Timing analysis and predictability of architectures

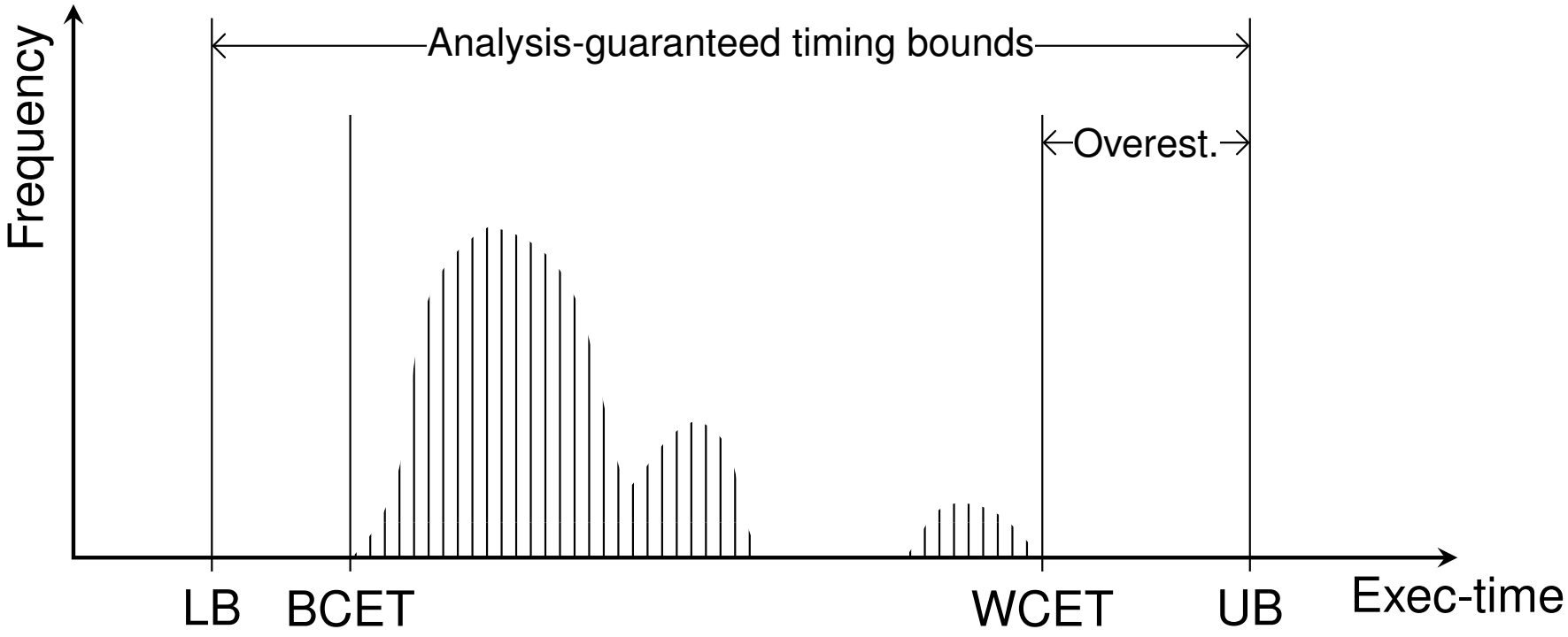
Cache analysis

Claire Maiza

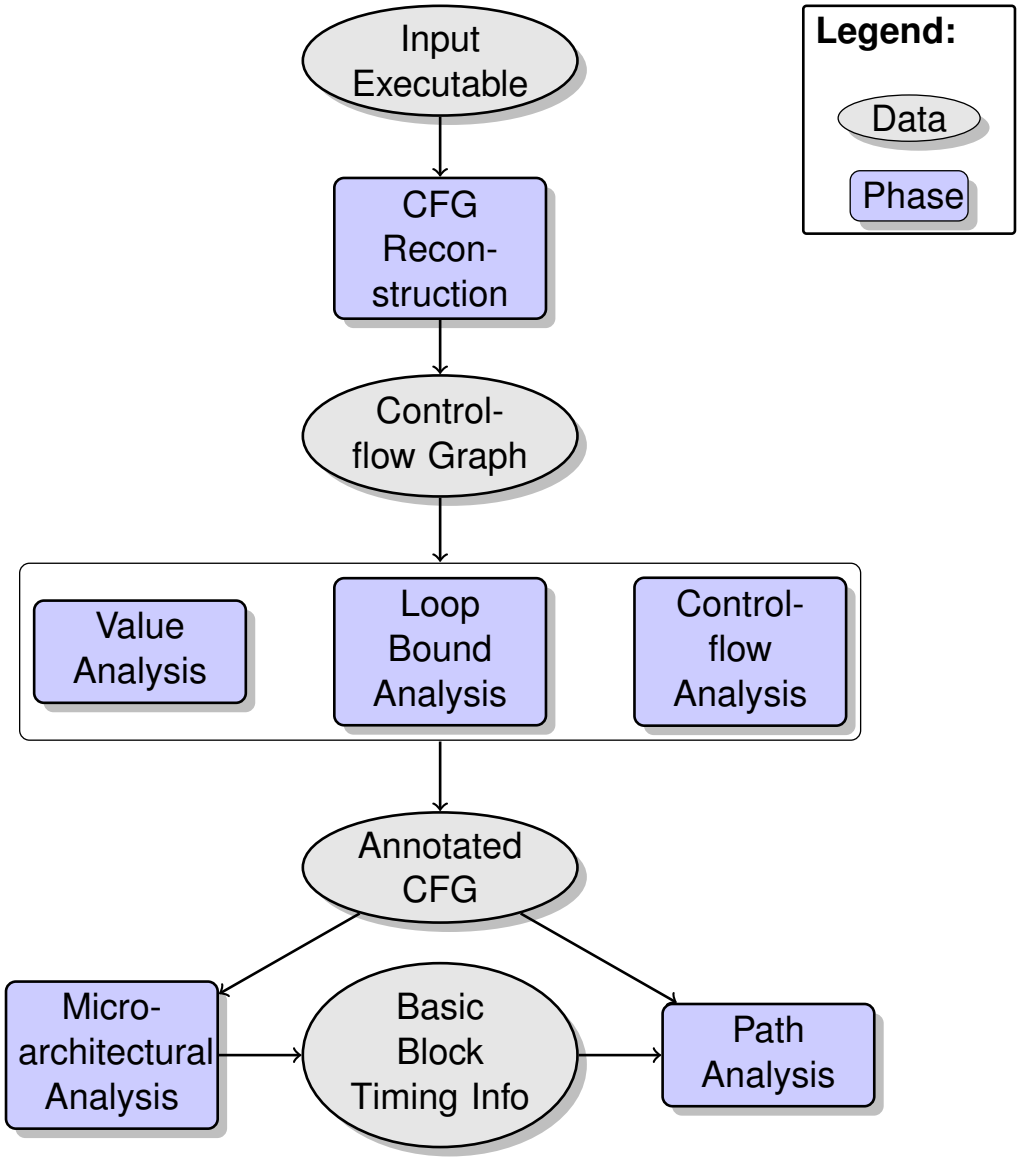
Verimag/INP

01/12/2010

Timing Analysis

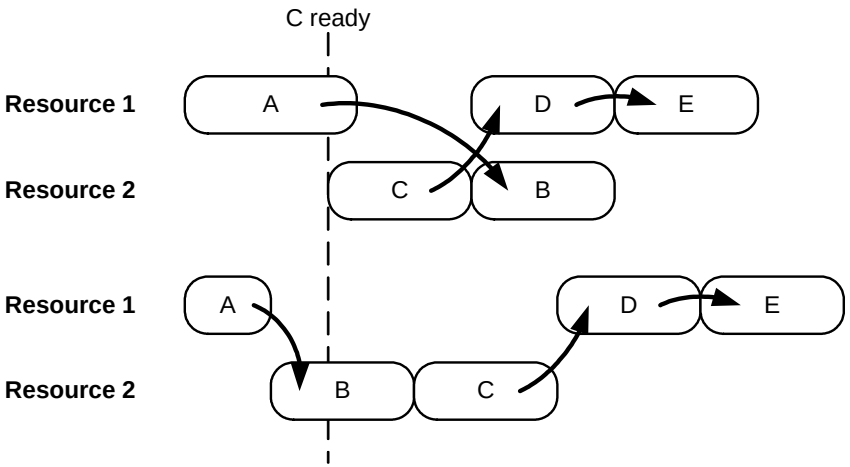


Static Timing Analysis

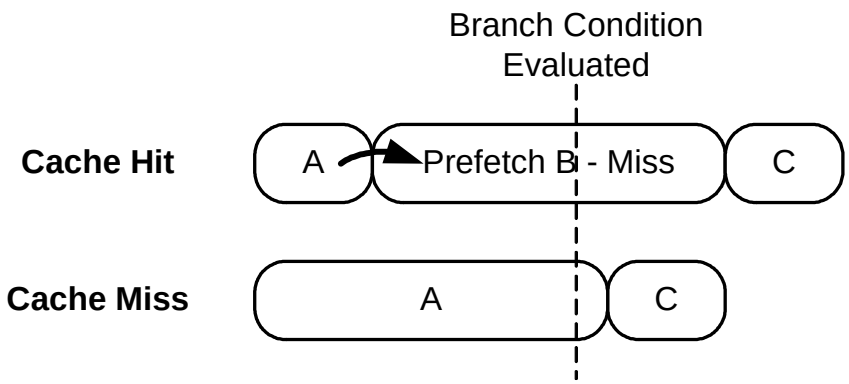


Timing anomalies

- When local worst-case does not lead to the global worst-case



Scheduling anomaly.



Speculation anomaly.

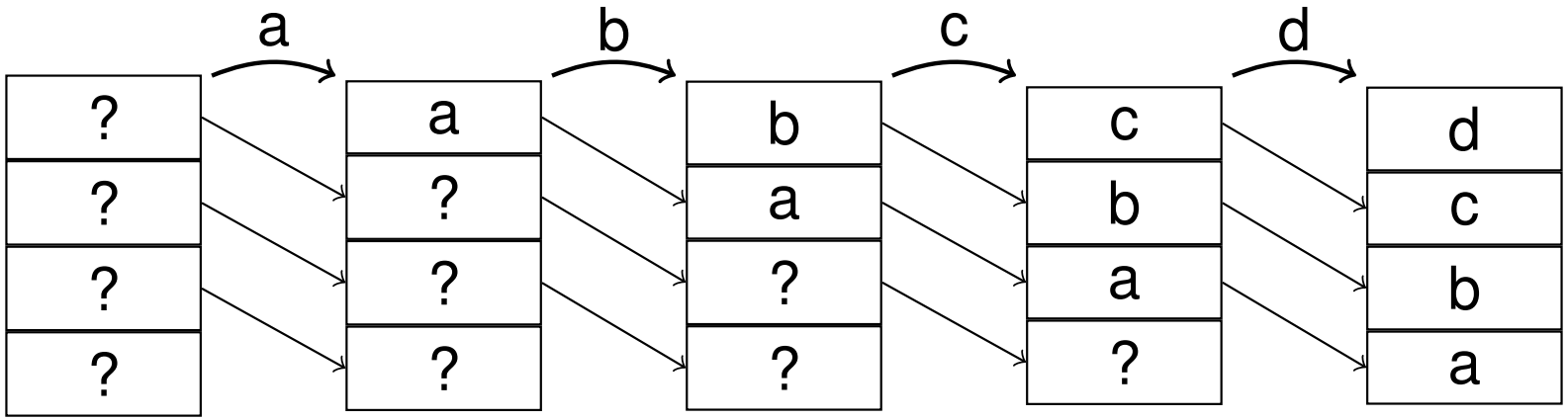
