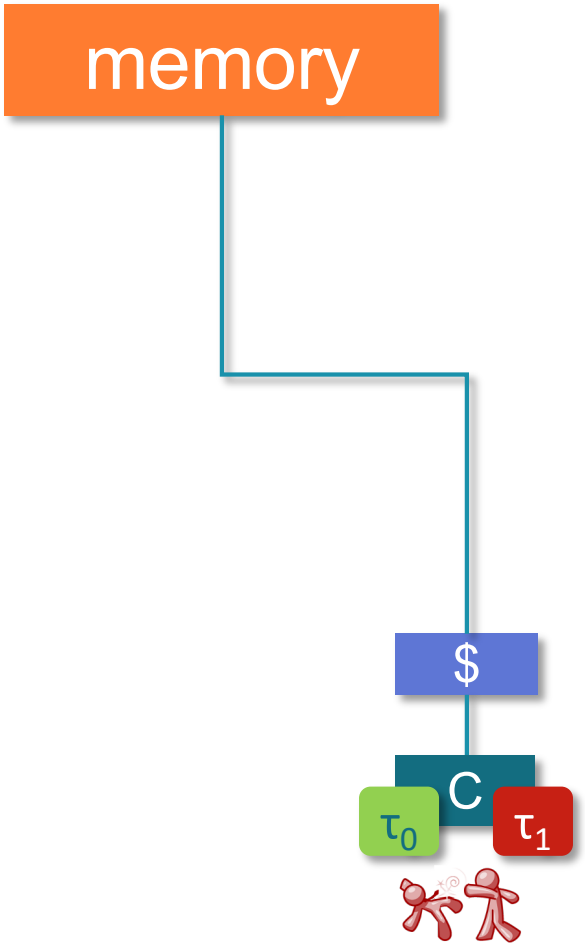
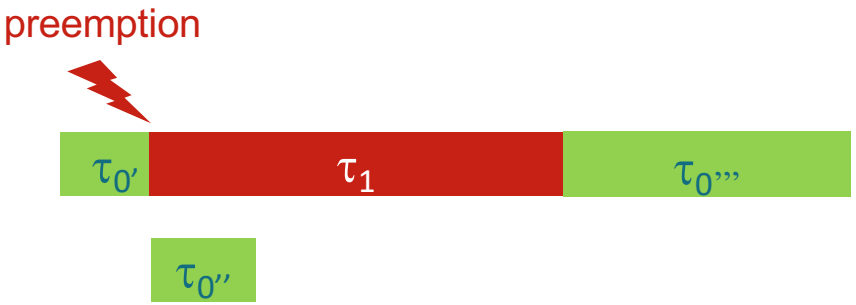
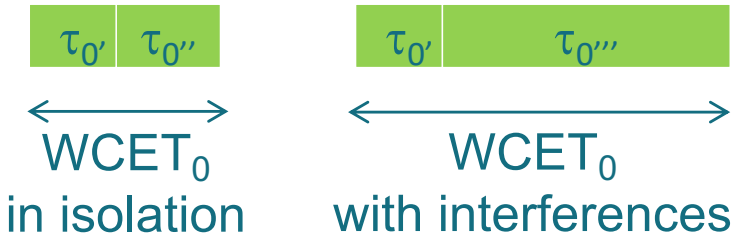
Timing analysis in a concurrent or parallel environment

Tasks usually do not run in isolation…



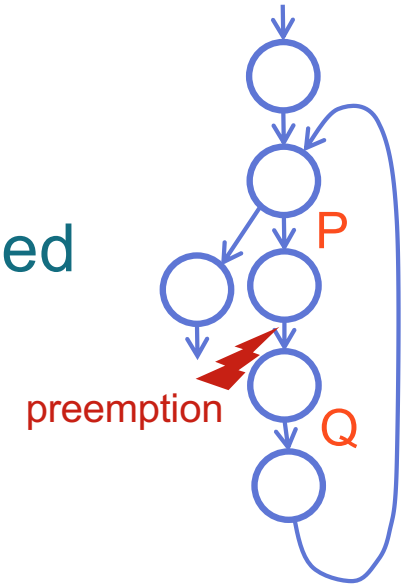




Cache-Related Preemption Delays (CRPDs)

Useful cache block (UCB)

- may be cached at P
- may be reused at Q, a point that can be reached from P, without being evicted on this path

