

Bounding the Cache-Related Preemption Delay

Jan Reineke

October 15, 2015



- 1 Why Preemptive Scheduling?
- 2 CRPD Computation
 - Analysis of the Preempted Task
 - Analysis of the Preempting Task
 - Analysis of the Preempted and the Preempting Task
 - Simplifying or Eliminating the Problem
 - Policies other than LRU
- 3 Accounting for CRPD within Response-Time Analysis
- 4 Summary

1 Why Preemptive Scheduling?

2 CRPD Computation

- Analysis of the Preempted Task
- Analysis of the Preempting Task
- Analysis of the Preempted and the Preempting Task
- Simplifying or Eliminating the Problem
- Policies other than LRU

3 Accounting for CRPD within Response-Time Analysis

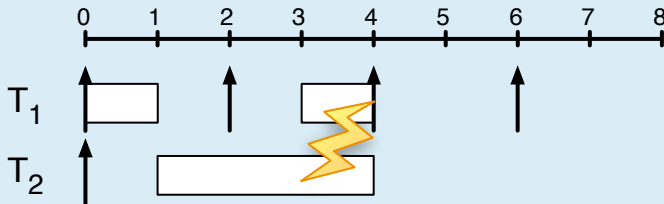
4 Summary

Why use preemptive scheduling?

- Preemption often increases schedulability of task sets.
- Tasks with short deadlines are often unschedulable non-preemptively.

Example

Given: Two periodic tasks T_1 and T_2 , with periods $P_1 = 2, P_2 = 8$, deadlines $D_1 = P_1, D_2 = P_2$, and execution times $C_1 = 1, C_2 = 3$.

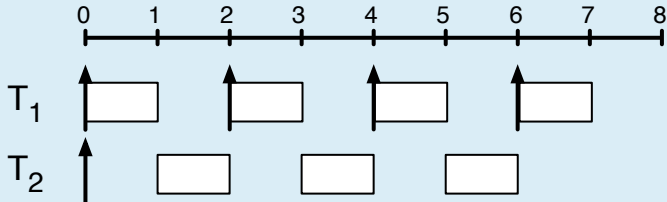


Why use preemptive scheduling?

- Preemption often increases schedulability of task sets.
- Tasks with short deadlines are often unschedulable non-preemptively.

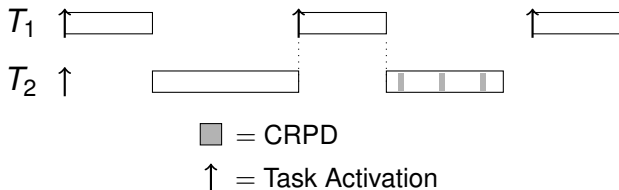
Example

Given: Two periodic tasks T_1 and T_2 , with periods $P_1 = 2, P_2 = 8$, deadlines $D_1 = P_1, D_2 = P_2$, and execution times $C_1 = 1, C_2 = 3$.



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*.



Where to account for preemption cost?

■ Integrate into WCET Analysis: [Schneider, 2000]

- ▶ Assume cache misses everywhere
- ▶ Very pessimistic but easy to use with existing schedulability analyses

■ 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 also requires new schedulability analyses taking into account the CRPD bounds

Where to account for preemption cost?

- Integrate into WCET Analysis: [Schneider, 2000]
 - ▶ Assume cache misses everywhere
 - ▶ Very pessimistic but easy to use with existing schedulability analyses
- 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 also requires new schedulability analyses taking into account the CRPD bounds

Focus of this talk: approaches to bound the CRPD

And a bit on: using these bounds within schedulability analyses

1 Why Preemptive Scheduling?

2 CRPD Computation

- Analysis of the Preempted Task
- Analysis of the Preempting Task
- Analysis of the Preempted and the Preempting Task
- Simplifying or Eliminating the Problem
- Policies other than LRU

3 Accounting for CRPD within Response-Time Analysis

4 Summary

- Preempted Task:
How many additional cache misses can a single preemption by *any* preempting task cause in a *given* preempted task?

- **Preempted Task:**
How many additional cache misses can a single preemption by *any* preempting task cause in a *given* preempted task?
- **Preempting Task:**
How many additional cache misses can a single preemption by a *given* preempting task cause in *any* preempted task?

- **Preempted Task:**
How many additional cache misses can a single preemption by *any* preempting task cause in a *given* preempted task?
- **Preempting Task:**
How many additional cache misses can a single preemption by a *given* preempting task cause in *any* preempted task?
- **Preempted + Preempting Task**
How many additional cache misses can a single preemption by a *given* preempting task cause in a *given* preempted task?