Classification of architectures

- *Timing compositional*
  - No timing anomalies
  - e. g., ARM7
- *Compositional with bounded effects*
  - Timing anomalies but no domino effects
  - e. g., TriCore (probably)
- *Non-compositional architectures*
  - Timing anomalies, domino effects
  - e. g., PPC 755

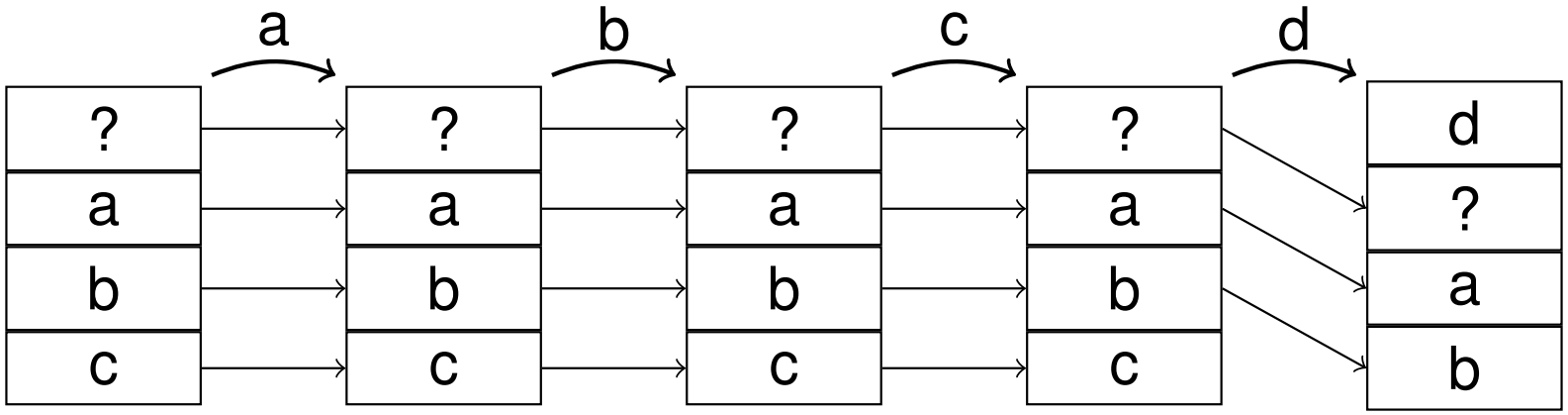
from Wilhelm et al.: Memory Hierarchies, Pipelines, and Buses for Future Architectures in Time-critical Embedded Systems, IEEE TCAD, July 2009

Why LRU is predictable?