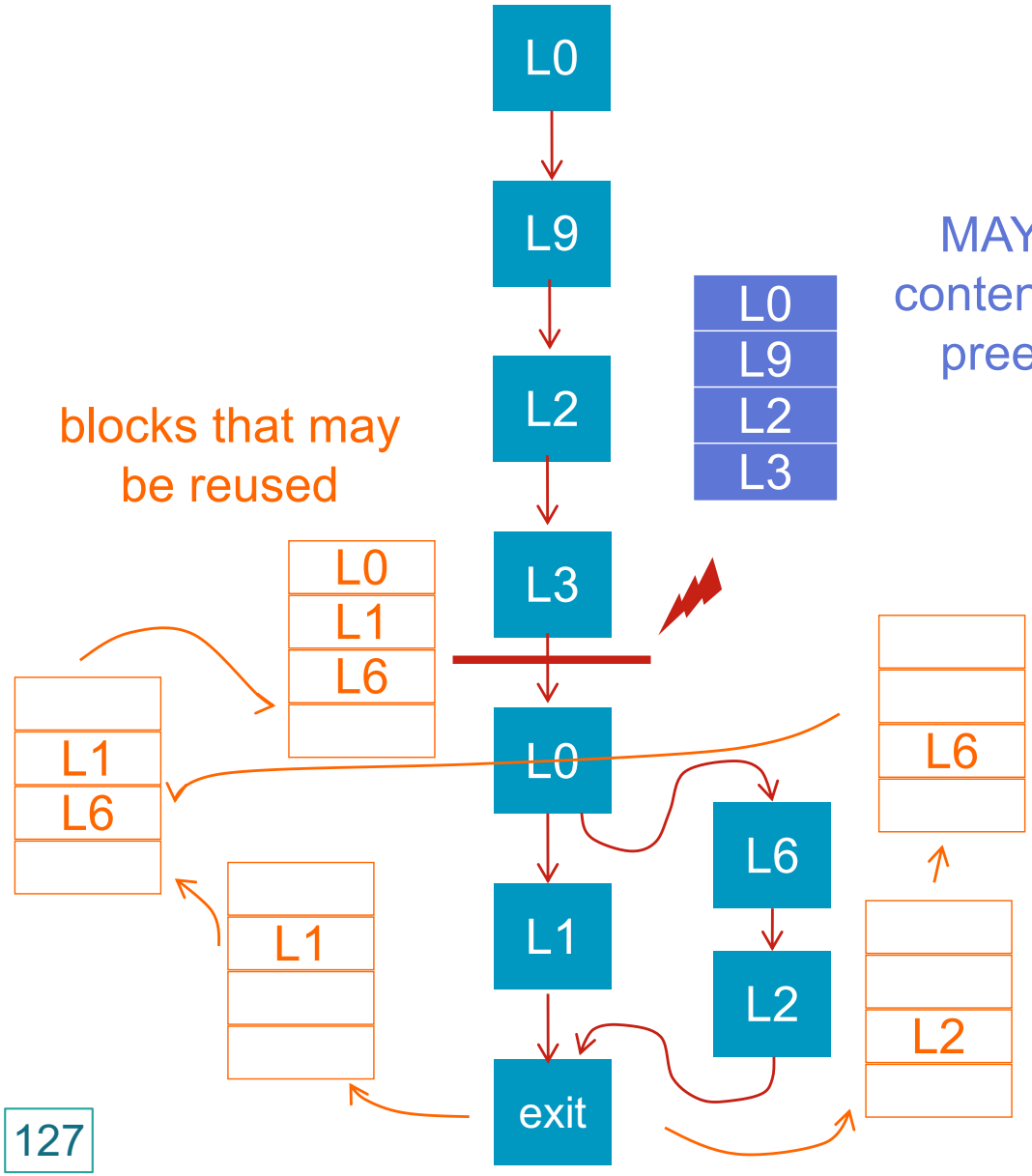


Number of UCBs

≥ number of extra cache misses due to preemption

- for preemption at point P: $CRPD_{UCB}(P) = \sum_{cache\ sets} \min(|UCB(P)|, a) \times t$
- for preemption at any point: $CRPD_{UCB} = \max_P CRPD_{UCB}(P)$

Analysis of Useful Cache Blocks



blocks that may be reused

MAY cache contents before preemption

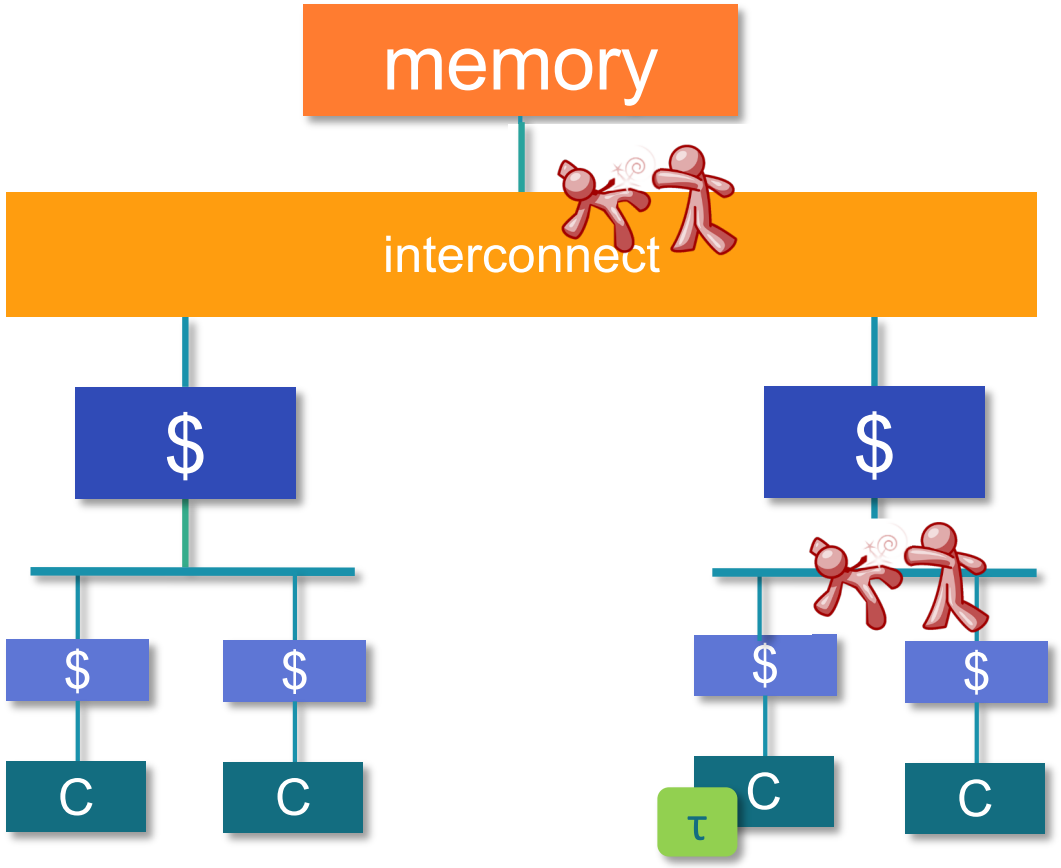
L0
L9
L2
L3

UCBs ?

