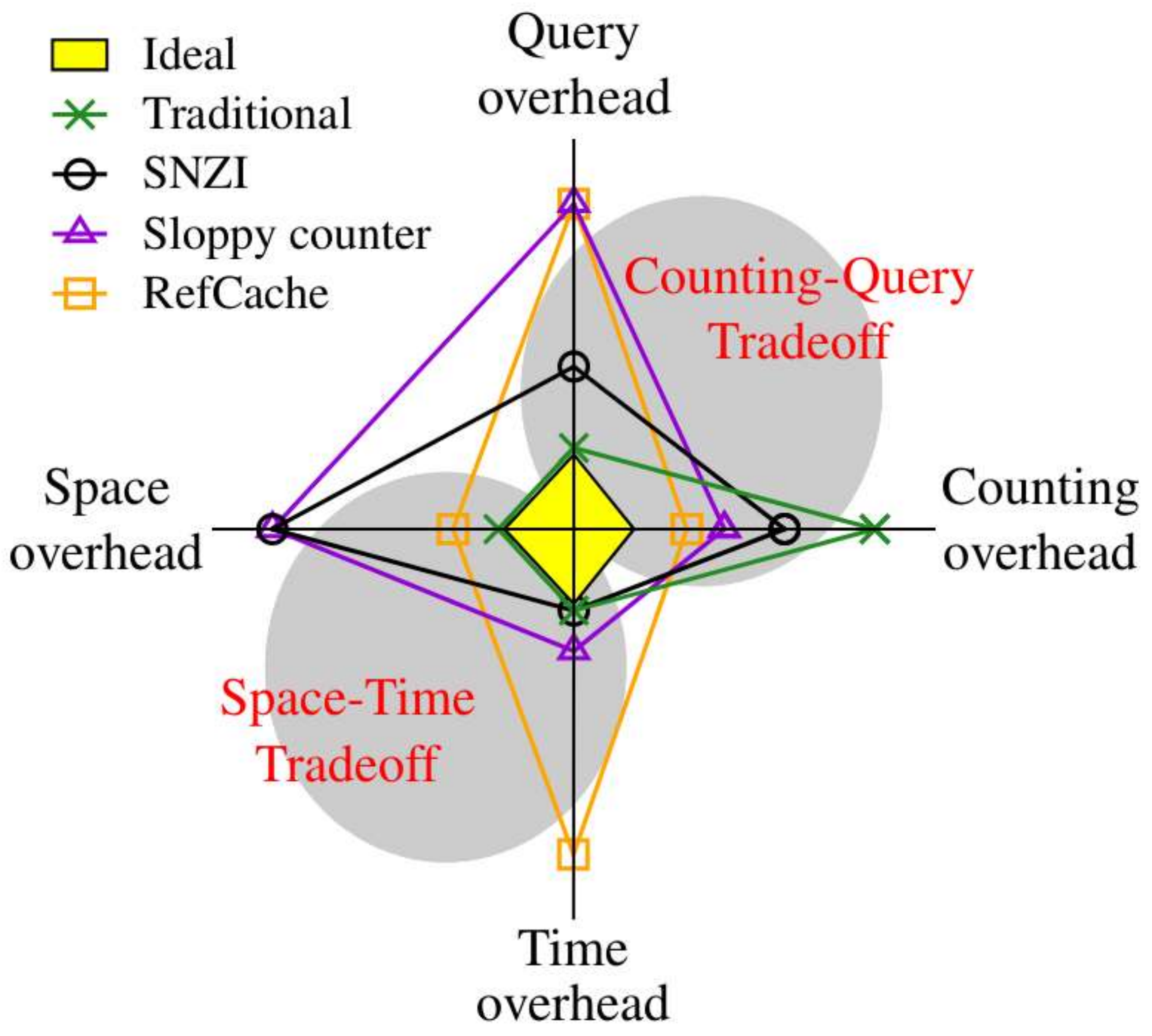




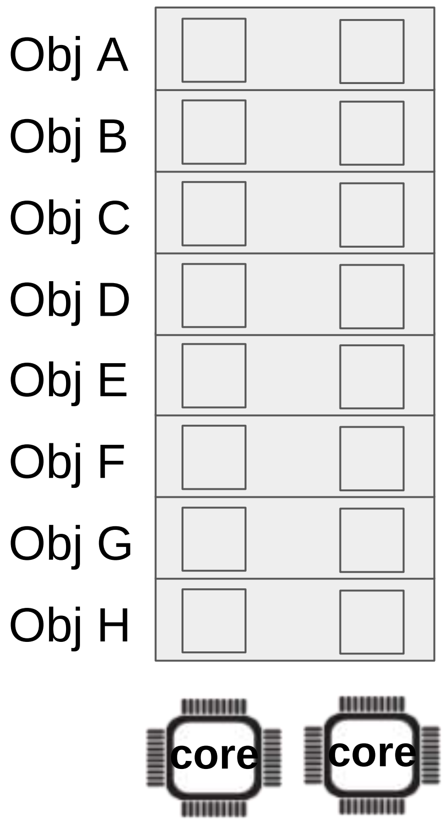
Summarizing all these ...