LRU (Least Recently Used) “forgets” about past quickly:

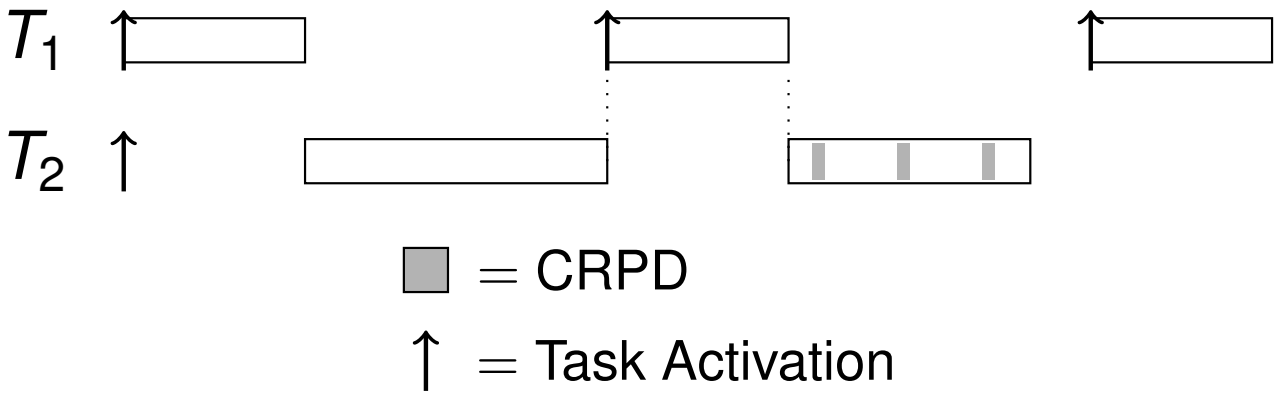


FIFO (First In First Out):



Preemption does not come for free!

- The preempting task “disturbs” the state of performance-enhancing features like caches and pipelines.
- Once the preempted task resumes its execution, the disturbance may cause additional *cache misses*.
- The additional execution time due to additional cache misses is known as the *cache-related preemption delay*.



How to take preemption cost into account?

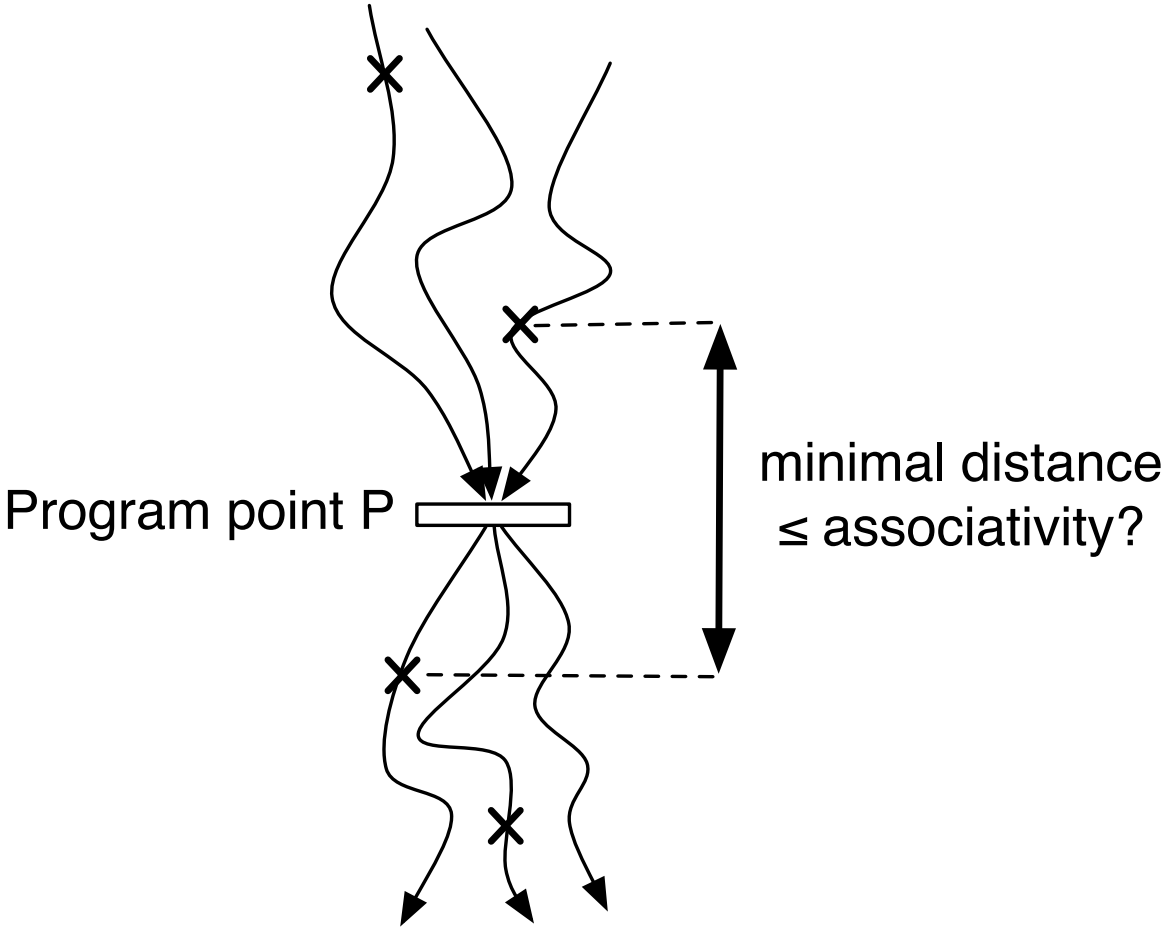
Where to account for preemption cost?

- Integrate into WCET Analysis: [Schneider, 2000]
  - assume cache misses everywhere
  - very pessimistic but easy to use in schedulability analysis
- WCET Analysis + CRPD Analysis: [Lee et al., 1996]
  - $WCET_{bound} + n \cdot CRPD_{bound} \geq$ execution time of task with up to n preemptions
  - more precise but only supported by very few schedulability analyses

CRPD Analyses

- Preempted Task:
  How many *useful* memory blocks are in the cache?
- Preempting Task:
  How much damage can the preempting task do to the cache contents of *any* task?
- Preempted + Preempting Task
  How much damage can the preempting task do to the useful cache contents of the preempted task?