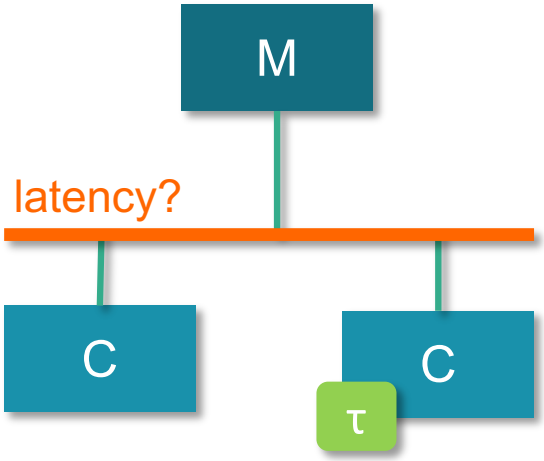
- L0 is is the cache and will be reused
- L1 is not in the cache
- L2 is in the cache but will not be reused

UCBs = {L0}

Tasks usually do not run in isolation…

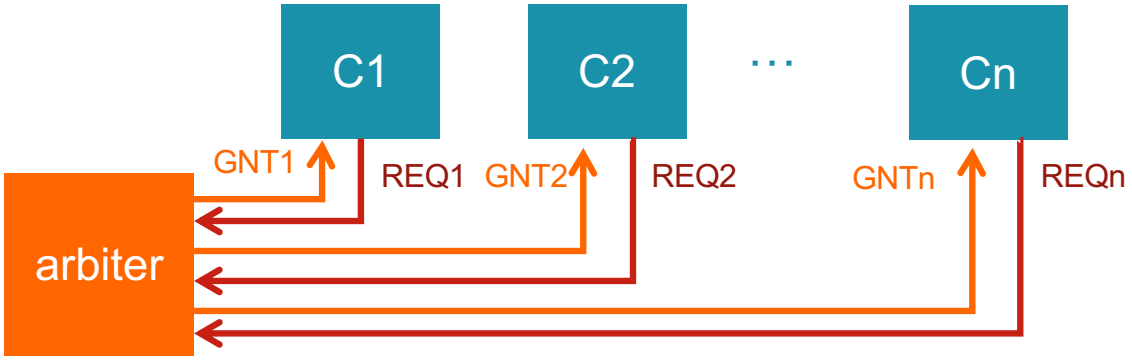


Shared bus



Bus arbitration scheme?

- static
- dynamic



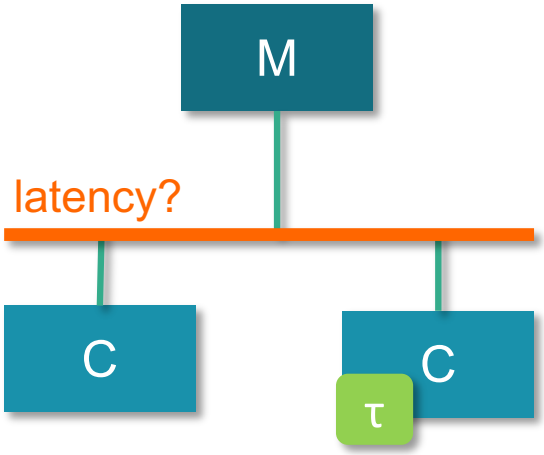

```
ldr r2,[r8,r9]
add r2,r2,#16
ldr r3,[r8,r10]
mul r3,r2,r3
str r3,[r8,r10]
```

latency?

latency?

latency?

Shared bus



Arbitration policy?

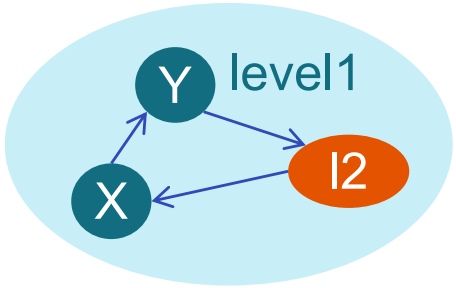
- fixed priority
- round-robin
- fair
- hybrid

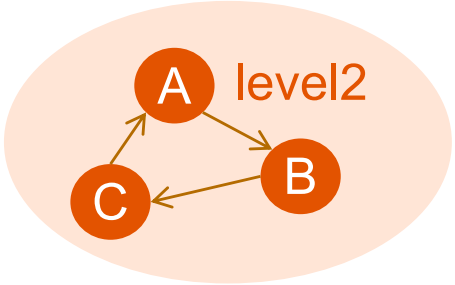
```
ldr r2,[r8,r9]
add r2,r2,#16
ldr r3,[r8,r10]
mul r3,r2,r3
str r3,[r8,r10]
```

latency?

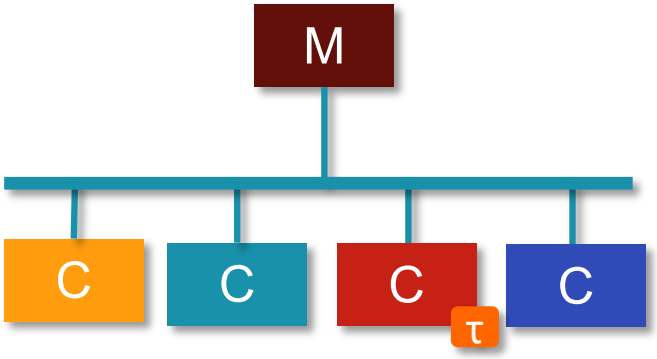
latency?

latency?