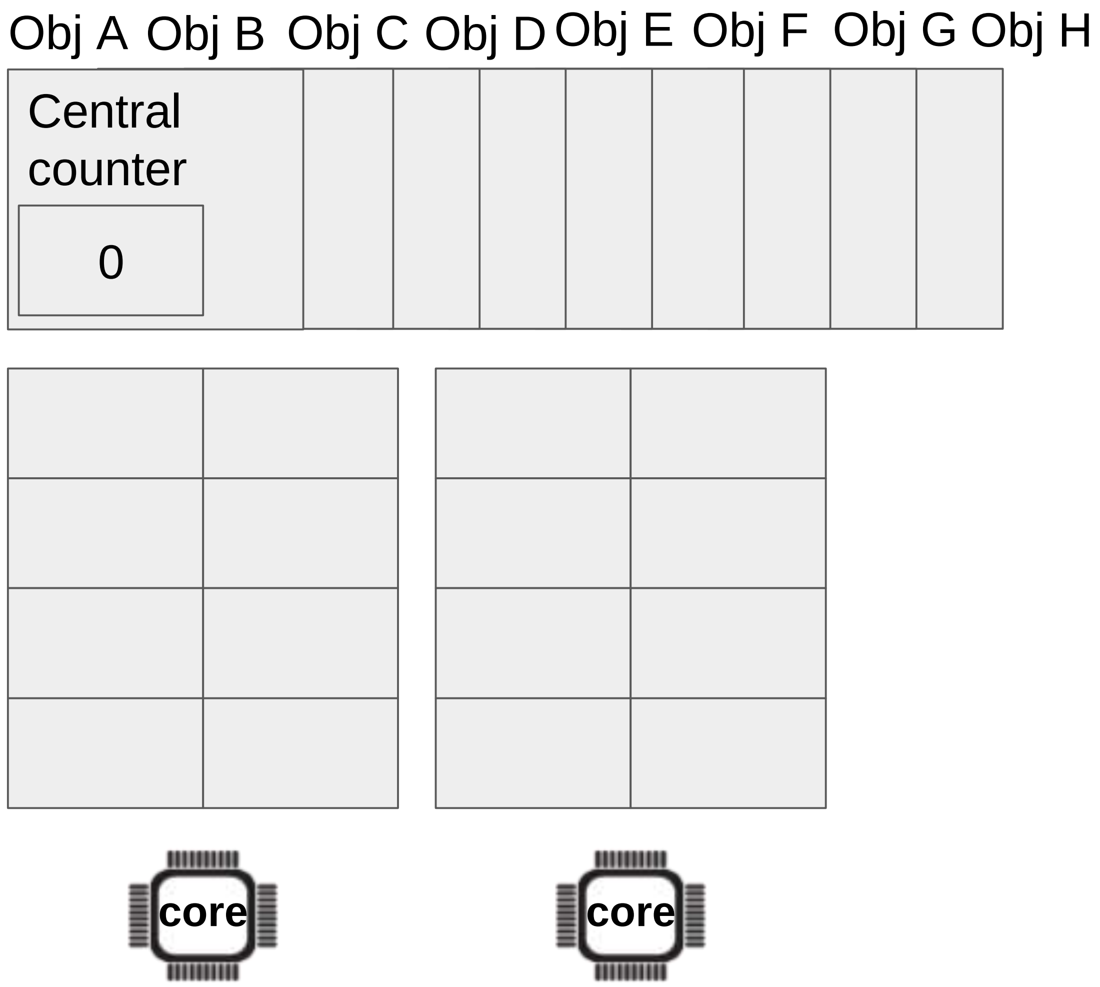
Summarizing all these ...



