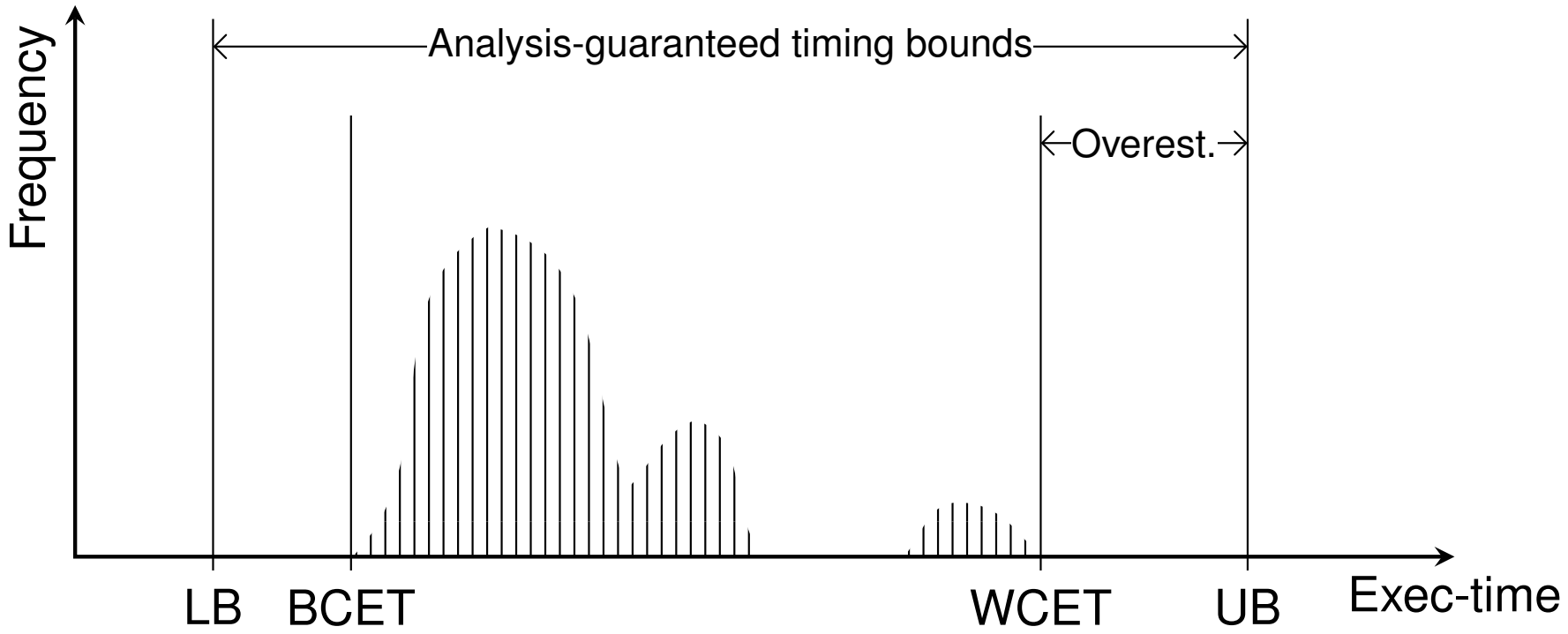
Static Cache Analysis and Design for Predictable Multi-Core

Claire Maïza (Burguière)

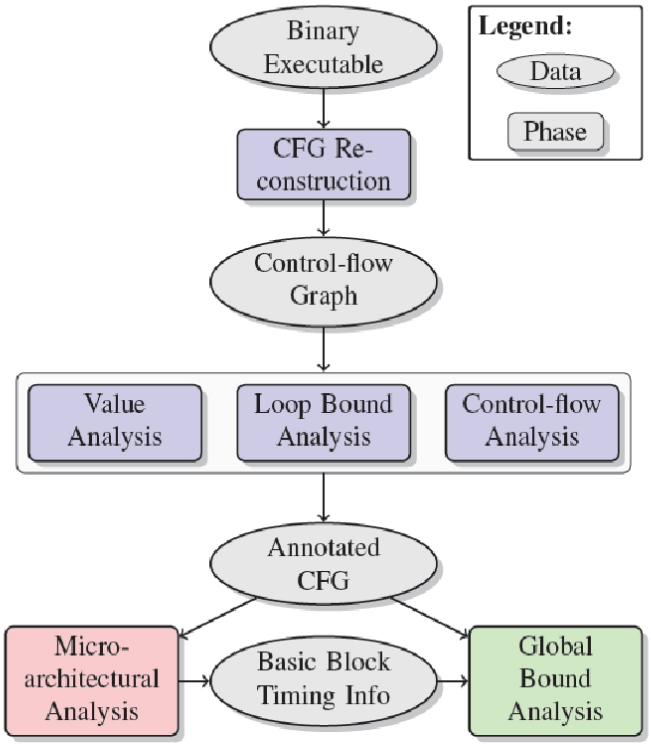
Department of Computer Science
Saarland University

LIP6 2010

Timing Analysis



Timing Analysis



- Restriction: uninterrupted execution of the task on a single-core

Structure of the Talk

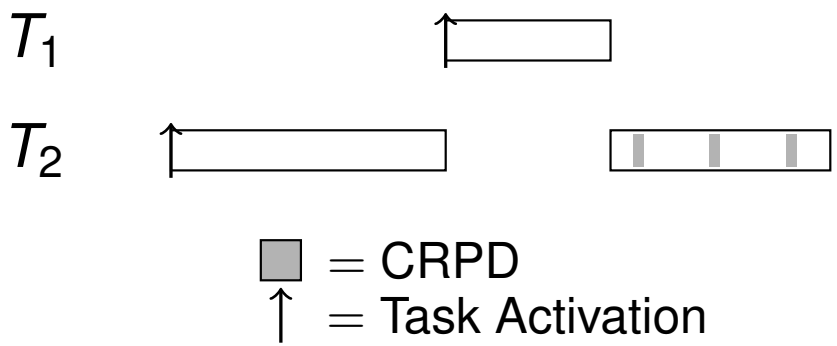
- Preemptive System:
  - Example of cache analysis: Resilience analysis
- Multi-Core:
  - Predictability on a single-core level
  - Design principles for predictable multi-core