- 1 Why Preemptive Scheduling?
- 2 **CRPD Computation**
 - **Analysis of the Preempted Task**
 - Analysis of the Preempting Task
 - Analysis of the Preempted and the Preempting Task
 - Simplifying or Eliminating the Problem
 - Policies other than LRU
- 3 Accounting for CRPD within Response-Time Analysis
- 4 Summary

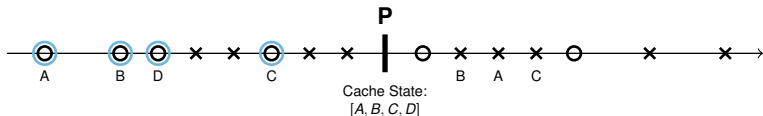
Analysis of the Preempted Task: Useful Cache Blocks (UCB)

Definition (Useful Cache Block, [Lee et al., 1996])

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

- m may be cached at P
- m may be reused at program point Q that may be reached from P with no eviction of m on this path.

x = hit
O = miss



$$UCB \supseteq \{A, B, C, D\}$$

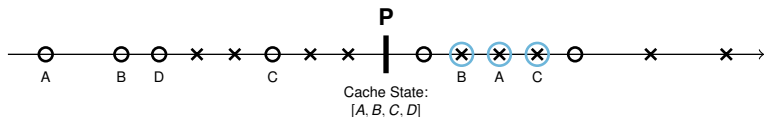
Analysis of the Preempted Task: Useful Cache Blocks (UCB)

Definition (Useful Cache Block, [Lee et al., 1996])

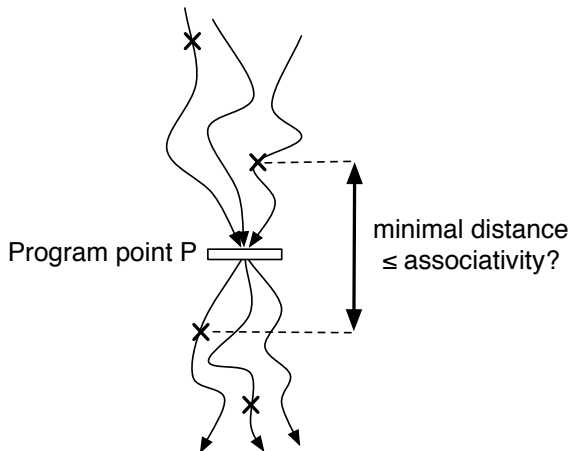
A memory block m at program point P is called a useful cache block, if

- m may be cached at P
- m may be reused at program point Q that may be reached from P with no eviction of m on this path.

x = hit
O = miss



$$\text{UCB} = \{A, B, C\}$$

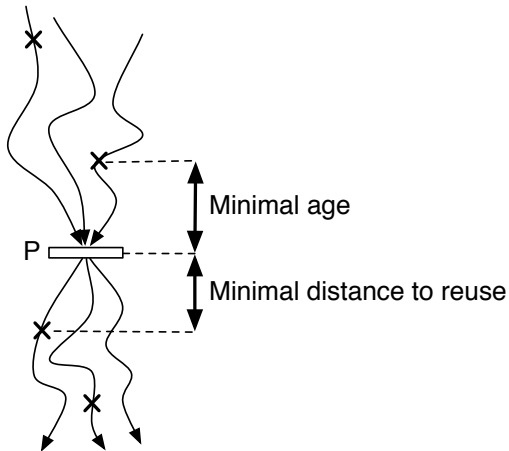


UCB Analysis

Combination of two LRU-may-analyses:

What may be cached?
Forward May-Analysis!

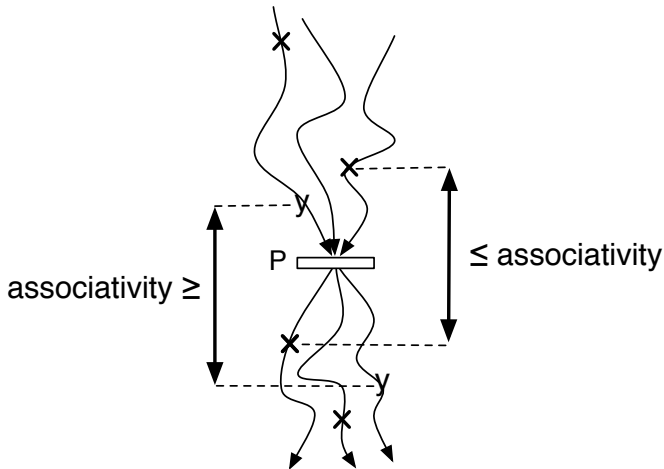
What may be reused?
Backward May-Analysis!



Minimal age + Minimal distance to reuse \leq associativity
 \implies Memory block may be useful

Improvement: Path Analysis

Some blocks are never useful at the same time:



Literature:

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

Improvement: Avoid Accumulating Overestimations

Schedulability analyses rely on:

$$WCET_{bound} + n \cdot CRPD_{bound} \geq \text{exec. time with up to } n \text{ preemptions}$$

\uparrow \nwarrow
 # of possible preemptions $BRT \cdot |UCB|$

Yet, we usually have:

$$WCET_{bound} \geq \text{execution time without preemptions}$$

$$CRPD_{bound} \geq \text{additional execution time due to one preemption}$$

⇒ Overestimation in both analyses adds up:
Some cache misses are counted twice!