Challenges for the space-time tradeoff

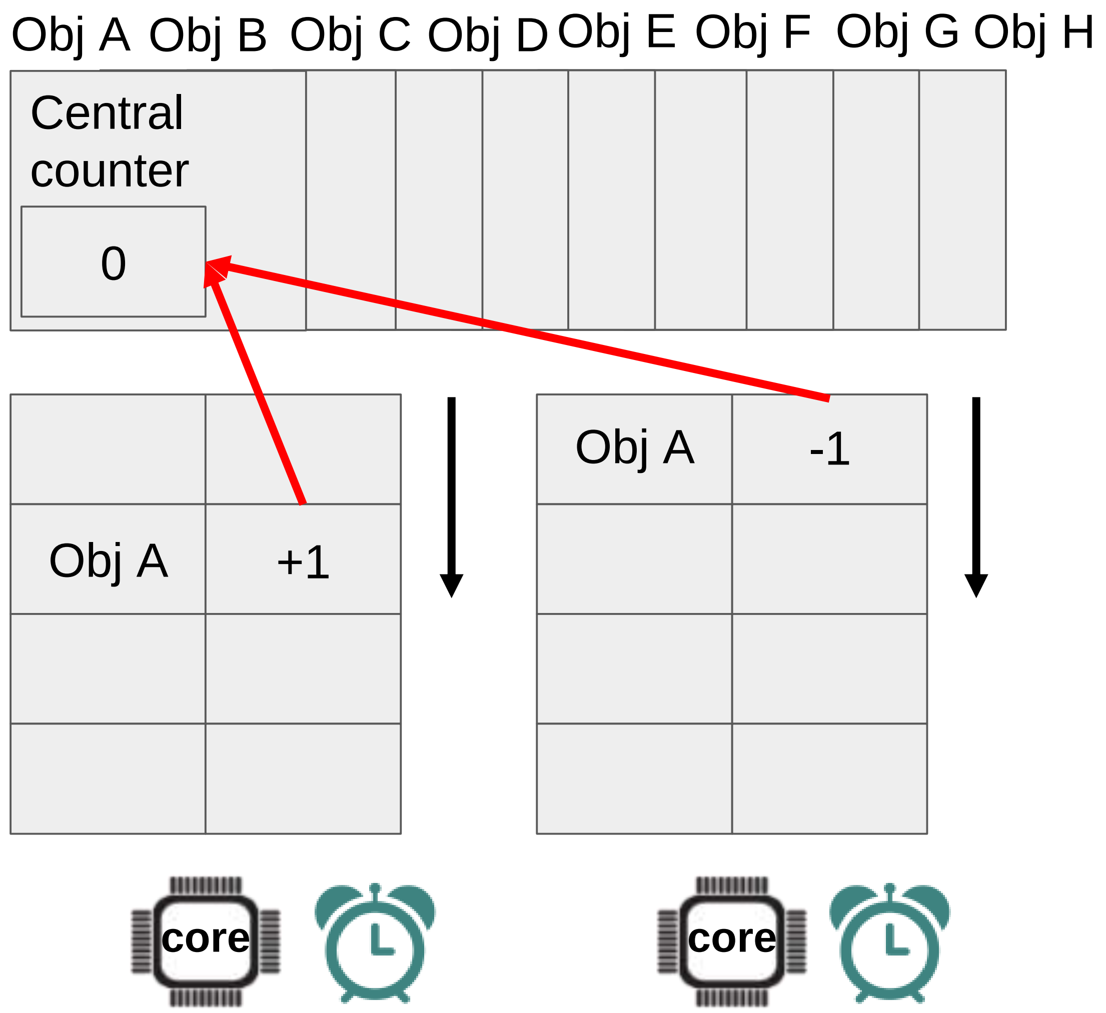


Cache Affinity



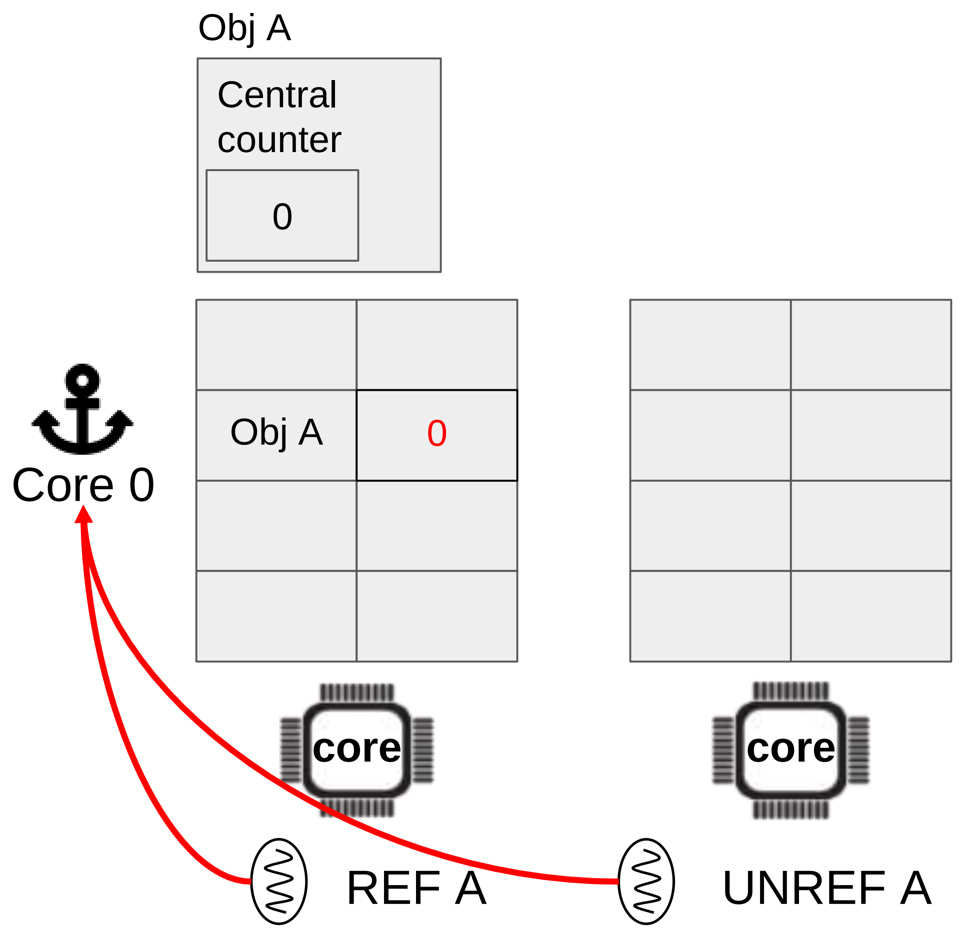
Per-core Hash

Challenges for the space-time tradeoff (cont.)



Our solution to this issue ...