Context - Preemptive Systems

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Cache related preemption delay (CRPD):
  - Impact of preemption on the cache content
  - Overall cost of additional reloads due to preemption

  $\Rightarrow$ Analysis aims at bounding the number of additional cache misses due to preemption



CRPD for Set-Associative Caches - LRU

- CRPD computation:
  - preempted task: Useful Cache Blocks (UCB)
  - preempting task: Evicting Cache Blocks (ECB)
- CRPD from UCB **and** ECB:
  - Previous combination overestimates

  $\Rightarrow$ Resilience analysis

Useful Cache Block - [Lee et al., 1996]

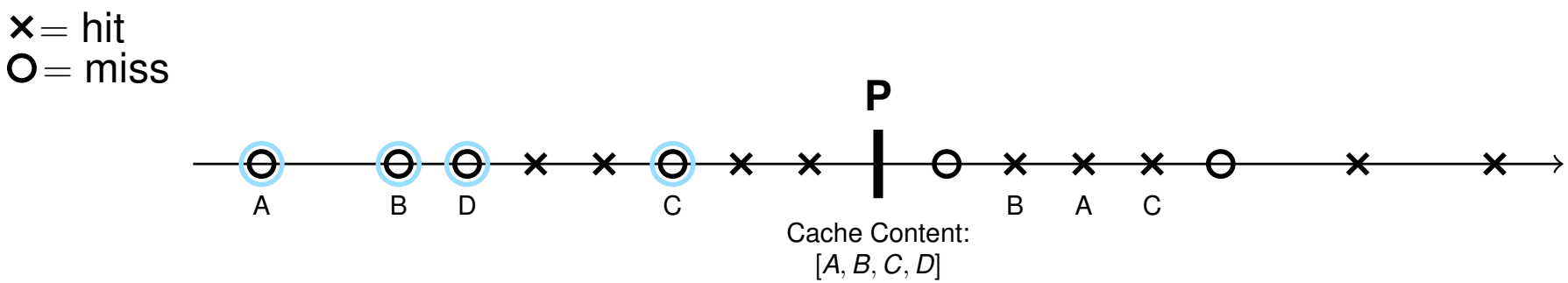
SAARLAND UNIVERSITY
COMPUTER SCIENCE

Definition (Useful Cache Block)

A memory block m at program point P is called a useful cache block, if

a) *m may be cached at P*

b) *m may be reused at program point P′ that may be reached from P with no eviction of m on this path.*



Useful Cache Block - [Lee et al., 1996]

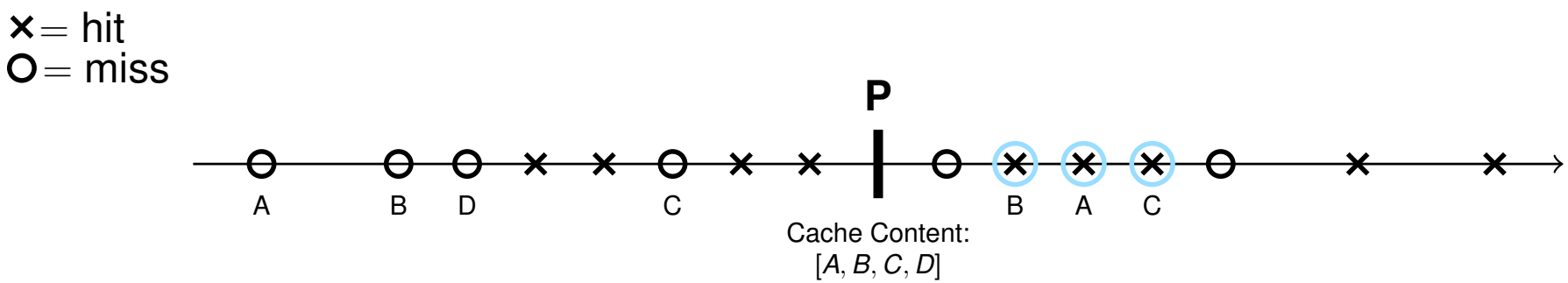
SAARLAND UNIVERSITY
COMPUTER SCIENCE

Definition (Useful Cache Block)

A memory block m at program point P is called a useful cache block, if

a) *m may be cached at P*

b) *m may be reused at program point P′ that may be reached from P with no eviction of m on this path.*



Useful Cache Block - [Lee et al., 1996]