Useful Cache Block Analysis



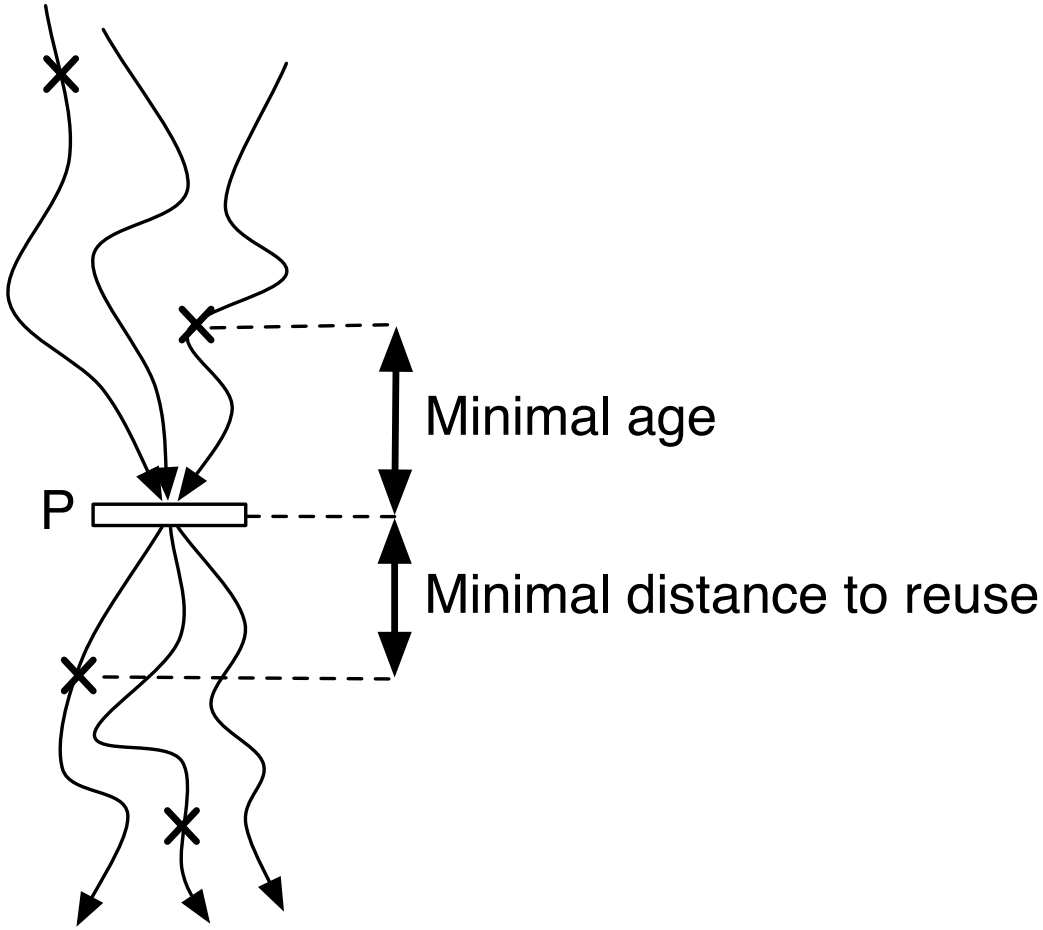
Useful = may be cached and may be reused

Useful Cache Block Analysis

Combination of two LRU-may-analyses:

What may be cached?

Forward May-Analysis!



What may be reused?

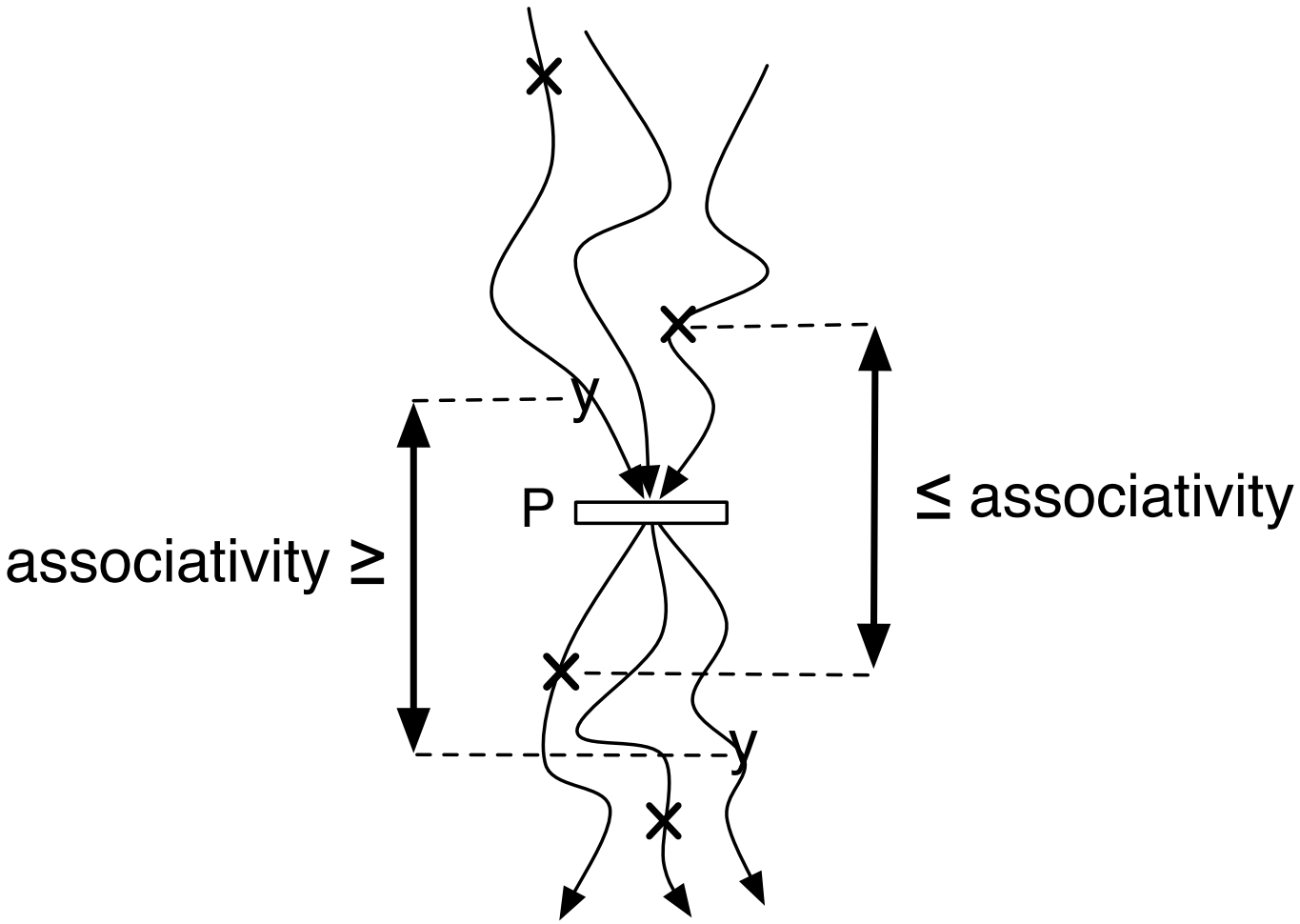
Backward May-Analysis!