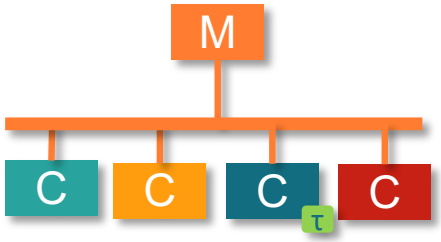




Worst-case latencies on a round-robin bus

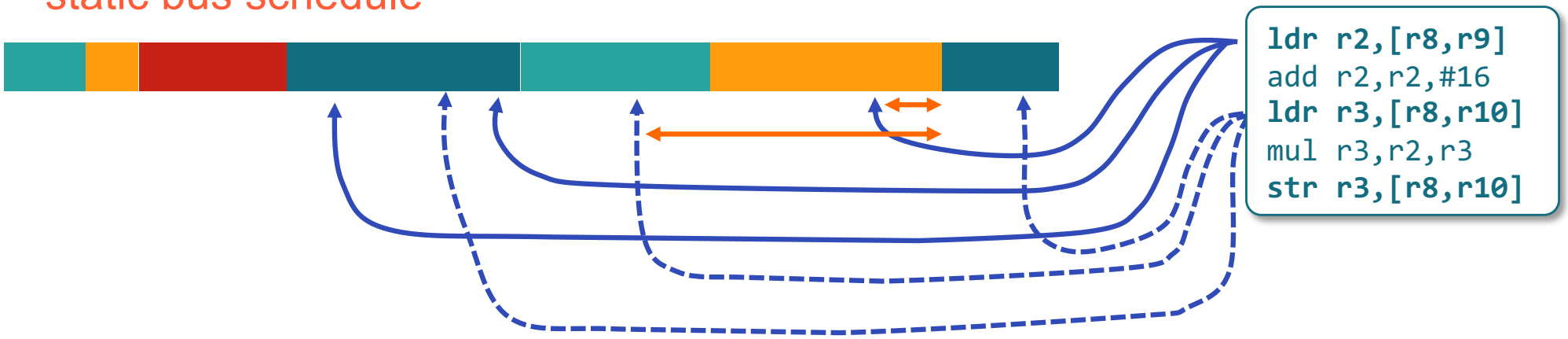


Worst-case latencies on a TDMA bus



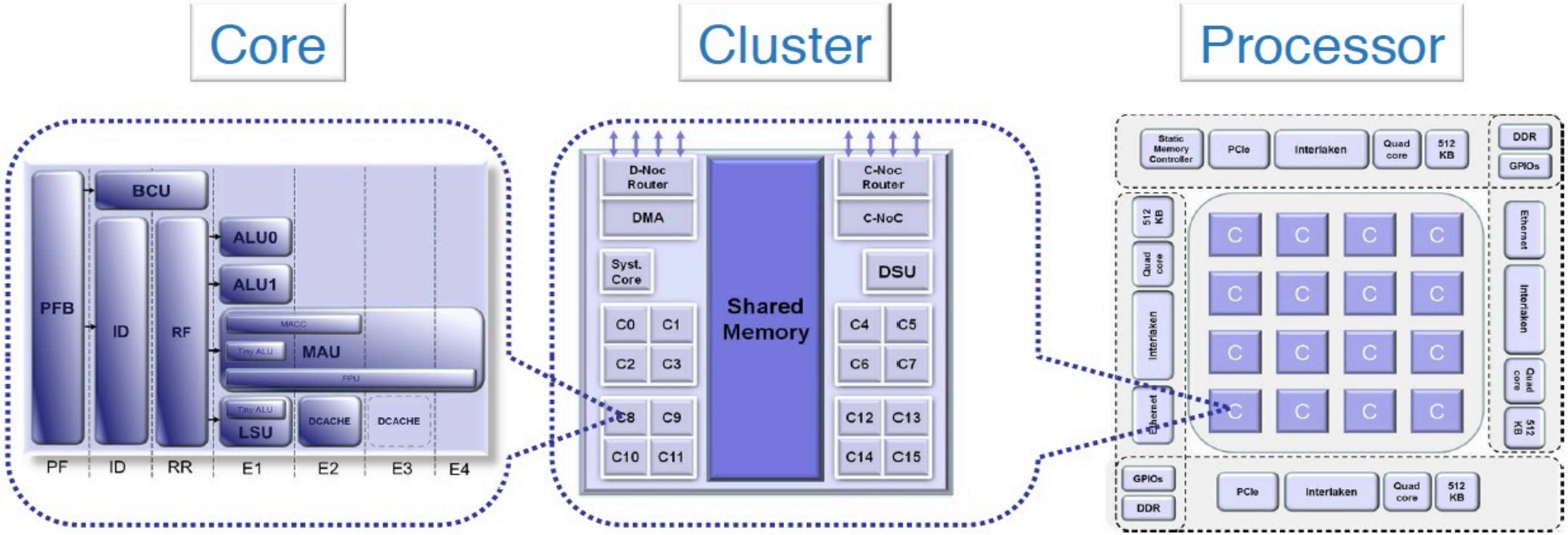
TDMA = Time Division Multiple Access

static bus schedule

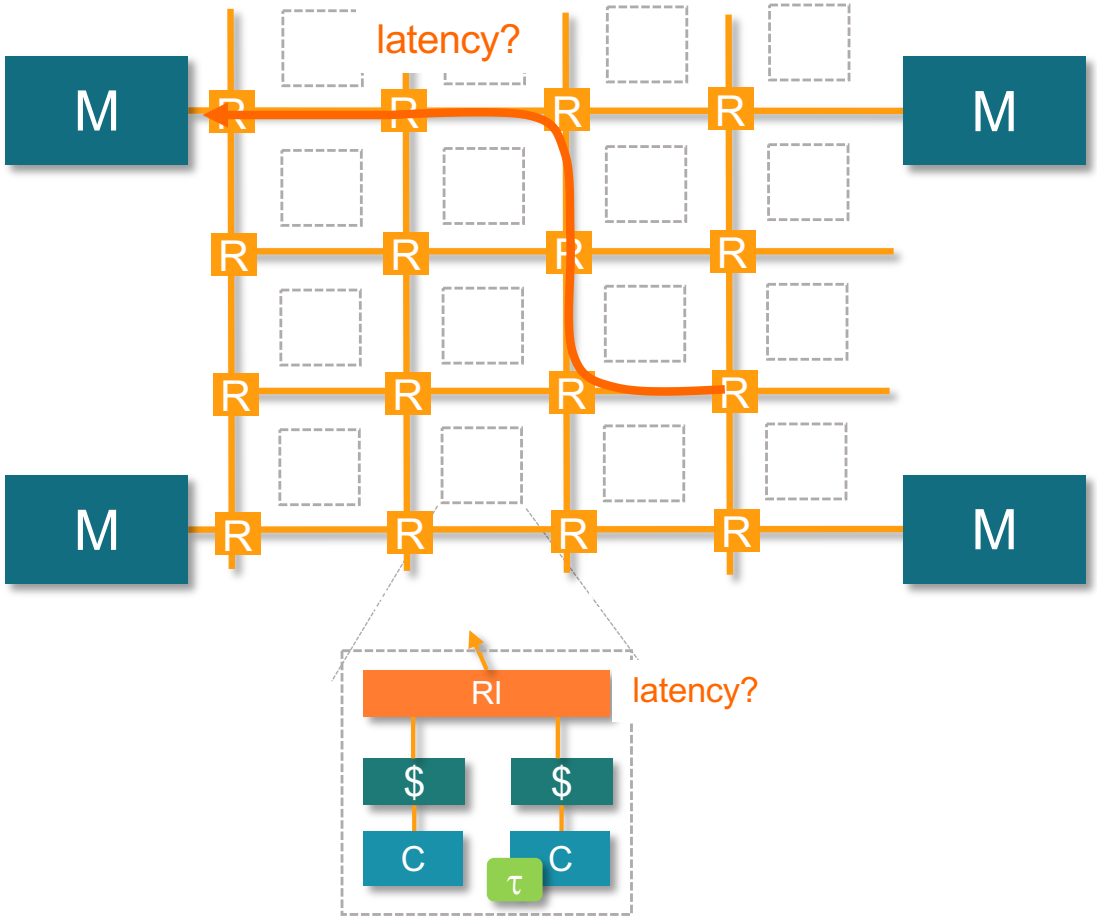