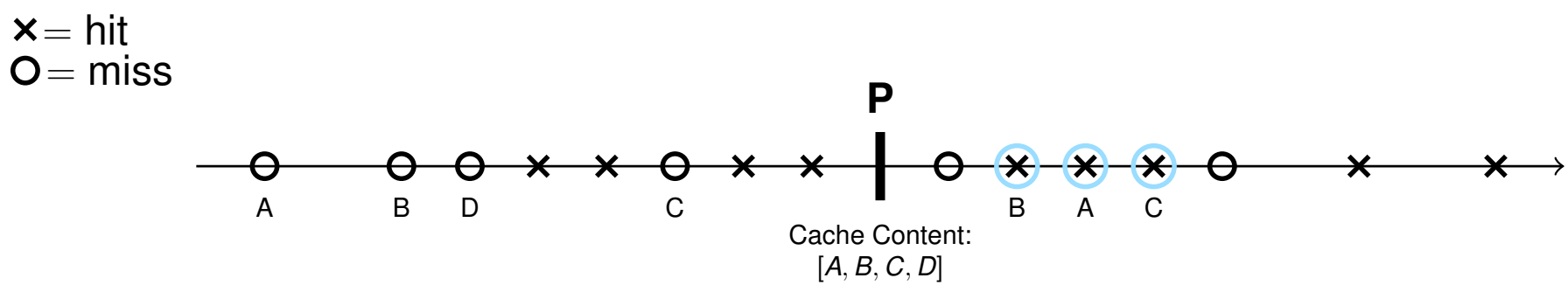
SAARLAND UNIVERSITY
COMPUTER SCIENCE

Definition (Useful Cache Block)

A memory block m at program point P is called a useful cache block, if

a) *m may be cached at P*

b) *m may be reused at program point P′ that may be reached from P with no eviction of m on this path.*



$$\mathrm{CRPD}_{\mathrm{UCB}} = \sum_{s=1}^{c} \mathrm{CRPD}^{s}_{\mathrm{UCB}}$$

$$\mathrm{CRPD}^{s}_{\mathrm{UCB}} = \mathrm{BRT} \times min(|\mathrm{UCB}(s)|, n)$$

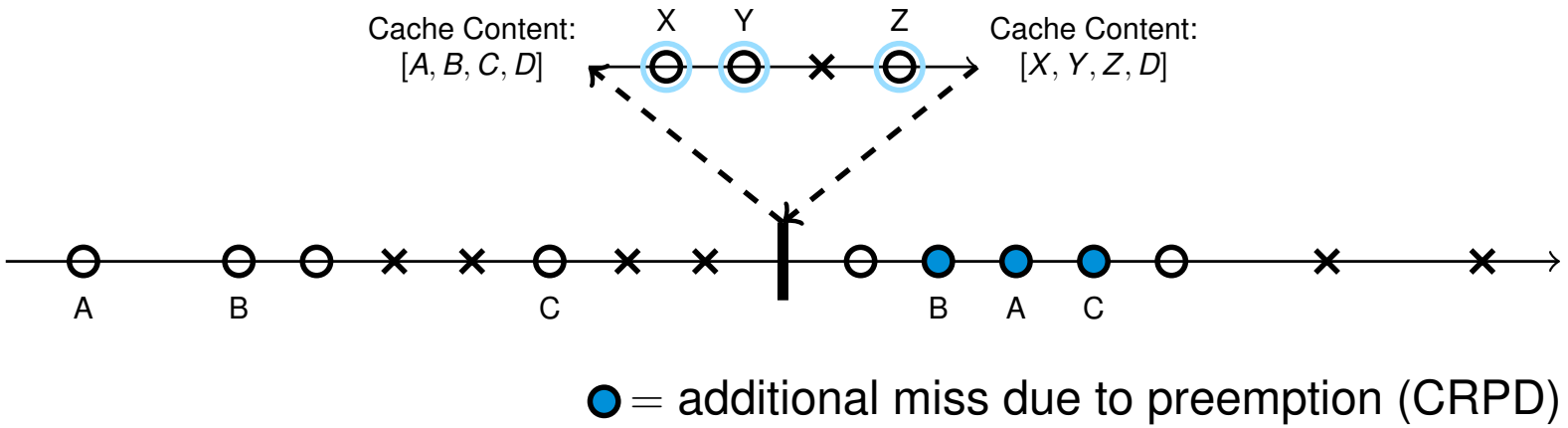
n = associativity

BRT = Block Reload Time

Evicting Cache Blocks [Tomiyama & Dutt, 2000]

Definition (Evicting Cache Blocks (ECB))

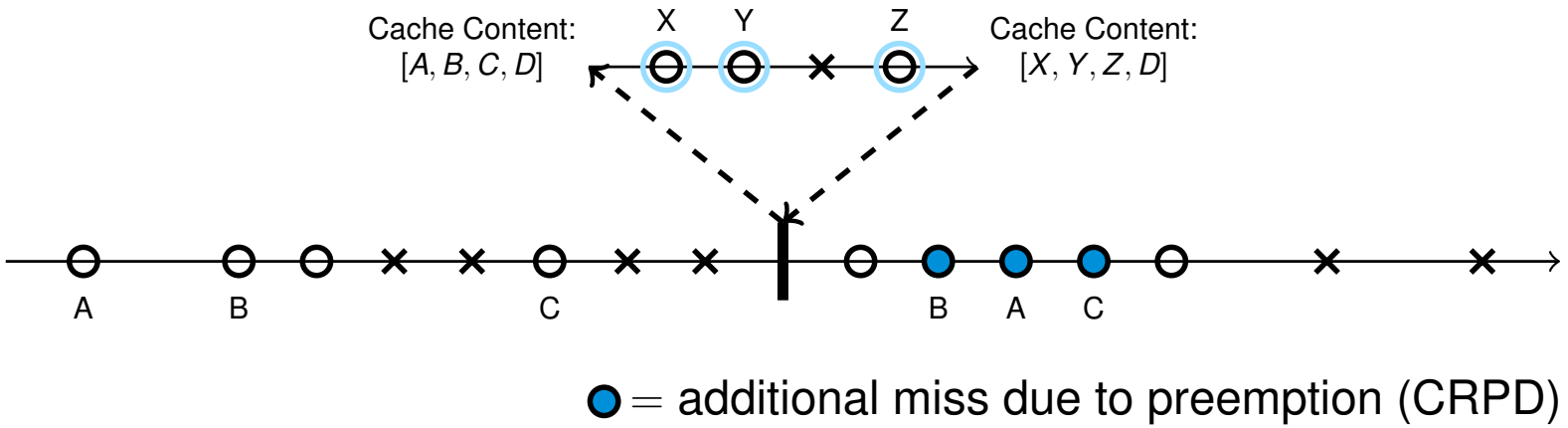
A memory block of the preempting task is called an evicting cache block, if it may be accessed during the execution of the preempting task.