Minimal age + Minimal distance to reuse $\leq$ associativity

$\Longrightarrow$ Memory block may be useful

Improvement: Path Analysis

Some blocks are never useful at the same time:



Literature:

[Tomiyama and Dutt, 2000, Negi et al., 2003, Staschulat et al., 2007]

Analysis of Preempted and Preempting Task: "Deeper" Combination [Altmeyer et al., 2010]

Definition (Resilience)

The resilience $res_P(m)$ of memory block m at program point P is the greatest l, such that all possible next accesses to m,

a) *that would be hits without preemption,*

b) *would still be hits in case of a preemption with l accesses at P.*

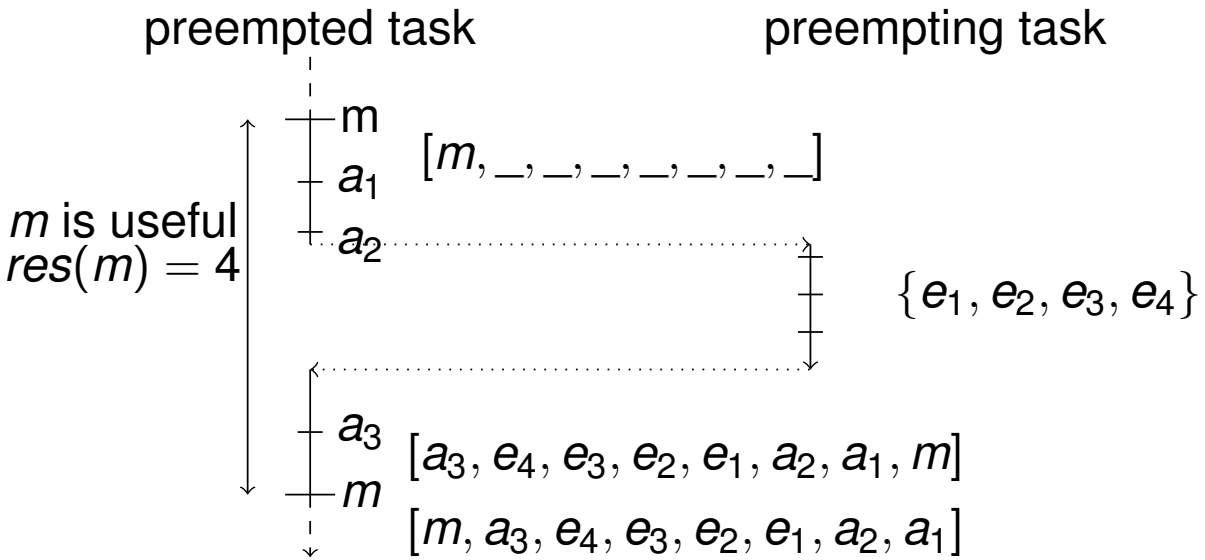
Analysis of Preempted and Preempting Task: "Deeper" Combination [Altmeyer et al., 2010]

Definition (Resilience)

The resilience $res_P(m)$ of memory block m at program point P is the greatest l, such that all possible next accesses to m,

a) *that would be hits without preemption,*

b) *would still be hits in case of a preemption with l accesses at P.*



Do existing approaches work for FIFO, PLRU, etc.?

Plain answer: No!

Do existing approaches work for FIFO, PLRU, etc.?

Plain answer: No!

Counterexample for FIFO [Burguière et al., 2009]:

$$\text{ECBs} = \{\mathbf{x}\} \quad \begin{cases} [b,a] \xrightarrow{a} [b,a] \xrightarrow{e^*} [e,b] \xrightarrow{b} [e,b] \xrightarrow{c^*} [c,e] \xrightarrow{e} [c,e] & \text{2 misses} \\ [\mathbf{x},b] \xrightarrow{a^*} [a,\mathbf{x}] \xrightarrow{e^*} [e,a] \xrightarrow{b^*} [b,e] \xrightarrow{c^*} [c,b] \xrightarrow{e^*} [e,c] & \text{5 misses} \end{cases}$$

- 2 *useful* cache blocks
- 1 block loaded by the preempting task
- associativity $= 2$
- But: number of additional misses$= 3$

Same result for PLRU.

Idea [Burguière et al., 2009]: Use Relative Competitiveness Results

Some relative competitiveness results:

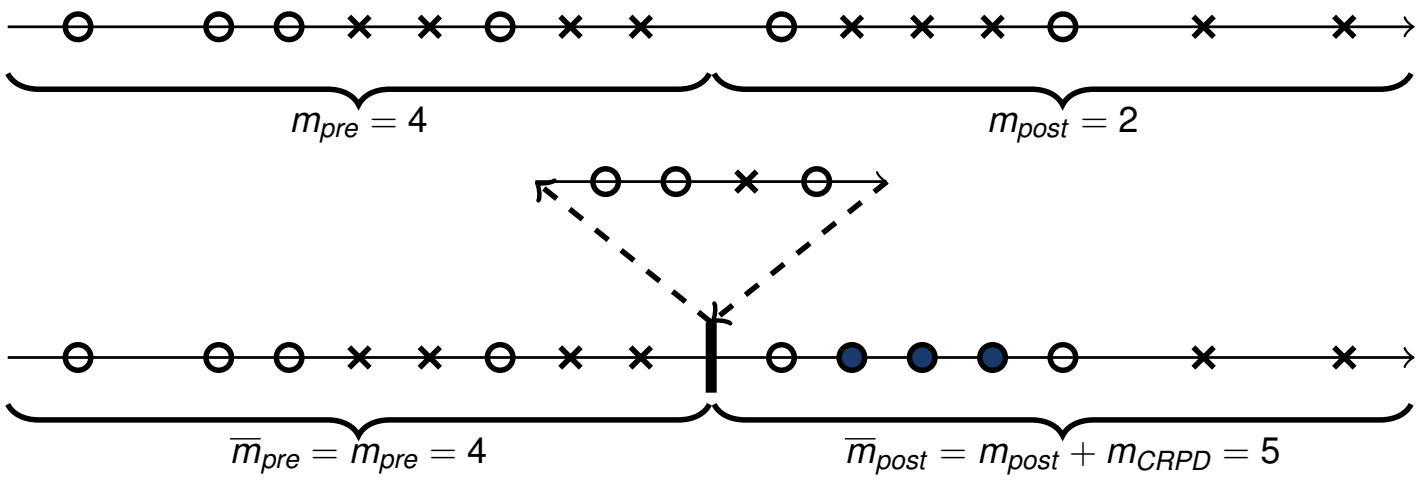
- PLRU(n) is $(1, 0)$-miss-competitive relative to LRU($1 + log_2 n$).
- FIFO(n) is $(\frac{n}{n-r+1}, r)$-miss-competitive relative to LRU(r).

$\Longrightarrow$ Performing WCET and CRPD analyses assuming LRU($1 + log_2 n$) replacement should give correct bounds for PLRU(n).

Can we also make use of non-$(1, 0)$-competitiveness?