- main challenge: determining the offset of each access to the bus wrt. schedule (periodic)

Kalray MPPA-256



Impact of resource sharing



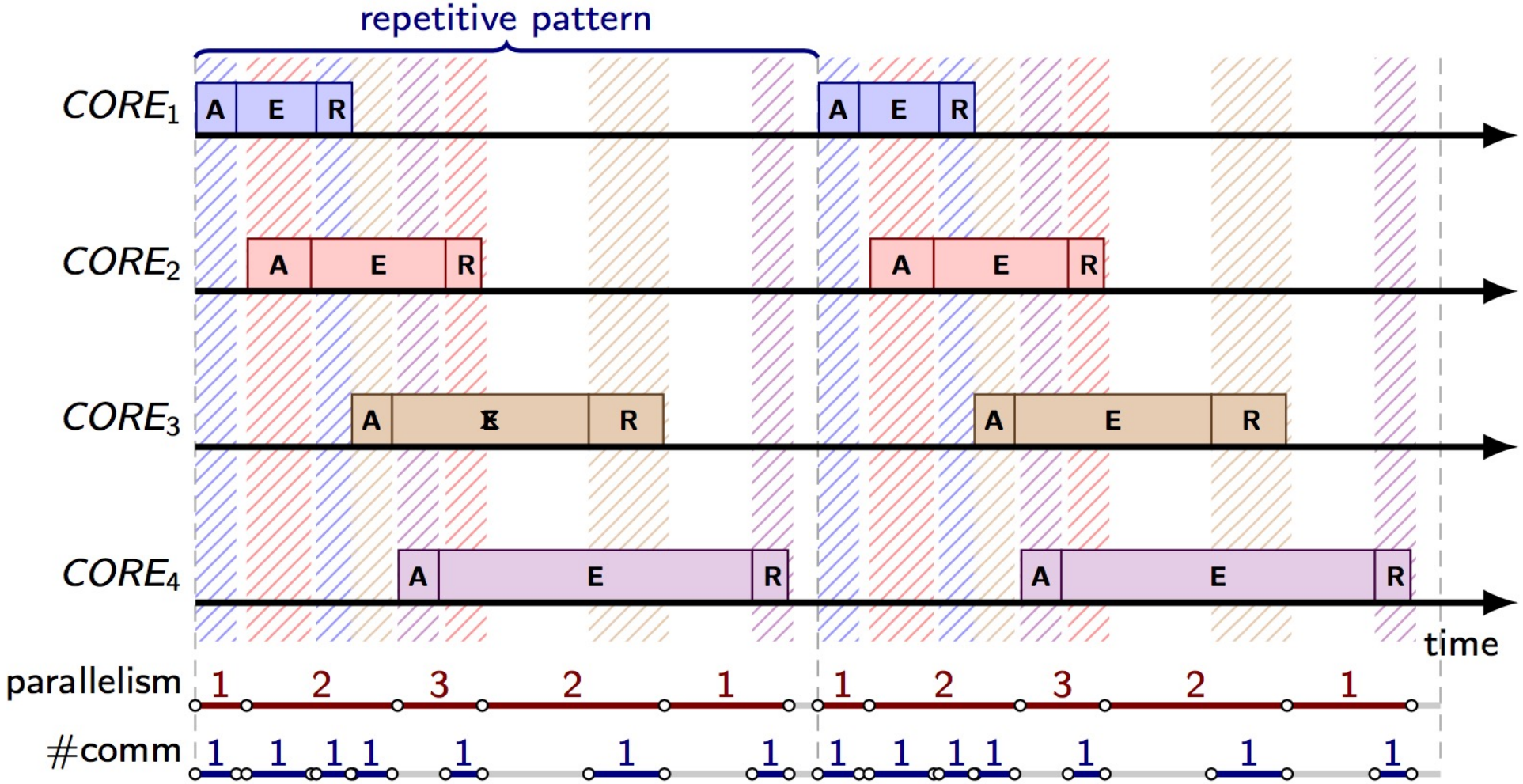

```
ldr r2,[r8,r9]
add r2,r2,#16
ldr r3,[r8,r10]
mul r3,r2,r3
str r3,[r8,r10]
```

latency?