Evicting Cache Blocks [Tomiyama & Dutt, 2000]

Definition (Evicting Cache Blocks (ECB))

A memory block of the preempting task is called an evicting cache block, if it may be accessed during the execution of the preempting task.



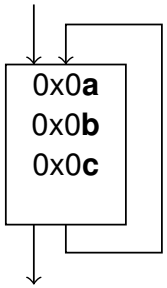
$$\text{CRPD}^{s}_{\text{ECB}} = \begin{cases} 0 & \text{if ECB}(s) = \emptyset \\ \text{BRT} \times n & \text{otherwise} \end{cases}$$

Impact of the Preempting Task on the Preempted Task

CRPD (using UCB and ECB)

$$CRPD_{UCB\&ECB} = \sum_{s=1}^{c} min(CRPD^{s}_{UCB}, CRPD^{s}_{ECB})$$

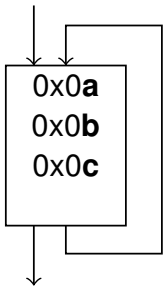
Impact of the Preempting Task on the Preempted Task (Example)



$$[c, b, a, x] \xrightarrow{a} [a, c, b, x] \xrightarrow{b} [b, a, c, x] \xrightarrow{c} [c, b, a, x]$$

no miss

Impact of the Preempting Task on the Preempted Task (Example)



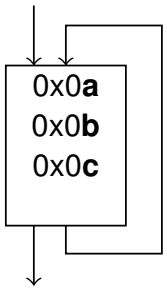
ECBs $= \{\mathbf{e}\}$

$[c, b, a, x] \xrightarrow{a} [a, c, b, x] \xrightarrow{b} [b, a, c, x] \xrightarrow{c} [c, b, a, x]$ no miss

$[\mathbf{e}, c, b, a] \xrightarrow{a} [a, \mathbf{e}, c, b] \xrightarrow{b} [b, a, \mathbf{e}, c] \xrightarrow{c} [c, b, a, \mathbf{e}]$ no miss

- $\text{CRPD}_{\text{UCB}} \Rightarrow |\text{UCB}| = 3$
- $\text{CRPD}_{\text{ECB}} \Rightarrow n = 4$
- $\text{CRPD}_{\text{UCB\&ECB}} = min(\text{CRPD}_{\text{UCB}}, \text{CRPD}_{\text{ECB}}) \Rightarrow 3$
  - Overestimation: number of additional misses$= 0 < 3$

Impact of the Preempting Task on the Preempted Task (Example)



ECBs $= \{\mathbf{e}\}$

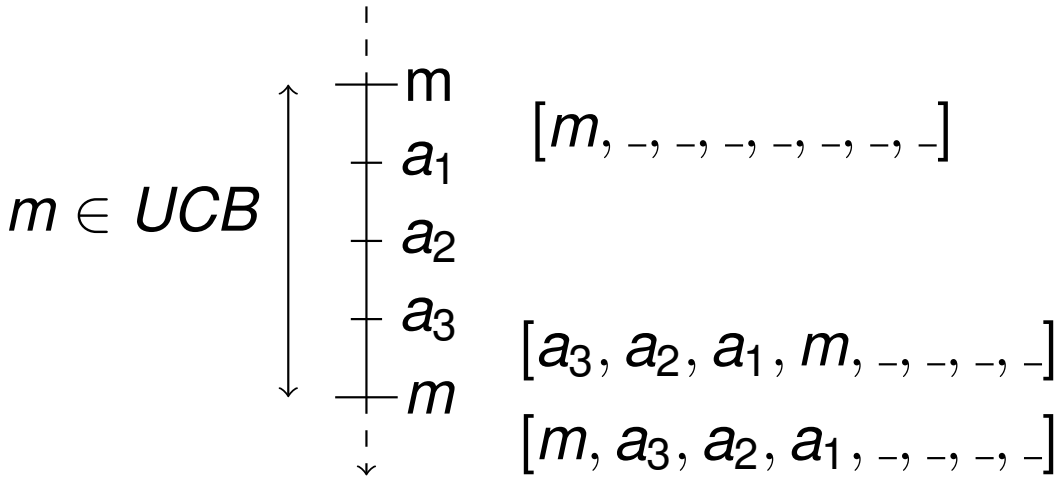
$[c, b, a, x] \xrightarrow{a} [a, c, b, x] \xrightarrow{b} [b, a, c, x] \xrightarrow{c} [c, b, a, x]$ no miss

$[\mathbf{e}, c, b, a] \xrightarrow{a} [a, \mathbf{e}, c, b] \xrightarrow{b} [b, a, \mathbf{e}, c] \xrightarrow{c} [c, b, a, \mathbf{e}]$ no miss

- $\text{CRPD}_{\text{UCB}} \Rightarrow |\text{UCB}| = 3$
- $\text{CRPD}_{\text{ECB}} \Rightarrow n = 4$
- $\text{CRPD}_{\text{UCB\&ECB}} = min(\text{CRPD}_{\text{UCB}}, \text{CRPD}_{\text{ECB}}) \Rightarrow 3$
  - Overestimation: number of additional misses$= 0 < 3$
- Why?
  - $|\text{ECB}|$ to evict a UCB $= 2$
  - but, $|\text{ECB}| = 1$
  - One single ECB is not sufficient to evict a UCB

