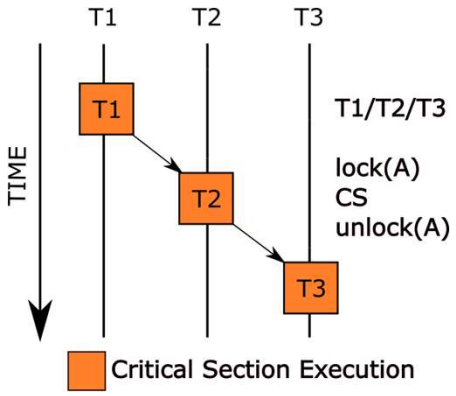
Komb: Transparent Combining for Kernel

EPFL RS3LAB

Vishal Gupta Kartikeya Kumar Dwivedi Sanidhya Kashyap

Existing lock design unable to provide optimal performance

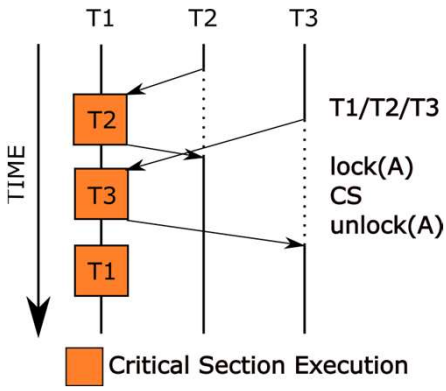
Traditional Design: Move Data to Computation



Existing Locks :

- Reduce contention of lock cache line.
- Require movement of shared data cache line

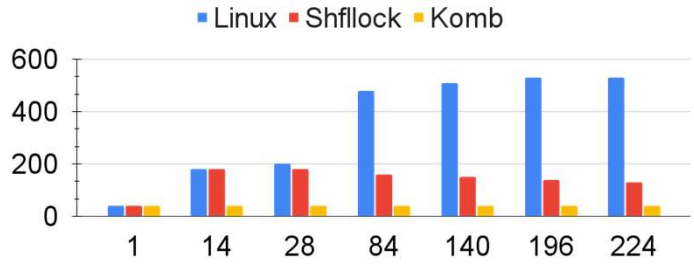
Alternative Design: Move Computation to Data



Existing Locks :

- Reduce contention of lock and shared data cache line.
- Require application rewrite

Critical Section (CS) Latency



Traditional design unable to reduce CS latency

Kernel Locks

- Fine-grained locking :
  - Level locking
  - Out-of-order (OOO) unlocks
- Blocking vs non-blocking locks
- Different context :
  - IRQ
  - Migration

Hard to automatically identify critical section

Komb: Transparent Combining for Kernel

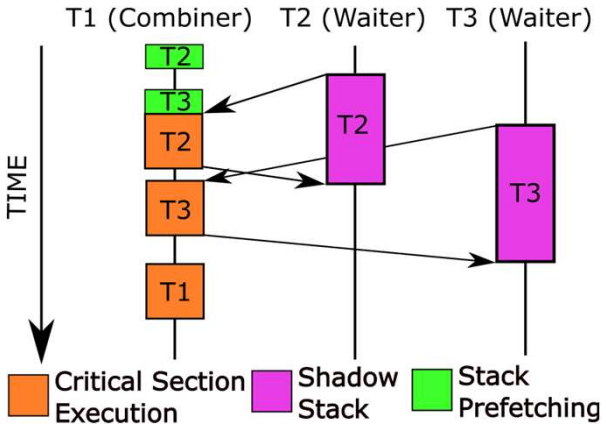
Komb Design

- **Stack Switch**
  - Transfer critical section context using stack.
- **Stack Prefetching**
  - Overlap stack movement with CS execution.

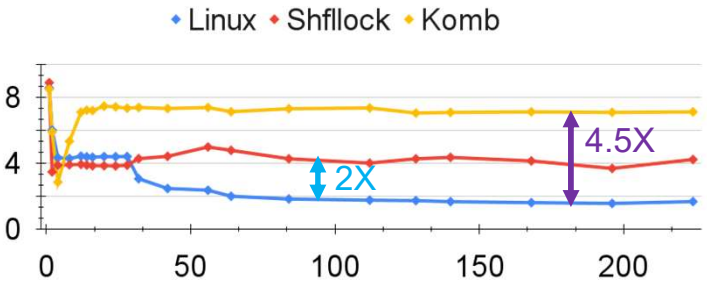
.

- **Shadow Stack enables :**
  - Async interrupt processing on waiter CPU.
  - Async sleeping of waiter thread.
  - Level locking and OOO unlocking.

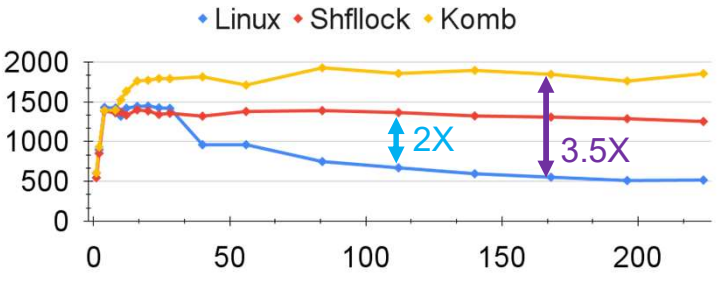
Komb Flow



Micro-benchmark: Hash-Table



Macro-benchmark: FxMark



Conclusion: Reduce shared data movement to scale lock performance