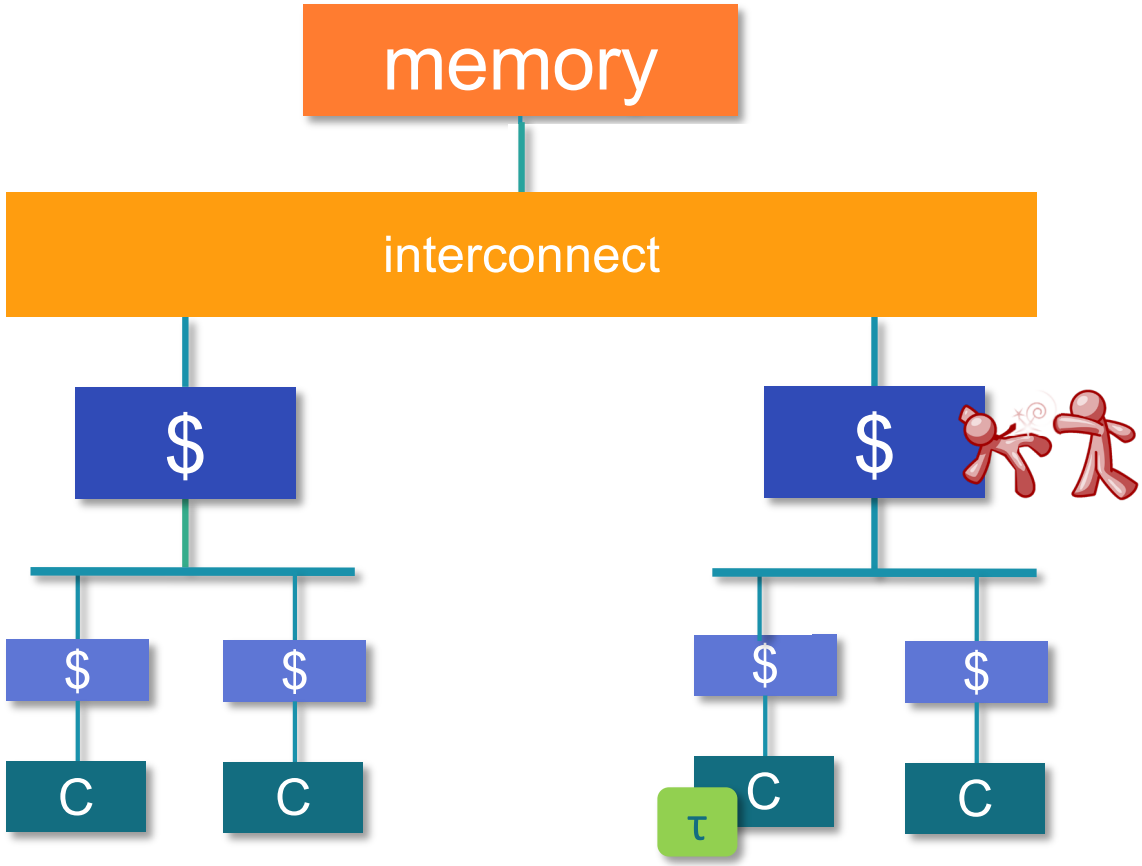
latency?

latency?