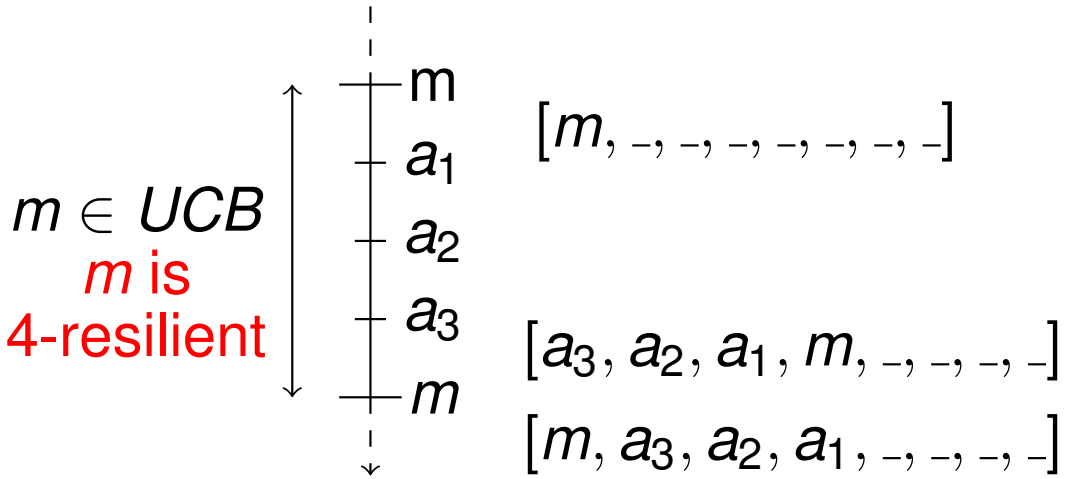
Refinement

Determining $max_{|ECB|}$ s.t. no additional cache miss occur



Refinement

Determining $max_{|ECB|}$ s.t. no additional cache miss occur



Resilience Analysis

Definition (l-Resilience)

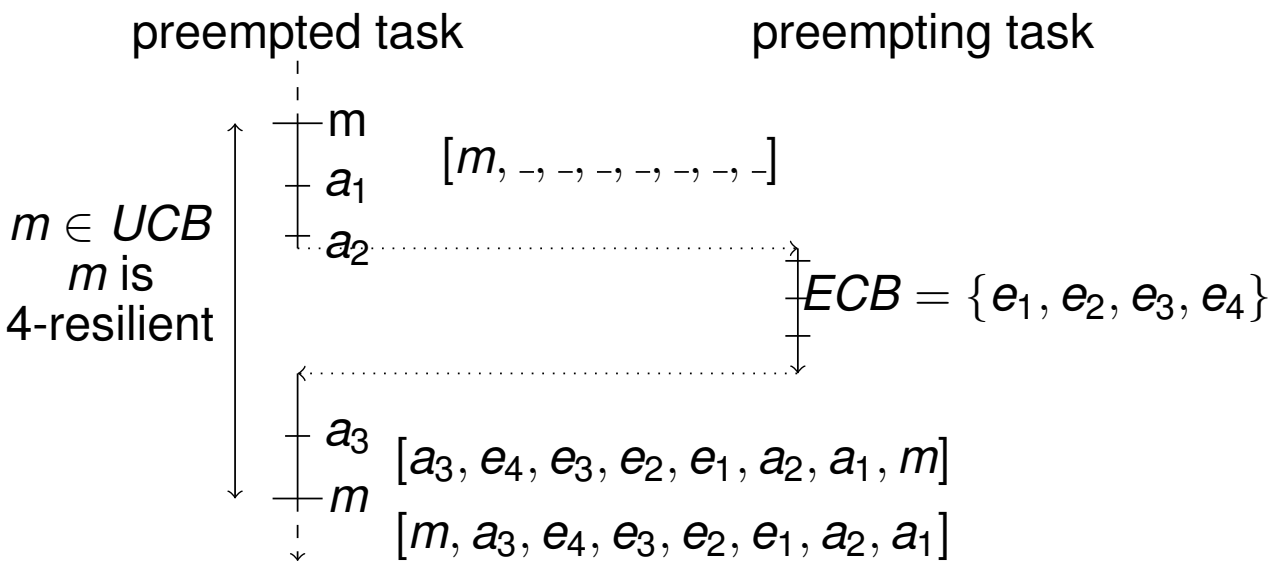
A memory block m is called l-resilient at program point P, if all possible next accesses to m that would be hits, would still be hits in case of a preemption at P with l accesses.