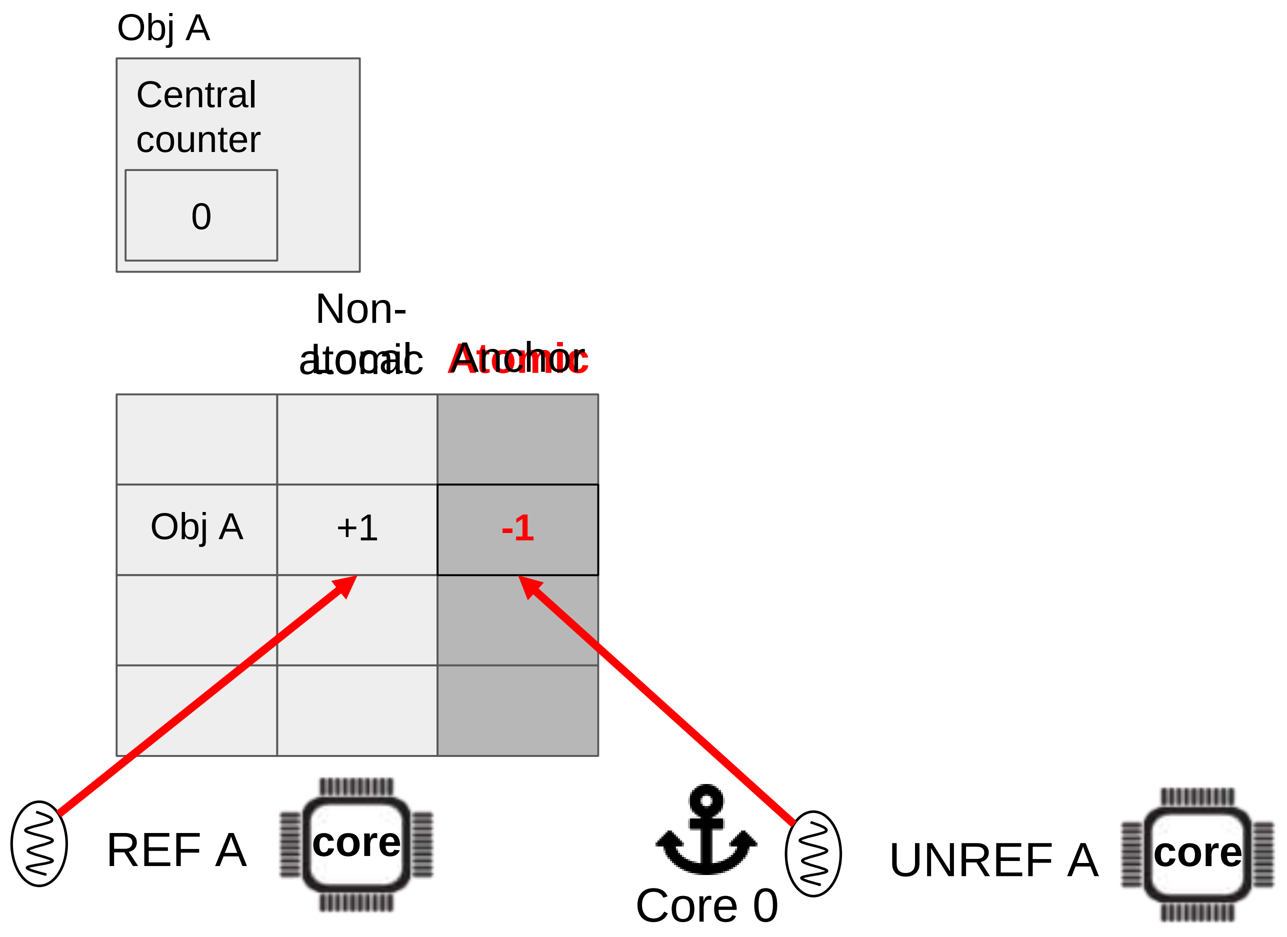
Anchoring



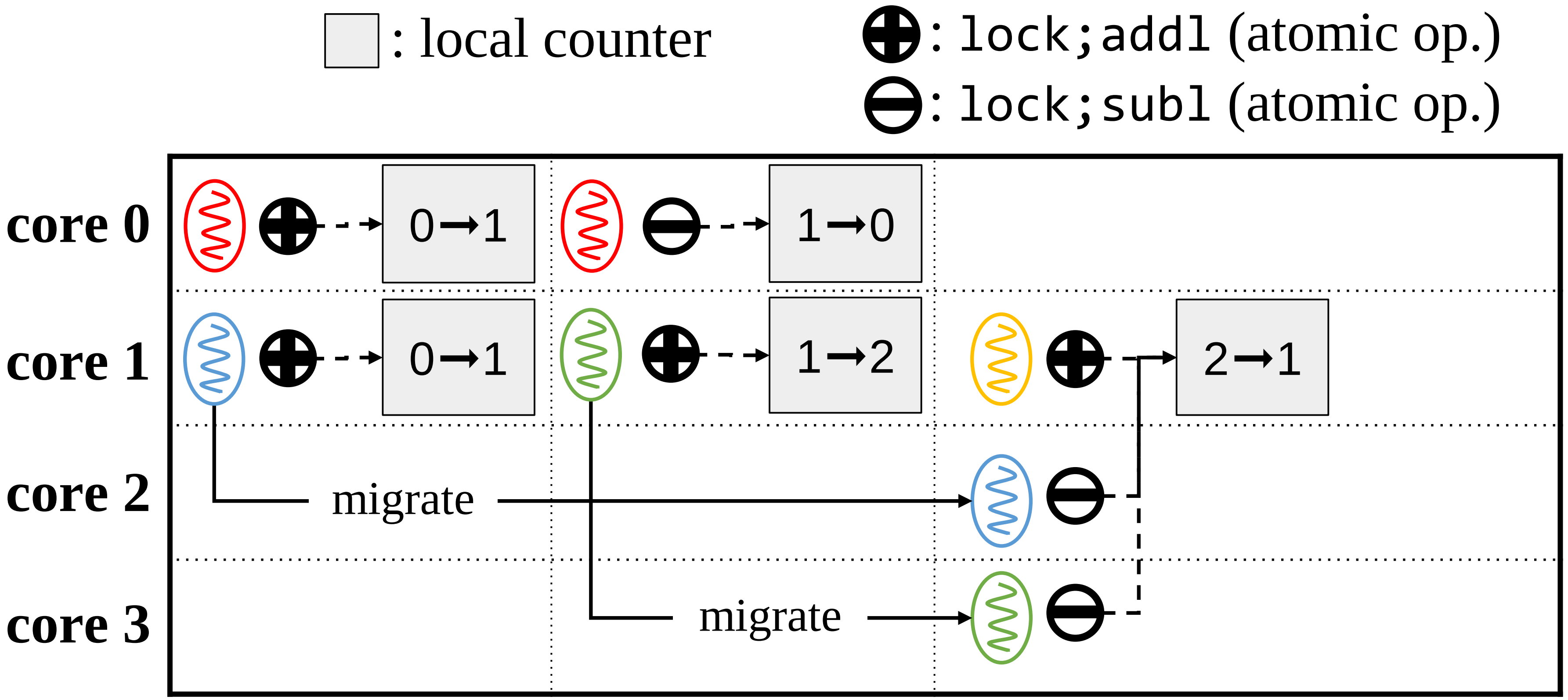
Our solution to this issue ...