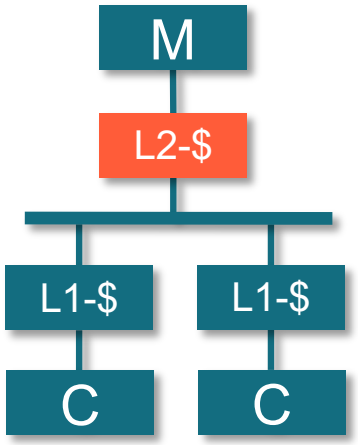
Predictable execution model

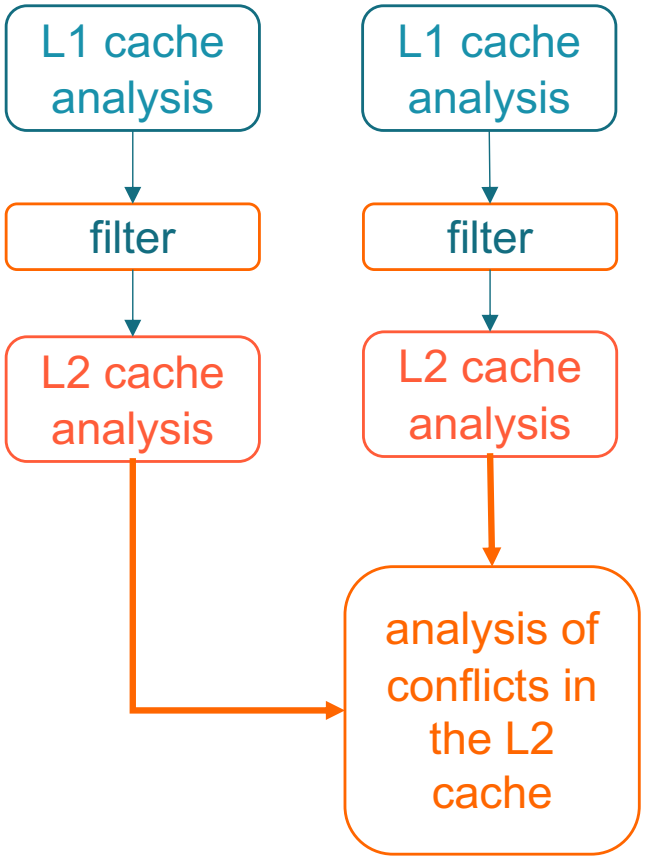


Tasks usually do not run in isolation…



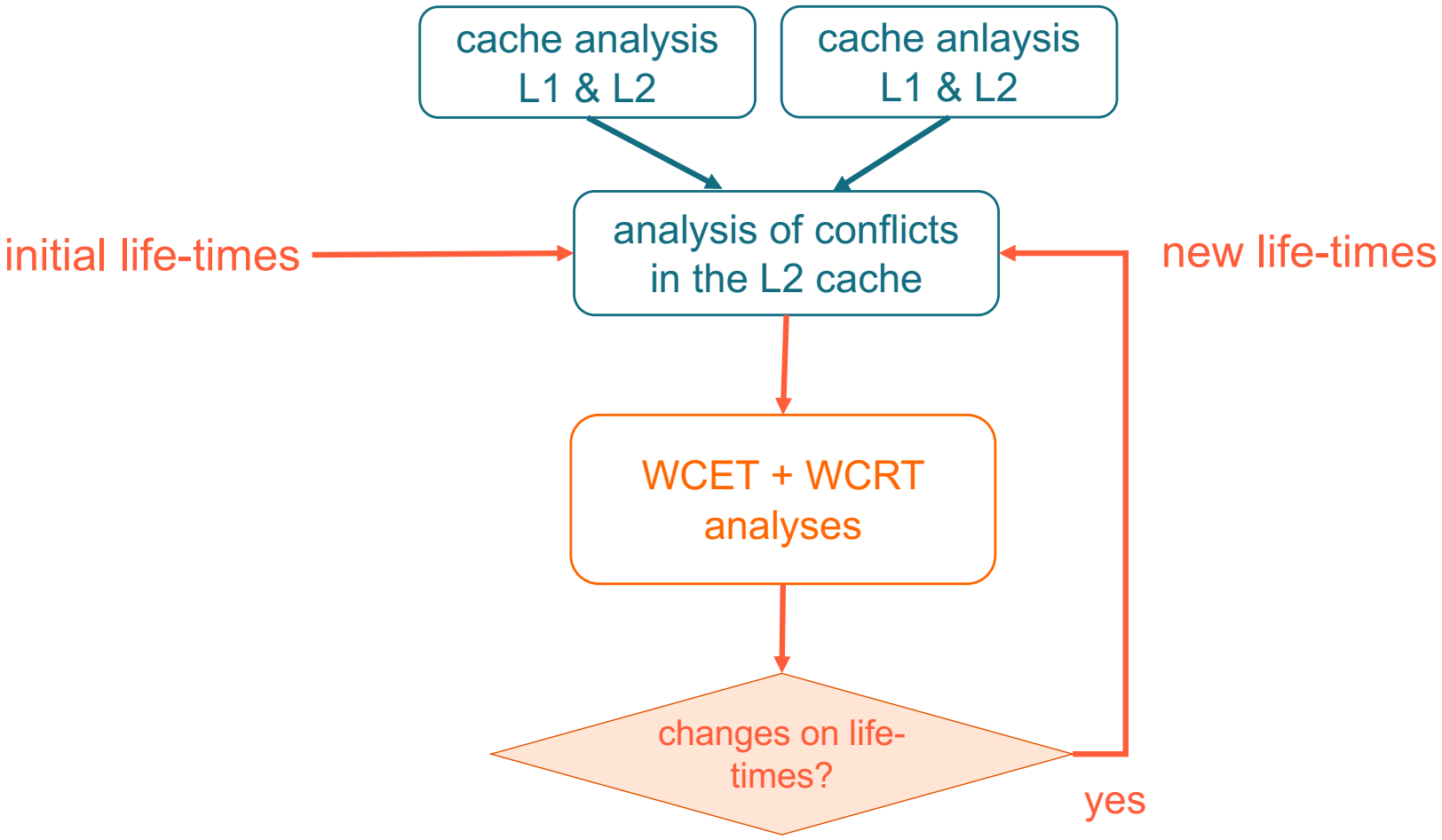
Behaviour of shared caches





every access to set S by a task in T may corrupt S in any abstract cache state computed for task T

Behaviour of shared caches



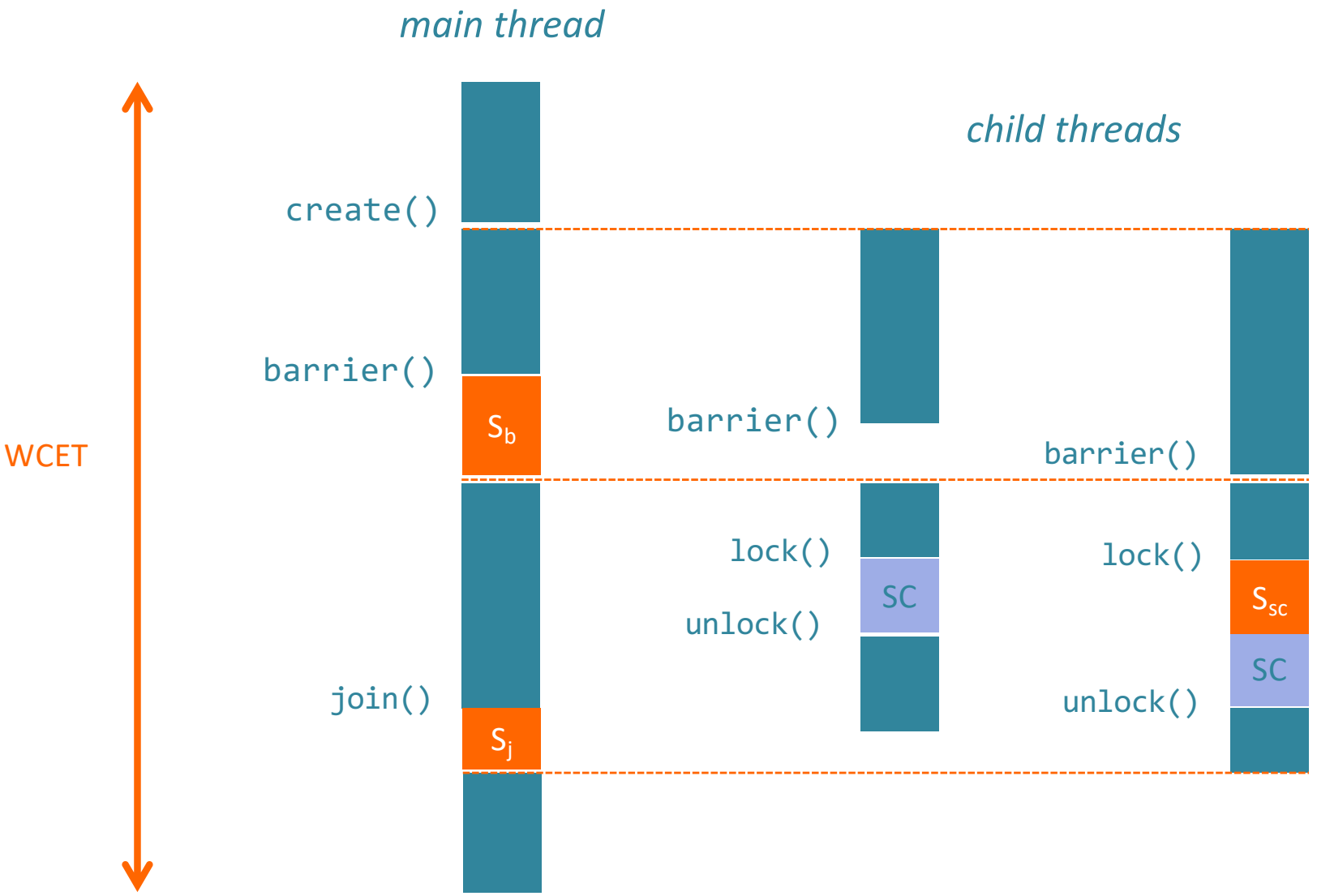
Software-level interferences

WCET analysis of parallel programs (running on multicore architectures)