Resilience Analysis

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Definition (l-Resilience)

A memory block m is called l-resilient at program point P, if all possible next accesses to m that would be hits, would still be hits in case of a preemption at P with l accesses.



if $|\text{ECB}| \leq l$ then the UCB is not evicted

CRPD using Resilience

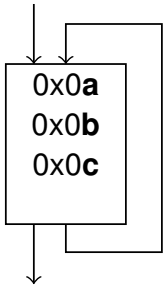
SAARLAND UNIVERSITY
COMPUTER SCIENCE

CRPD (combining UCB and ECB by using resilience)

$$CRPD \leq BRT \times \Big| \overbrace{\underbrace{UCB}_{\text{may be useful}} \setminus \underbrace{\{m \mid m \text{ is } |ECB|\text{-resilient}\}}_{\text{must remain useful}}}^{\text{may contribute to CRPD}} \Big|$$

CRPD using Resilience - Example

CRPD using Resilience - Example

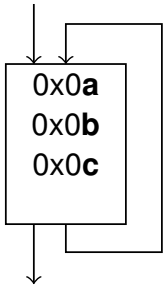


ECBs $= \{\mathbf{e}\}$

$[c, b, a, x] \xrightarrow{a} [a, c, b, x] \xrightarrow{b} [b, a, c, x] \xrightarrow{c} [c, b, a, x]$ no miss

$[\mathbf{e}, c, b, a] \xrightarrow{a} [a, \mathbf{e}, c, b] \xrightarrow{b} [b, a, \mathbf{e}, c] \xrightarrow{c} [c, b, a, \mathbf{e}]$ no miss

CRPD using Resilience - Example



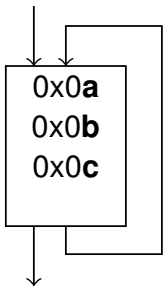
ECBs $= \{\mathbf{e}\}$

$[c, b, a, x] \xrightarrow{a} [a, c, b, x] \xrightarrow{b} [b, a, c, x] \xrightarrow{c} [c, b, a, x]$ no miss

$[\mathbf{e}, c, b, a] \xrightarrow{a} [a, \mathbf{e}, c, b] \xrightarrow{b} [b, a, \mathbf{e}, c] \xrightarrow{c} [c, b, a, \mathbf{e}]$ no miss

- $|\text{ECB}| = 1$
- a, b and c are 1-resilient
- $CRPD^{res}_{\text{UCB\&ECB}} = BRT \times |UCB \setminus \{m \mid m \text{ is } |\text{ECB}|\text{-resilient}\}| = 0$

CRPD using Resilience - Example



ECBs $= \{\mathbf{e}\}$

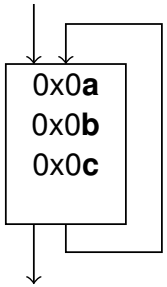
$[c, b, a, x] \xrightarrow{a} [a, c, b, x] \xrightarrow{b} [b, a, c, x] \xrightarrow{c} [c, b, a, x]$ no miss

$[\mathbf{e}, c, b, a] \xrightarrow{a} [a, \mathbf{e}, c, b] \xrightarrow{b} [b, a, \mathbf{e}, c] \xrightarrow{c} [c, b, a, \mathbf{e}]$ no miss

- $|\text{ECB}| = 1$
- a, b and c are 1-resilient
- $CRPD^{res}_{\text{UCB\&ECB}} = BRT \times |UCB \setminus \{m \mid m \text{ is } |\text{ECB}|\text{-resilient}\}| = 0$

- Instead of: $\text{CRPD}_{\text{UCB\&ECB}} = min(\text{CRPD}_{\text{UCB}}, \text{CRPD}_{\text{ECB}}) = 3 \times BRT$

CRPD using Resilience - Example



ECBs $= \{\mathbf{e}\}$

$[c, b, a, x] \xrightarrow{a} [a, c, b, x] \xrightarrow{b} [b, a, c, x] \xrightarrow{c} [c, b, a, x]$ no miss

$[\mathbf{e}, c, b, a] \xrightarrow{a} [a, \mathbf{e}, c, b] \xrightarrow{b} [b, a, \mathbf{e}, c] \xrightarrow{c} [c, b, a, \mathbf{e}]$ no miss

- $|\text{ECB}| = 1$
- a, b and c are 1-resilient
- $CRPD^{res}_{\text{UCB\&ECB}} = BRT \times |UCB \setminus \{m \mid m \text{ is } |\text{ECB}|\text{-resilient}\}| = 0$

- Instead of: $\text{CRPD}_{\text{UCB\&ECB}} = min(\text{CRPD}_{\text{UCB}}, \text{CRPD}_{\text{ECB}}) = 3 \times BRT$
- Resilience analysis leads to a more precise CRPD

Limitations of CRPD Estimation

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Cache Policy: LRU
  - FIFO and PLRU cannot be ”directly” analysed using UCB/ECB.
- Architecture: Need a constant bound on the block reload time
  - The number of additional misses is used to bound the interferences.

Predictable Multi-Core Architecture

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Predictability: hard to quantify
- Predictable cores are a prerequisite for predictable multi-cores
- General problem: *Sharing* of resources
  - Main memory, caches
  - Busses
  - I/O
  - Flash memory
- Sharing may be
  - fundamental (necessary access to application global variables)
  - incidental (processors happen to use the same bus for access to non-shared devices)

Predictability on the Single-Core Level: Caches

Several new notions regarding cache replacement policies:

- *Predictability*:
  quantitative measure of how fast information about cache state can be gained
- *Competitiveness*:
  quantitative measure of how numbers of hits and misses of different policies relate
- *Sensitivity*:
  quantitative measure of how the number of hits and misses are influenced by intial cache state

Gives a sound and precise quantitative definition of predictability of caches

Predictability on the Single-Core Level: Caches

Predictability of Caches depends on their replacement policy:

- Least-Recently-Used (LRU) highly predictable, precise and efficient abstractions exist
- First-In First-Out (FIFO) inherently less predictable
- Pseudo Round-Robin as in Motorola ColdFire, extremely unpredictable

Predictability on the Single-Core Level: Pipelines

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Classification of architectures:

- *Timing compositional* (e. g., ARM7):
  No timing anomalies present,
  local worst-case behaviour safely approximates global worst-case behaviour
- *Compositional with bounded effects* (e. g., TriCore (probably)):
  Timing anomalies but no domino effects,
  local worst-case behaviour safely approximates global worst-case behaviour up to a constant, additive factor
- *Non-compositional architectures* (e. g., PPC 755):
  Timing anomalies, domino effects,
  all global paths have to be considered