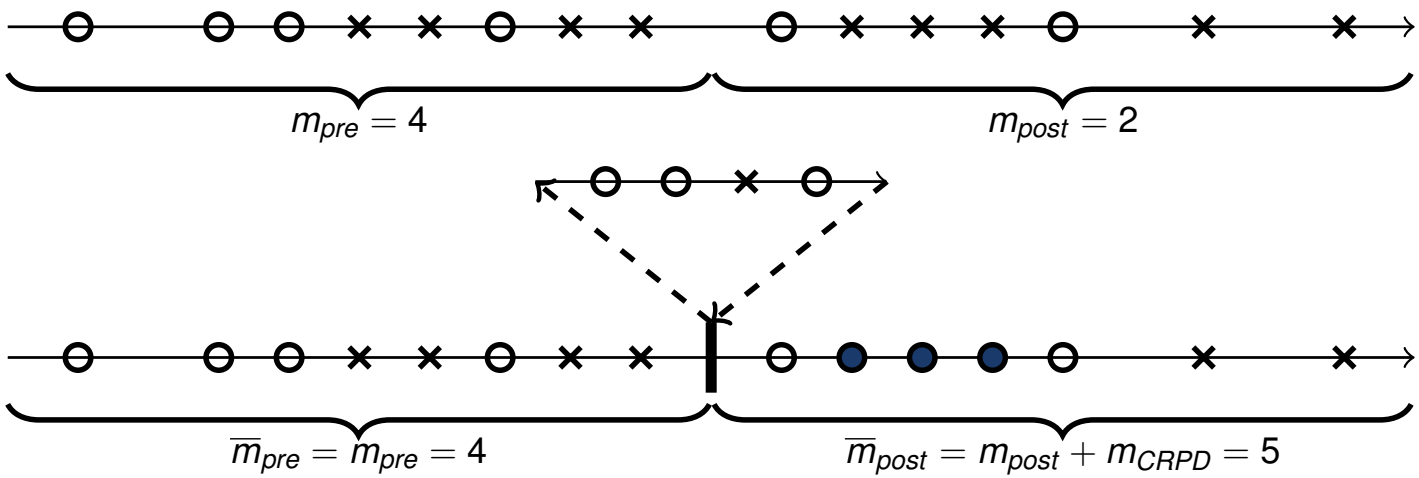
Applying Relative Competitiveness: A sequence of memory accesses

- Notation:
  - m = number of misses
  - $\overline{m}$ = number of misses in the case of preemption



Applying Relative Competitiveness: A sequence of memory accesses

- Notation:
  - m = number of misses
  - $\overline{m}$ = number of misses in the case of preemption

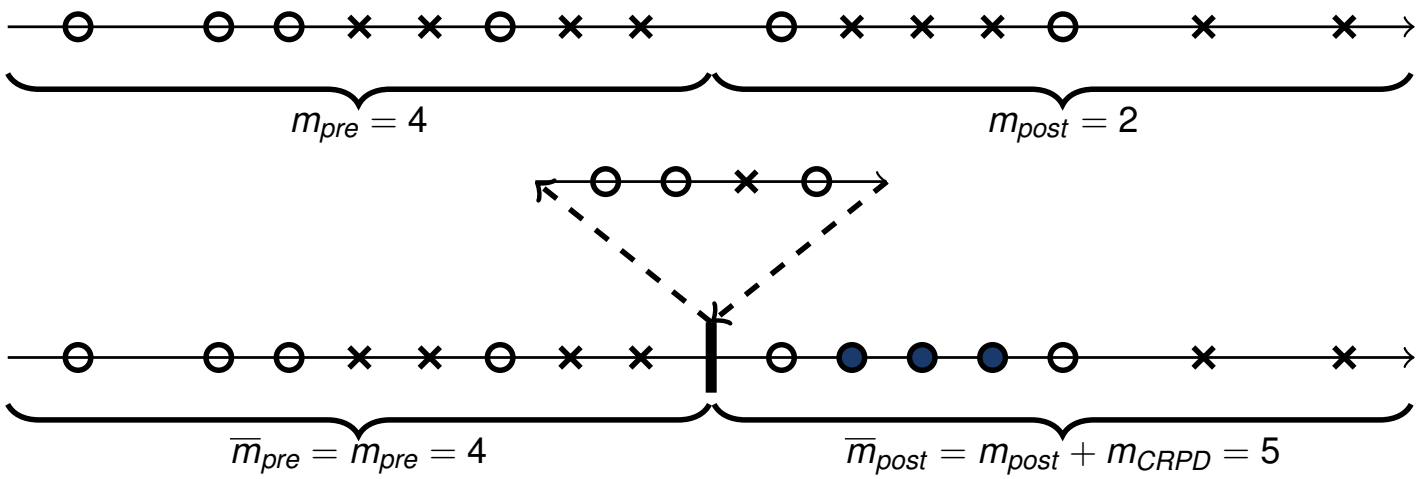


- Assume P(t) is (k, c)-miss-competitive rel. to LRU(s). Then:

$$\overline{m}^{\mathrm{P}(t)} = \overline{m}_{pre}^{\mathrm{P}(t)} + \overline{m}_{post}^{\mathrm{P}(t)}$$

Applying Relative Competitiveness: A sequence of memory accesses

- Notation:
  - m = number of misses
  - $\overline{m}$ = number of misses in the case of preemption

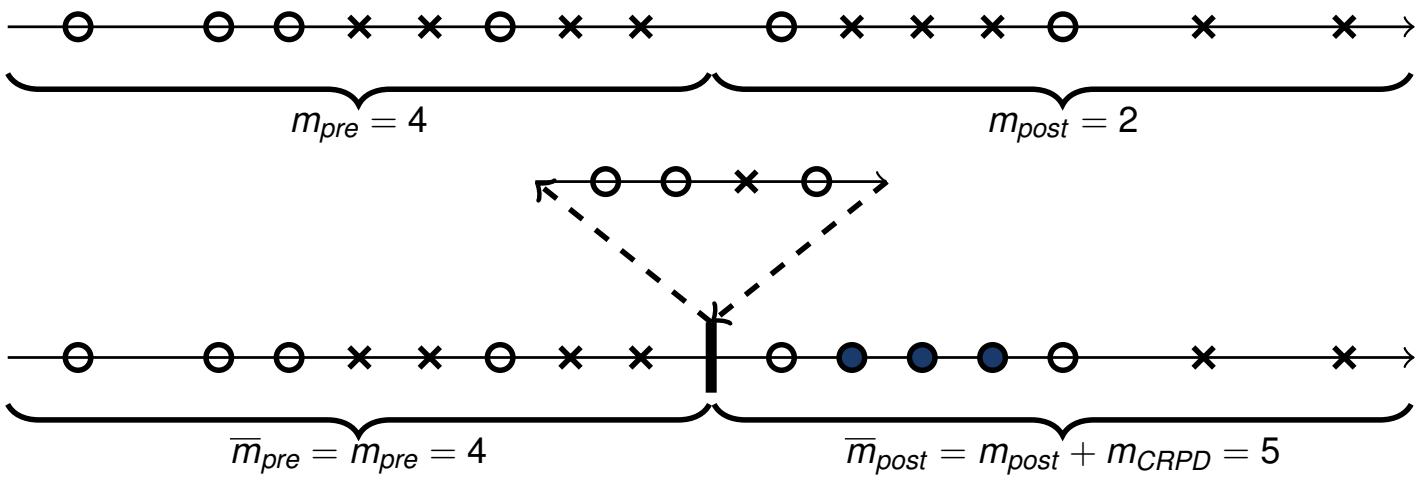


- Assume P(t) is (k, c)-miss-competitive rel. to LRU(s). Then:

$$\begin{aligned} \overline{m}^{\mathrm{P}(t)} &= \overline{m}^{\mathrm{P}(t)}_{pre} + \overline{m}^{\mathrm{P}(t)}_{post} \\ &\leq [k \cdot m^{\mathrm{LRU}(s)}_{pre} + c] + [k \cdot (m^{\mathrm{LRU}(s)}_{post} + m^{\mathrm{LRU}(s)}_{CRPD}) + c] \end{aligned}$$

Applying Relative Competitiveness: A sequence of memory accesses

- Notation:
  - m = number of misses
  - $\overline{m}$ = number of misses in the case of preemption



- Assume P(t) is (k, c)-miss-competitive rel. to LRU(s). Then:

$$\begin{aligned}
\overline{m}^{\mathrm{P}(t)} &= \overline{m}^{\mathrm{P}(t)}_{pre} + \overline{m}^{\mathrm{P}(t)}_{post} \\
&\leq [k \cdot m^{\mathrm{LRU}(s)}_{pre} + c] + [k \cdot (m^{\mathrm{LRU}(s)}_{post} + m^{\mathrm{LRU}(s)}_{CRPD}) + c] \\
&= [k \cdot m^{\mathrm{LRU}(s)} + c] + [k \cdot m^{\mathrm{LRU}(s)}_{CRPD} + c]
\end{aligned}$$

Applying Relative Competitiveness

- Assume $\mathrm{P}(t)$ is (k, c)-miss-competitive rel. to $\mathrm{LRU}(s)$. Then:

$$\overline{m}^{\mathrm{P}(t)} \leq [k \cdot m^{\mathrm{LRU}(s)} + c] + [k \cdot m_{CRPD}^{\mathrm{LRU}(s)} + c]$$

- In WCET analysis:
  Take into account $k \cdot m^{\mathrm{LRU}(s)} + c$ misses
- In CRPD analysis:
  Take into account $k \cdot m_{CRPD}^{\mathrm{LRU}(s)} + c$ misses

Summary

- CRPD bounded using a number of reloads:
  - a miss is the worst-case
  - the reload cost is bounded
- For LRU, the CRPD can be bounded by analyzing
  - the preempted task: useful cache block analysis
  - the preempting task
  - both, the preempted and the preempting task
    - Resilience Analysis
- Approaches do not carry over to FIFO, PLRU, etc. immediately
  - First approach: relative competitiveness

Altmeyer, S., Burguière, C., and Wilhelm, R. (2009).
Computing the maximum blocking time for scheduling with deferred preemption.
In *Workshop on Software Technologies for Future Dependable Distributed Systems*.

Altmeyer, S., Maiza, C., and Reineke, J. (2010).
Resilience analysis: Tightening the crpd bound for set-associative caches.
In *LCTES '10: Proceedings of the ACM SIGPLAN/SIGBED 2010 conference on Languages, compilers, and tools for embedded systems*, pages 153–162, New York, NY, USA. ACM.

Burguière, C., Reineke, J., and Altmeyer, S. (2009).
Cache-related preemption delay computation for set-associative caches—pitfalls and solutions.
In *Proceedings of 9th International Workshop on Worst-Case Execution Time (WCET) Analysis*.

Chiou, D. T. (1999).

Extending the reach of microprocessors: column and curious caching.
PhD thesis.
Supervisor-Arvind, and Supervisor-Rudolph, Larry.

Kirk, D. B. and Strosnider, J. K. (1990).
Smart (strategic memory allocation for real-time) cache design using the mips r3000.
In *IEEE Real-Time Systems Symposium*, pages 322–330.

Lee, C.-G., Hahn, J., Min, S. L., Ha, R., Hong, S., Park, C. Y., Lee, M., and Kim, C. S. (1996).
Analysis of cache-related preemption delay in fixed-priority preemptive scheduling.
In *Proceedings of the 17th IEEE Real-Time Systems Symposium (RTSS'96)*, page 264, Washington, DC, USA. IEEE Computer Society.

Lee, S., Lee, C.-G., Lee, M., Min, S. L., and Kim, C.-S. (1998).

Limited preemptible scheduling to embrace cache memory in real-time systems.
In *LCTES '98: Proceedings of the ACM SIGPLAN Workshop on Languages, Compilers, and Tools for Embedded Systems*, pages 51–64, London, UK. Springer-Verlag.

Mueller, F. (1995).
Compiler support for software-based cache partitioning.
SIGPLAN Not., 30(11):125–133.

Negi, H. S., Mitra, T., and Roychoudhury, A. (2003).
Accurate estimation of cache-related preemption delay.
In *Proceedings of the 1st ACM international conference on Hardware/software codesign and system synthesis (CODES+ISSS'03)*, pages 201–206, New York, NY, USA. ACM.

Schneider, J. (2000).
Cache and pipeline sensitive fixed priority scheduling for preemptive real-time systems.

In *In Proceedings of the 21st IEEE Real-Time Systems Symposium (RTSS'2000)*, pages 195–204.

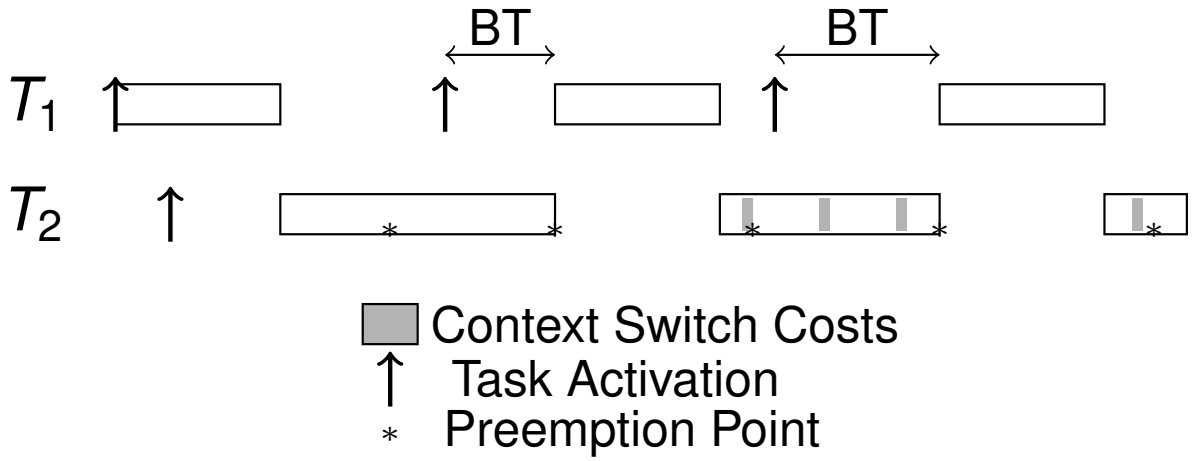
Staschulat, J. et al. (2007).
Scalable precision cache analysis for real-time software.
Trans. on Embedded Computing Sys., 6(4):25.