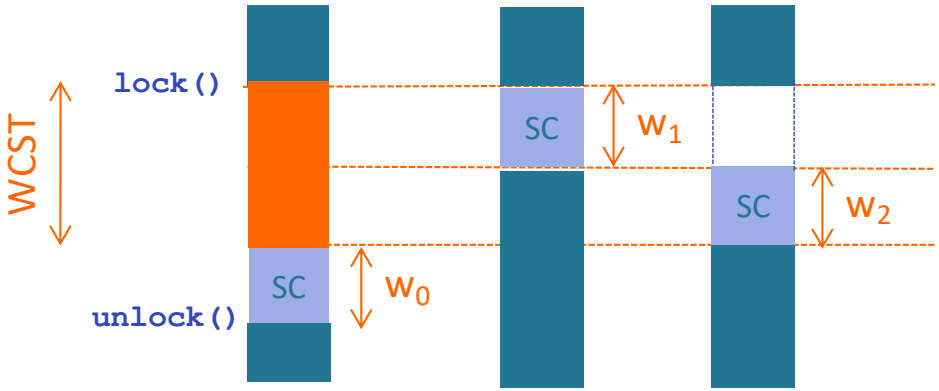
Assumptions

- the target multicore architecture is time predictable
  - all off-core latencies can be determined (upper bounded)
- all the threads run in parallel
  - one core per thread, one thread per core
  - no preemption, no migration
  - no interrupt
- synchronisation primitives are time analysable
  - fair servicing policy

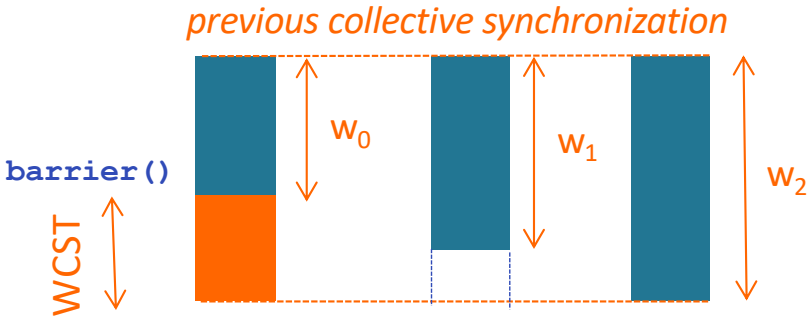
WCET of a parallel program



Analysis of synchronizations



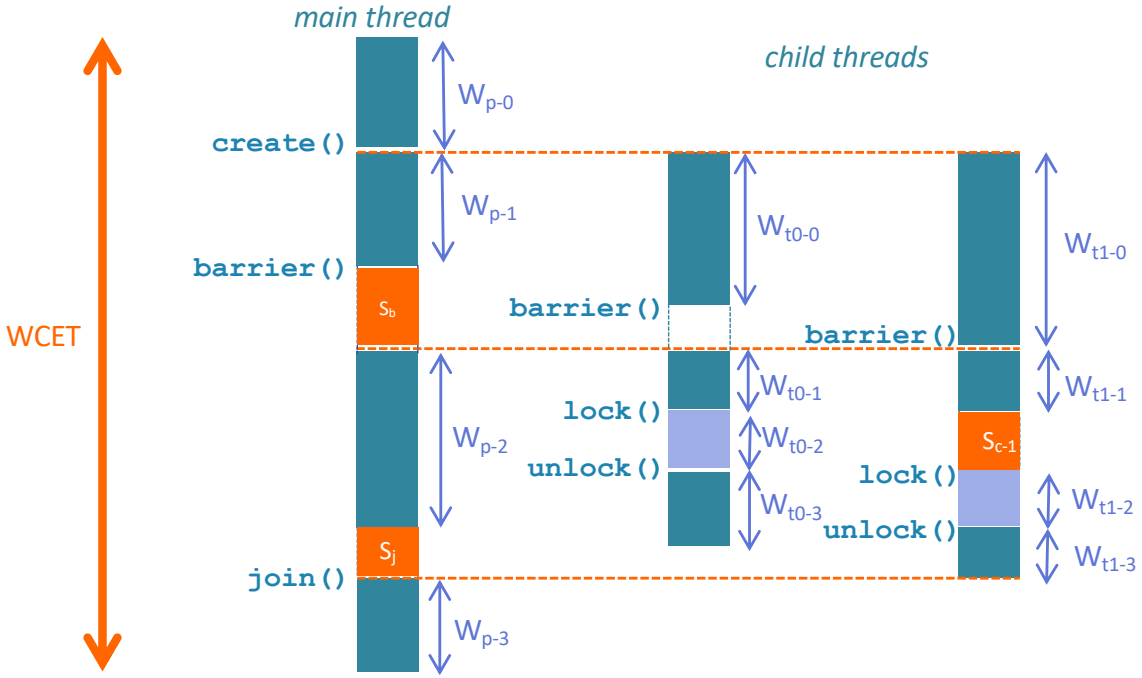
$WCST = w_1 + w_2$



$WCST = \max(0, w_1-w_0, w_2-w_0)$

WCST = Worst-Case Stall Time

WCET of a parallel program



$$WCET = W_{p-0} + W_{p-1} + S_b + W_{p-2} + S_j + W_{p-3}$$

$$S_b = \max(0, W_{t0-0} - W_{p-1}, W_{t1-0} - W_{p-1})$$

$$S_j = \max(0, W_{t0-1}+S_{c-0}+W_{t0-2}+W_{t0-3} - W_{p-2}, W_{t1-1}+S_{c-1}+W_{t1-2}+W_{t1-3} - W_{p-2})$$

$$S_{c-0} = W_{t1-2}$$

$$S_{c-1} = W_{t0-2}$$