PayGo

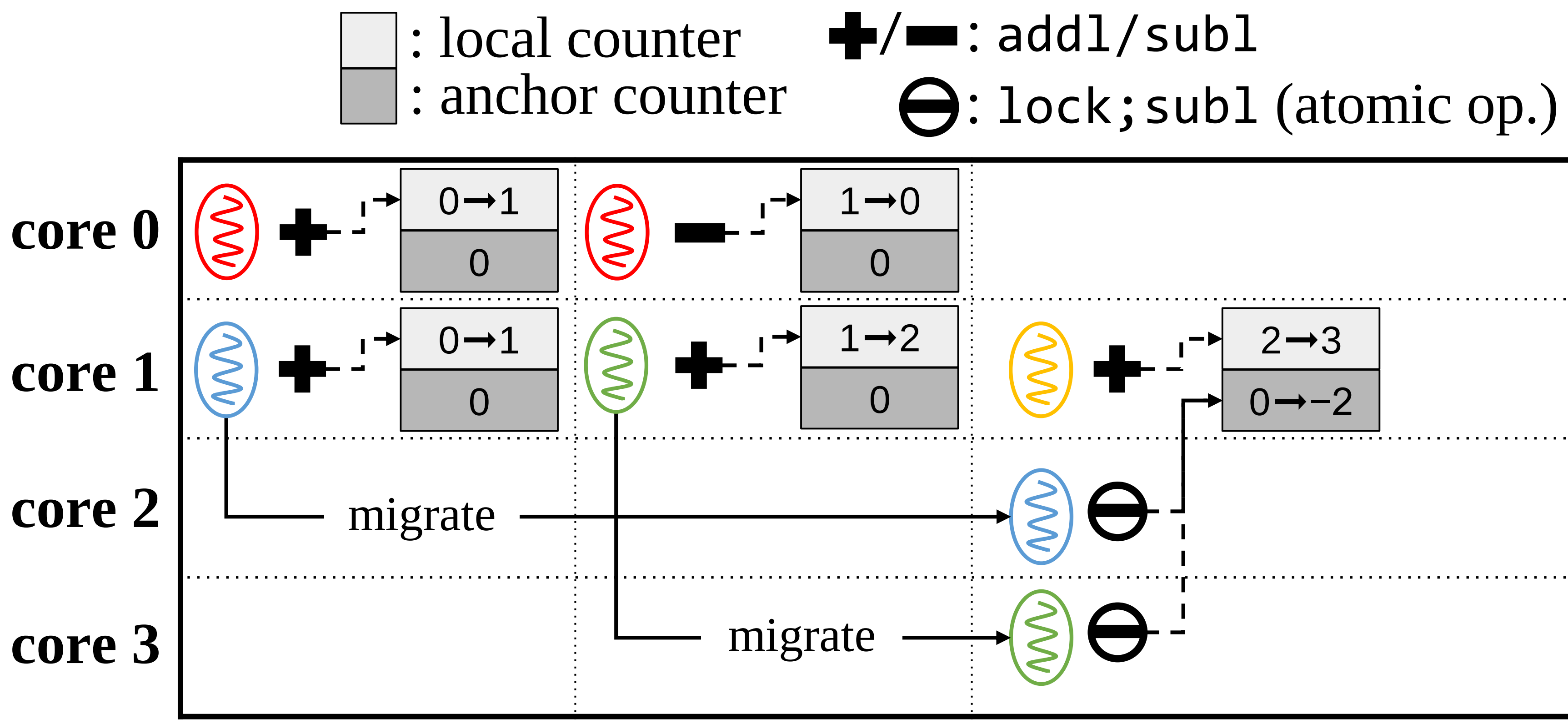
Pay Migration Tax to Homeland



Issue for a single local counter



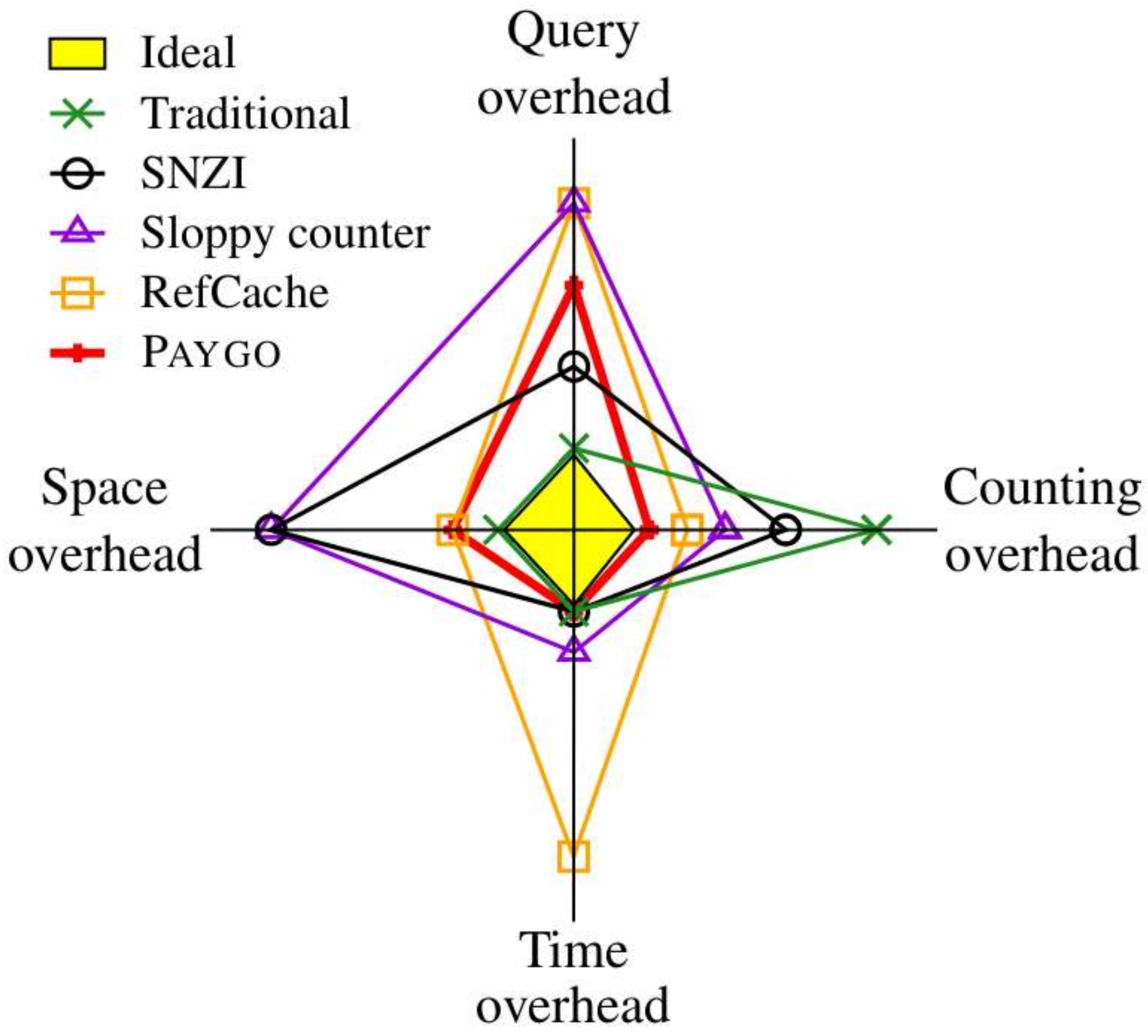
Anchoring in action



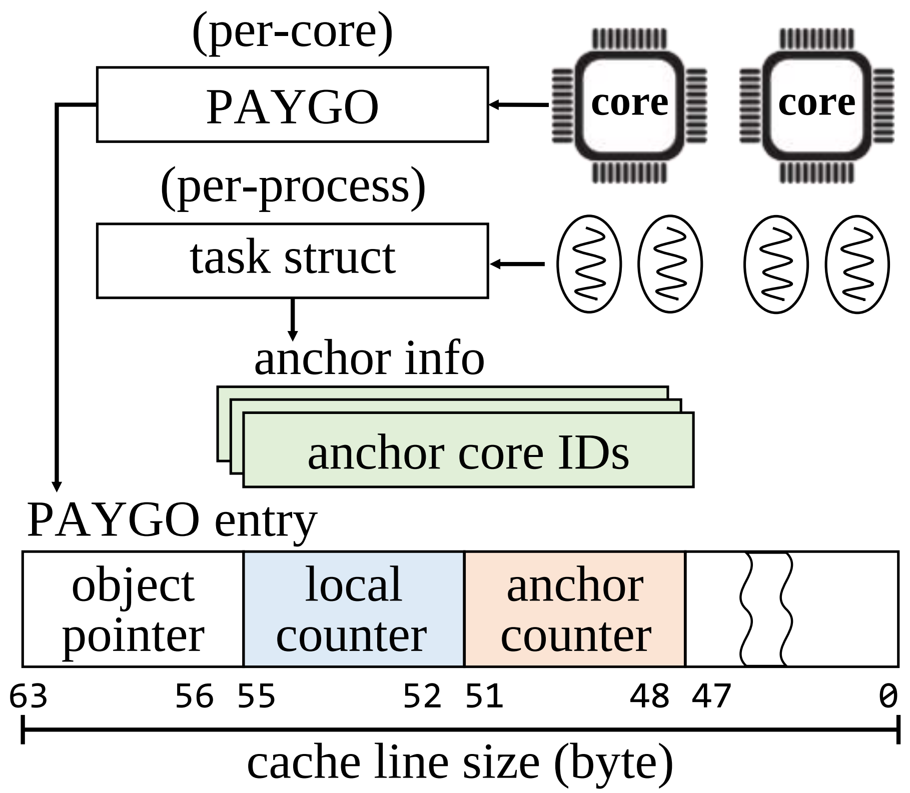
Paygo
(Pay migration tax as you **go** to other core)

- Low counting overhead
  - Scalable for local counters
- Low space overhead (per core hash)