Tomiyama, H. and Dutt, N. D. (2000).
Program path analysis to bound cache-related preemption delay in preemptive real-time systems.
In *Proceedings of the 8th ACM international workshop on Hardware/software codesign (CODES'00)*, pages 67–71, New York, NY, USA. ACM.

Wolfe, A. (1994).
Software-based cache partitioning for real-time applications.
J. Comput. Softw. Eng., 2(3):315–327.

Deferred Preemption - simplifying the problem

- Restrict preemptions to a set of predefined *preemption points*.
- Introduces new problem: blocking time, time until next preemption point is reached.



Where to place preemptions points, s.t.

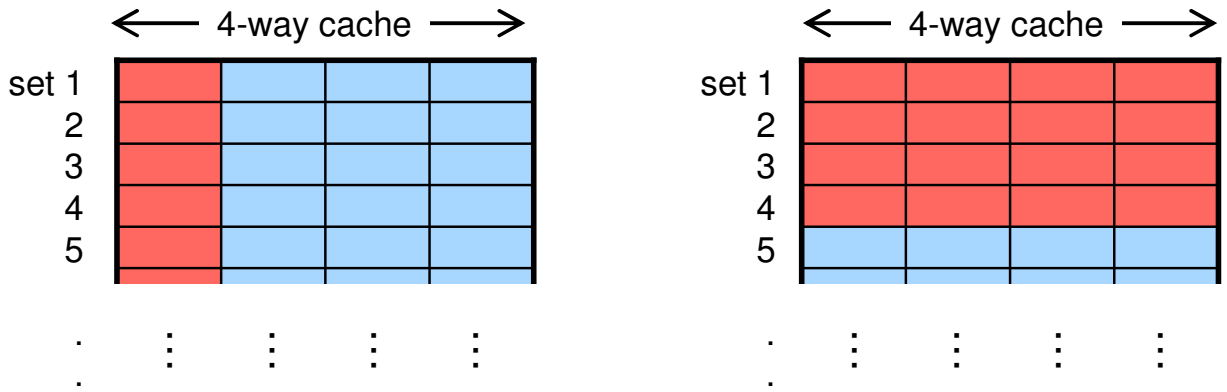
- CRPD is minimized, and
- *Maximum Blocking Time* is minimized.

Analysis to determine maximum blocking time for given set of preemption points: [Lee et al., 1998, Altmeyer et al., 2009]

Cache Partitioning - eliminating the problem

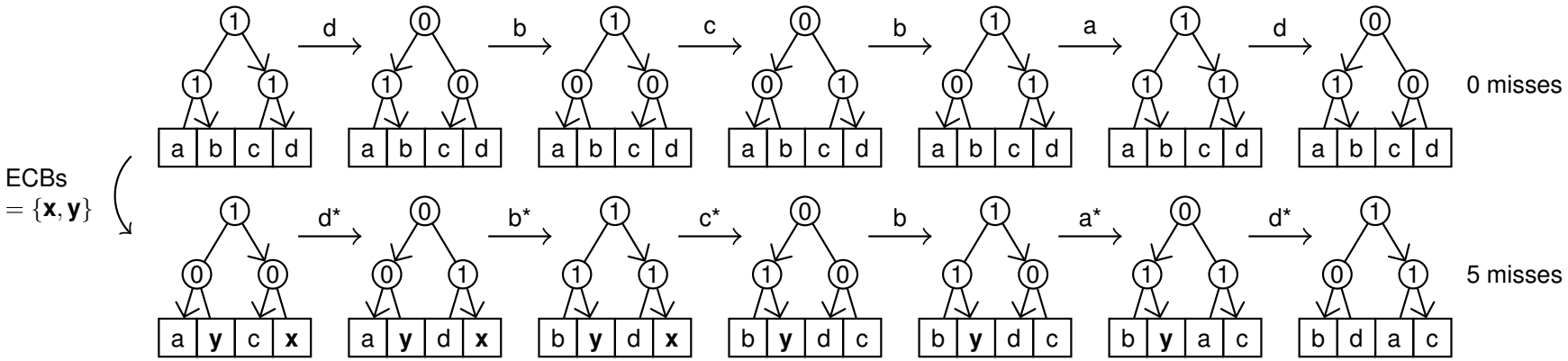
Additional cache misses are due to interference on the cache.

$\Longrightarrow$ Cache Partitioning eliminates this interference.



- Software-based Cache Partitioning [Wolfe, 1994, Mueller, 1995]:
  - Change layout of instructions and data such that tasks map to disjoint cache sets
  - Particularly difficult for large arrays
- Hardware-based Cache Partitioning [Kirk and Strosnider, 1990, Chiou, 1999]:
  - Partition cache by cache sets and/or cache ways
  - Increases hardware cost
  - Renewed interest in multi-cores with shared caches

CRPD for PLRU: Pitfalls



- $|\mathrm{UCB}(s)| = 4$
- $|\mathrm{ECB}(s)| = 2$
- $n = 4$
- But: number of additional misses$= 5$

Resilience - Domain of the analysis

$$\mathbb{D} : \mathbb{D}_{ca} \times \mathbb{D}_{ua} \tag{1}$$

with

$$\mathbb{D}_{ca} : \mathbb{M} \rightarrow \{0, \ldots, k-1\} \tag{2}$$

and

$$\mathbb{D}_{ua} : \mathbb{M} \rightarrow \{0, \ldots, k-1, \infty\} \tag{3}$$

Resilience - Transfer functions

$$t_{ua} : \mathbb{D}_{ua} \times \mathbb{M} \rightarrow \mathbb{D}_{ua}$$

$$t_{ua}(ua, m) := \lambda m'. \begin{cases} 0 & m' = m \\ ua(m') & ua(m') \geq ua(m) \\ ua(m') + 1 & ua(m') < ua(m) \wedge ua(m') < k - 1 \\ \infty & otherwise \end{cases} \quad (4)$$

$$t_{ca} : \mathbb{D}_{ca} \times \mathbb{D}_{ua} \times 2^{\mathbb{M}} \times \mathbb{M} \rightarrow \mathbb{D}_{ca}$$

$$t_{ca}(ca, ua, \text{UCB}, m) := \lambda m'. \begin{cases} 0 & m' = m \vee m' \notin \text{UCB} \\ ca(m') & ca(m') \geq ua(m) \vee ca(m') = k - 1 \\ ca(m') + 1 & ca(m') < ua(m) \end{cases} \quad (5)$$