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

Predictability of Multi-Core Architectures: What One Should Avoid in a Multi-Core Architecture

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- a complex architecture, not fully-compositional:
  - high complexity of the analysis
  - bound on additional delays are not constant (access to a shared resource, preemption)
- a cache with a non-LRU policy
  - less precise WCET bounds
  - more complicated computation of less precise CRPD bounds
- a unified cache
  - interferences impair precision
  - more complex analysis
- shared bus protocol with unbounded access delay
  - unbounded execution time
  - no guarantees on timing constraints
- frequently used shared resources
  - may lead to an unschedulable system
  - TDMA-like protocol: limited by idle-time

PROMPT

Minimise sharing in multi-processor architectures:

- Interferences might be huge (bus contention, cache pollution)
- Huge overestimation when analysis is possible
  - Set of tasks that might be executed in parallel
  - Cache contents

PROMPT (PRedictability Of Multi-Processor Timing)

- Start with a generic, parameterisable architecture with predictable (fully timing compositional) cores
- Instantiate architecture for given set of applications, based on their resource requirements

Design Principles

- Simplifications of components of the architecture
- Elimination of interferences on shared resources:
  - Wherever it is not absolutely needed
  - Private resources for private uses
  - Data sharing for global state
- Accesses to the shared global state
  - Determination of delays, or
  - Cumulative analyses of WCET, bus arbiter and scheduling

Conclusions

- Preemptive systems:
  - Precise analysis of cache interferences
  - Requires LRU cache and constant-bounded cache access delay
- Predictability: not a boolean property
  - Introduction of shared resources decreases predictability but may benefit efficiency
- PROMPT:
  - Predictable components of the architecture (core)
  - Reduction of interferences (when possible)
  - Determination of access delay

Further reading - resilience

Altmeyer, S. & Burguière, C. (2009).
In Proceedings of the 21st Euromicro Conference on Real-Time Systems (ECRTS '09) pp. 109–118, IEEE Computer Society.

Altmeyer, S., Maiza, C. & Reineke, J.
In LCTES'10.

Lee, C.-G., Hahn, J., Min, S. L., Ha, R., Hong, S., Park, C. Y., Lee, M. & Kim, C. S. (1996).
In RTSS'96 p. 264, IEEE Computer Society.

Negi, H. S., Mitra, T. & Roychoudhury, A. (2003).
In CODES+ISSS'03 ACM.

Reineke, J. (2008).
Caches in WCET Analysis.
PhD thesis, Universität des Saarlandes, Saarbrücken.

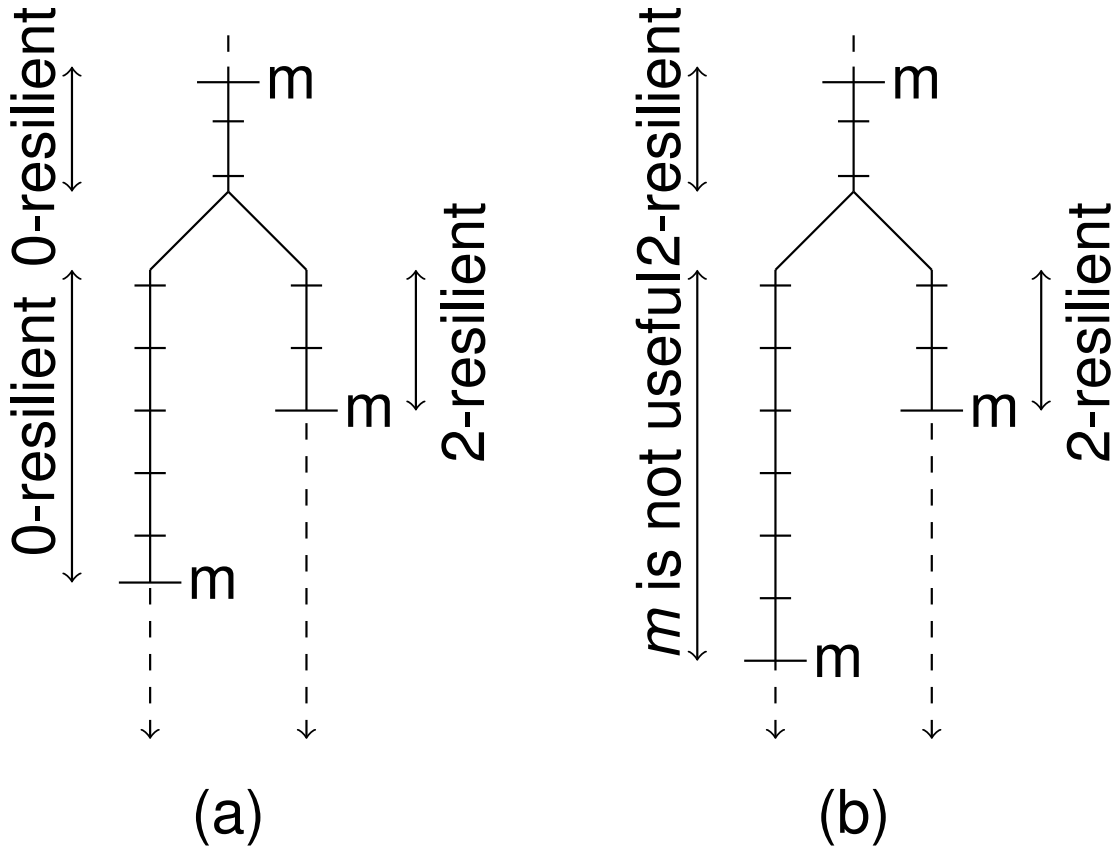
Tan, Y. & Mooney, V. (2004).
In SCOPES'04 pp. 182–199,.