  - Proportional to the number of CPU cores
- Query overhead is still high
  - Escaping the counting-query tradeoff is beyond the scope of this work.

Overhead Analysis for a Reference Counter

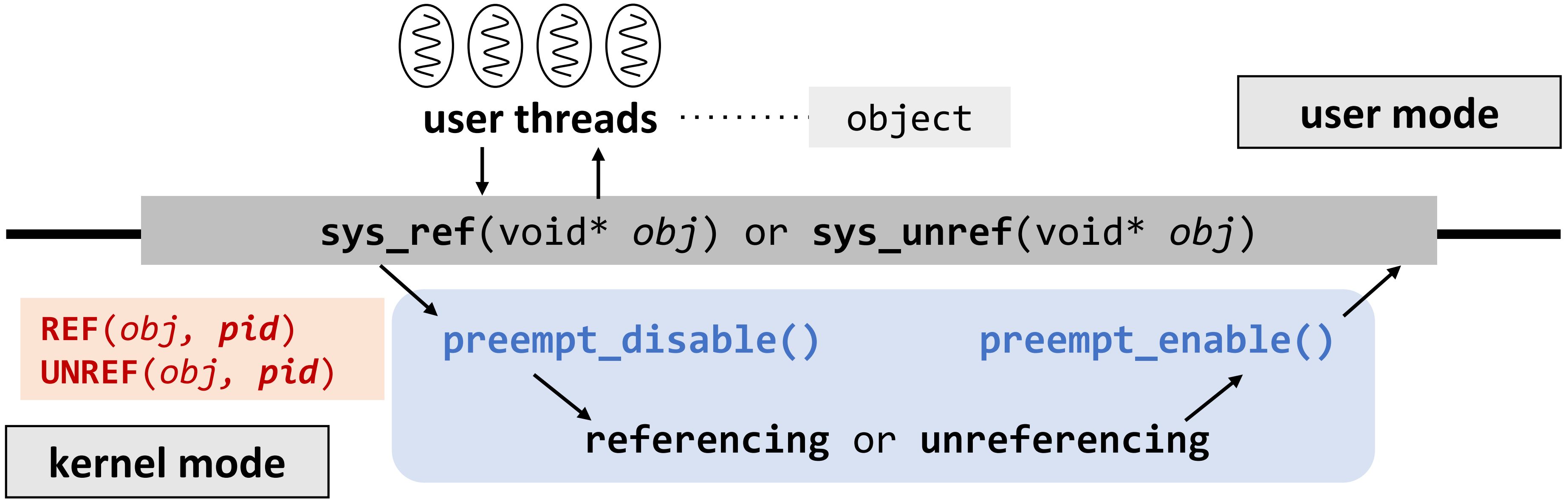


Data structures in PayGo



- Local counter increases the local count by REF operation
- A process records core IDs along with object pointer when REF operation is performed