Tomiyama, H. & Dutt, N. D. (2000).
In CODES'00 ACM.

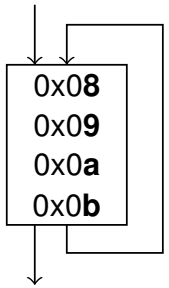
Steps of the Design Process

1. Hierarchical privatization
   - decomposition of the set of applications according to the sharing relation on the global state
   - allocation of private resources for non-shared code and state
   - sound (and precise) determination of delays for accesses to the shared global state
2. Sharing of lonely resources
   - Costly lonely resources will be shared
   - Access rate is low compared to CPU and memory bandwidth
   - Accesses happen infrequently $\Rightarrow$ access delay contributes little
3. Controlled socialization
   - introduction of additional sharing to reduce costs
   - controlling loss of predictability

/-resilience analysis



CPRD using ECB: Pitfall



$$\text{ECBs} = \{\mathbf{e}\}: \quad [b,a,9,8] \xrightarrow{8} [8,b,a,9] \xrightarrow{9} [9,8,b,a] \xrightarrow{a} [a,9,8,b] \xrightarrow{b} [b,a,9,8] \quad \text{0 misses}$$

$$[\mathbf{e},b,a,9] \xrightarrow{8^*} [8,\mathbf{e},b,a] \xrightarrow{9^*} [9,8,\mathbf{e},b] \xrightarrow{a^*} [a,9,8,\mathbf{e}] \xrightarrow{b^*} [b,a,9,8] \quad \text{4 misses}$$

- $|\text{UCB}(s)| = 4$
- $|\text{ECB}(s)| = 1$
- $n = 4$
- number of additional misses$= 4$

Upper-bound on the CRPD - direct-mapped caches

- using UCB [Lee et al., 1996]:

$$\mathrm{CRPD}_{\mathrm{UCB}} = \mathrm{BRT} \cdot |\{s_i \mid \exists m \in \mathrm{UCB} : m \bmod c = s_i\}|$$

- using ECB [Tomiyama & Dutt, 2000]:

$$\mathrm{CRPD}_{\mathrm{ECB}} = \mathrm{BRT} \cdot |\{s_i \mid \exists m \in \mathrm{ECB} : m \bmod c = s_i\}|$$

- using UCB and ECB [Negi et al., 2003, Tan & Mooney, 2004]:

$$\begin{aligned}\mathrm{CRPD}_{\mathrm{UCB\&ECB}} = \mathrm{BRT} \cdot |\{s_i \mid\ & \exists m \in \mathrm{UCB} : m \bmod c = s_i \\ & \wedge \exists m' \in \mathrm{ECB} : m' \bmod c = s_i\}|\end{aligned}$$

CRPD for FIFO: Pitfalls

SAARLAND UNIVERSITY
COMPUTER SCIENCE

ECBs $= \{\mathbf{x}\}$

$[b,a] \xrightarrow{a} [b,a] \xrightarrow{e^*} [e,b] \xrightarrow{b} [e,b] \xrightarrow{c^*} [c,e] \xrightarrow{e} [c,e]$ 2 misses

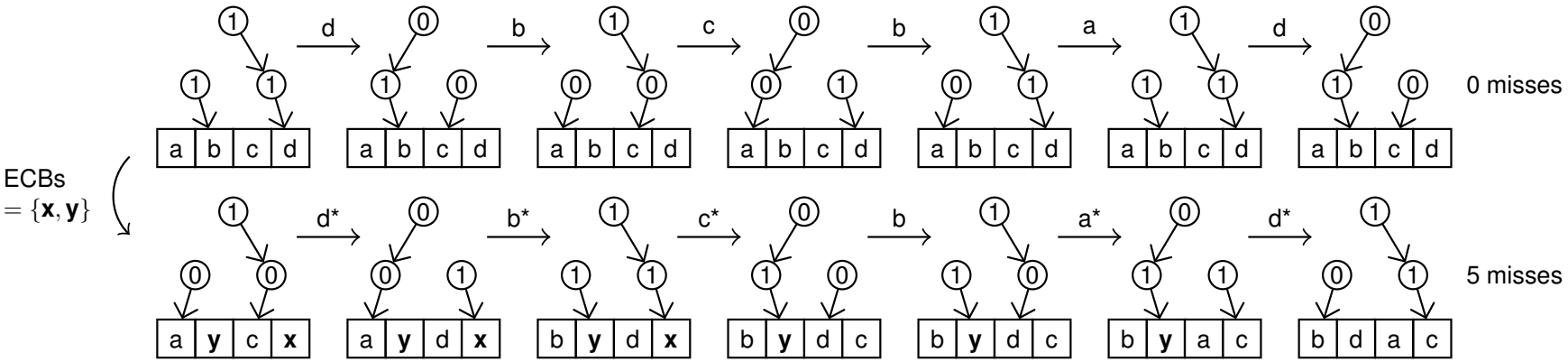
$[\mathbf{x},b] \xrightarrow{a^*} [a,\mathbf{x}] \xrightarrow{e^*} [e,a] \xrightarrow{b^*} [b,e] \xrightarrow{c^*} [c,b] \xrightarrow{e^*} [e,c]$ 5 misses

CRPD for FIFO: Pitfalls

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

- $|\text{UCB}(s)| = 2$
- $|\text{ECB}(s)| = 1$
- $n = 2$
- But: number of additional misses$= 3$

CRPD for PLRU: Pitfalls



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