Extending PayGo design to support user-level objects



Experimental Setup

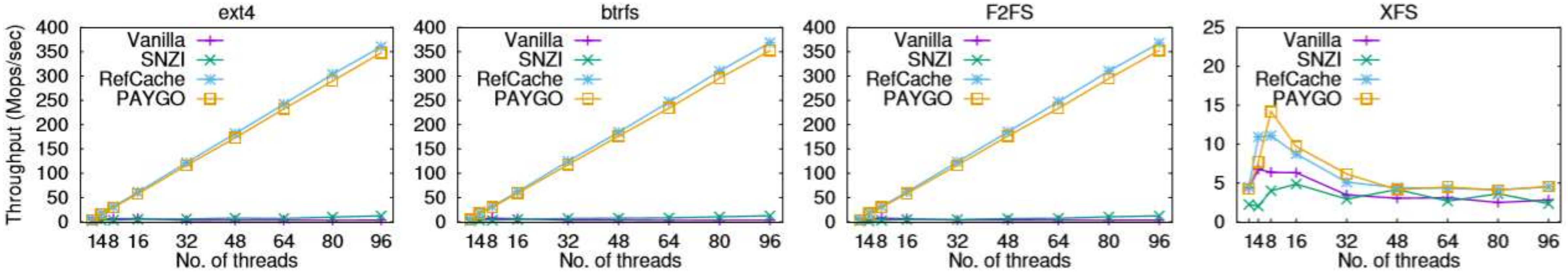
Kernel: Linux 4.12.5

CPU: four 24-core Intel Xeon E7-8890 v4 CPUs

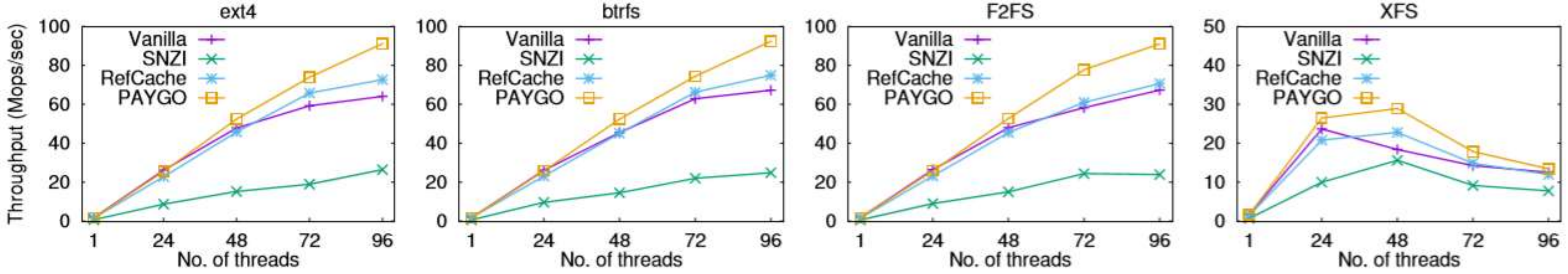
RAM: 1 TiB DDR4 DRAM

Storage: Samsung SM1725 NVMe SSD

Scalability Comparison of the Linux Page Cache

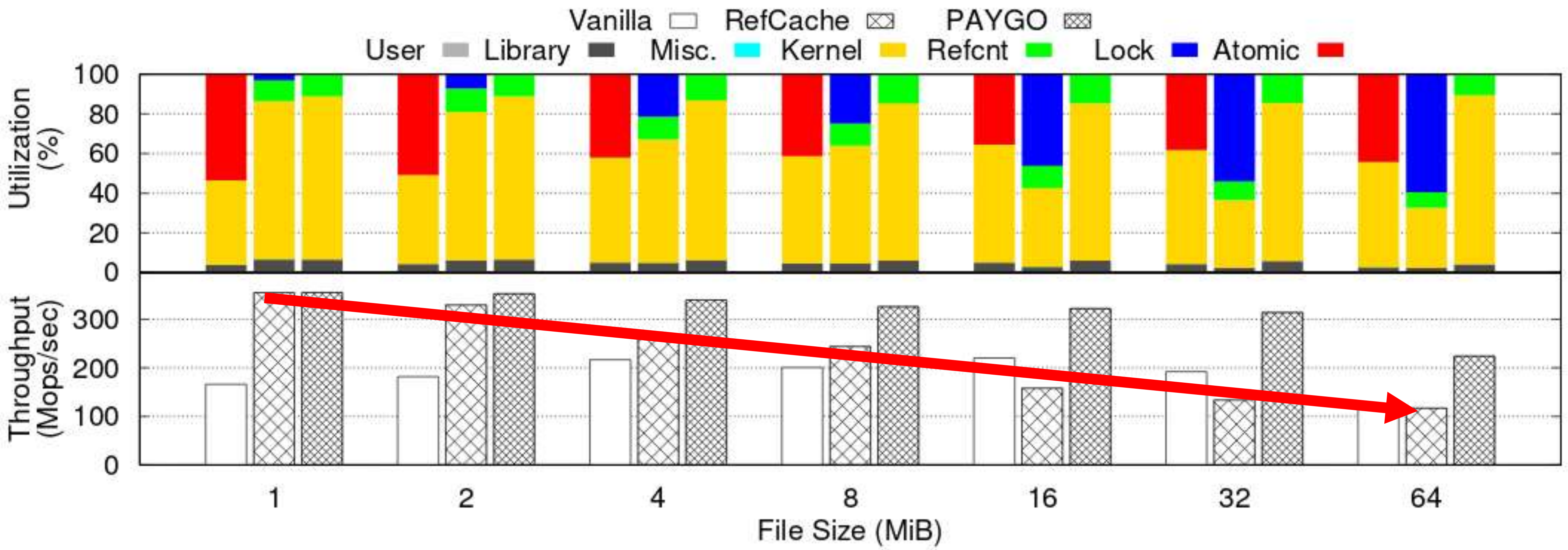


Strongly contending workloads: FxMark DRBH workload

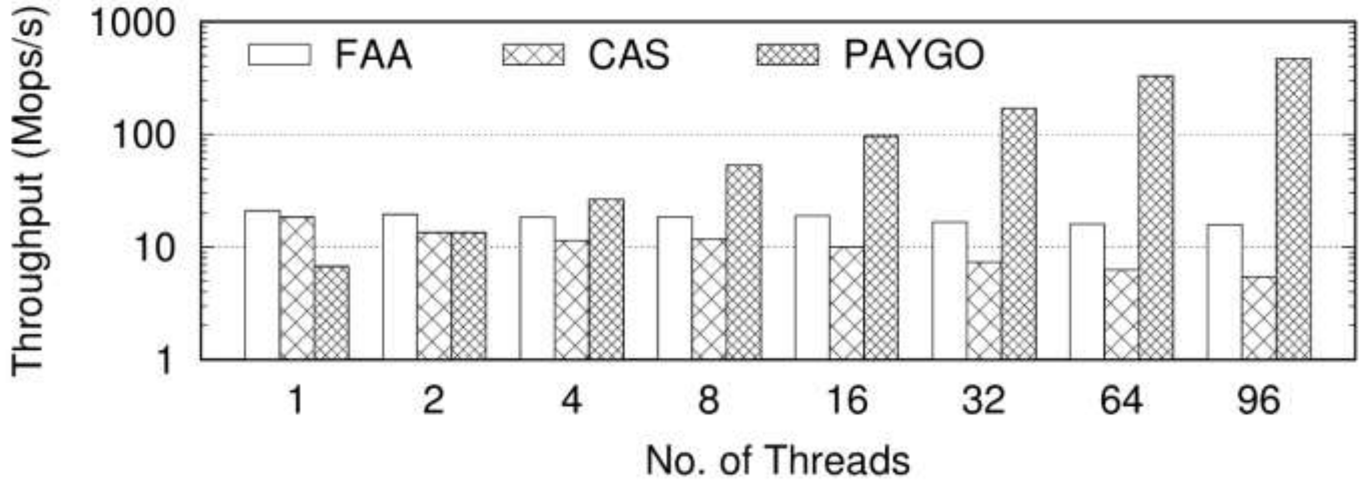


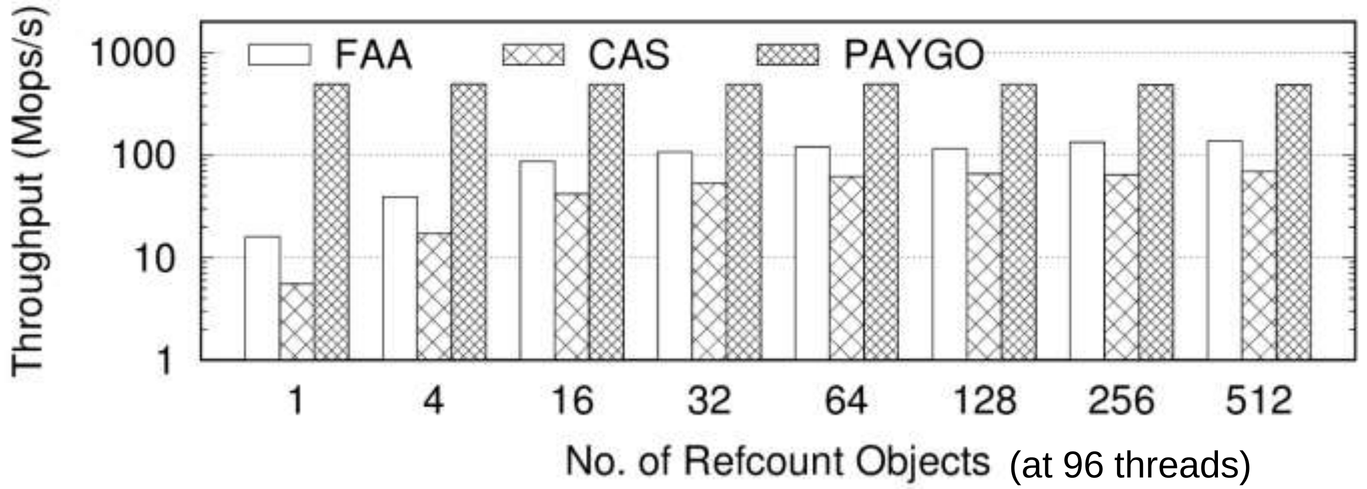
Weakly contending workloads: filebench modified fileserver workload

Performance Spectrum on Varying Contention Levels



Scalability of User-level Paygo





(at 96 threads)

Conclusion

- Designing scalable reference counting techniques should consider space-time tradeoff as well as counting-query tradeoff.
- PayGo escapes the space-time tradeoff by using anchoring technique.
- PayGo provides scalable counting and space efficiency with negligible time delay for reclaiming obsolete hash entries.