Bounding the CRPD using UCBs for Fully-Associative Caches

- CRPD bound at program point P :

$$\text{CRPD}_{\text{UCB}}^{\text{LRU}}(P) = \text{BRT} \cdot \min(|\text{UCB}(P)|, k),$$

where $k = \text{associativity}$ and $\text{BRT} = \text{Block Reload Time}$

- CRPD bound independent of program point:

$$\text{CRPD}_{\text{UCB}}^{\text{LRU}} = \max_P \text{CRPD}_{\text{UCB}}^{\text{LRU}}(P)$$

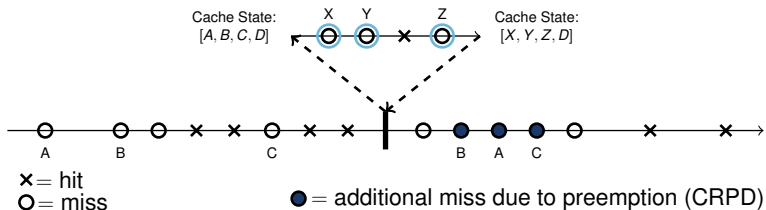
Slightly more complicated for set-associative caches:
sum up bounds of all cache sets.

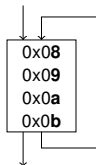
- 1 Why Preemptive Scheduling?
- 2 **CRPD Computation**
 - Analysis of the Preempted Task
 - **Analysis of the Preempting Task**
 - Analysis of the Preempted and the Preempting Task
 - Simplifying or Eliminating the Problem
 - Policies other than LRU
- 3 Accounting for CRPD within Response-Time Analysis
- 4 Summary

Analysis of the Preempting Task: Evicting Cache Blocks

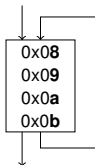
Definition (Evicting Cache Blocks (ECB),
[Tomiya and Dutt, 2000])

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.





$[b, a, 9, 8] \xrightarrow{8} [8, b, a, 9] \xrightarrow{9} [9, 8, b, a] \xrightarrow{a} [a, 9, 8, b] \xrightarrow{b} [b, a, 9, 8] \quad 0 \text{ misses}$



$$\begin{array}{l}
 \text{ECBs} \\
 = \{e\}
 \end{array}
 \left(\begin{array}{l}
 [b, a, 9, 8] \xrightarrow{8} [8, b, a, 9] \xrightarrow{9} [9, 8, b, a] \xrightarrow{a} [a, 9, 8, b] \xrightarrow{b} [b, a, 9, 8] \quad 0 \text{ misses} \\
 [e, b, a, 9] \xrightarrow{8^*} [8, e, b, a] \xrightarrow{9^*} [9, 8, e, b] \xrightarrow{a^*} [a, 9, 8, e] \xrightarrow{b^*} [b, a, 9, 8] \quad 4 \text{ misses}
 \end{array} \right)$$

- $|UCB| = 4$
- $|ECB| = 1$
- $k = \text{associativity} = 4$
- **number of additional misses = 4**

CRPD Computation for LRU using ECB: Sound but Imprecise

- ECB analysis only to determine whether the set is used at all by the preempting task or not:

$$\text{CRPD}_{\text{ECB}}^{\text{LRU}} = \begin{cases} 0 & \text{if } \text{ECB} = \emptyset \\ \text{BRT} \cdot k & \text{otherwise} \end{cases}$$

- Cannot do better than that without knowledge of preempted task.

- 1 Why Preemptive Scheduling?
- 2 CRPD Computation**
 - Analysis of the Preempted Task
 - Analysis of the Preempting Task
 - Analysis of the Preempted and the Preempting Task**
 - Simplifying or Eliminating the Problem
 - Policies other than LRU
- 3 Accounting for CRPD within Response-Time Analysis
- 4 Summary

Analysis of Preempted and Preempting Task: “Shallow” Combination

Take the minimum of the *UCB*- and *ECB*-based estimations.

- CRPD bound for entire program:

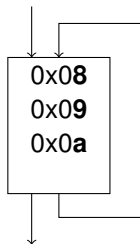
$$\text{CRPD}_{\text{UCB}+\text{ECB}}^{\text{LRU}} = \min(\text{CRPD}_{\text{ECB}}^{\text{LRU}}, \text{CRPD}_{\text{UCB}}^{\text{LRU}})$$

Becomes slightly more complicated for set-associative caches:
For a program point: sum of point-wise minima of all cache sets.

Literature:

- For direct-mapped caches: [Negi et al., 2003]
- For set-associative caches:
[Tan and Mooney, 2004, Burguière et al., 2009]

Analysis of Preempted and Preempting Task: “Deeper” Combination



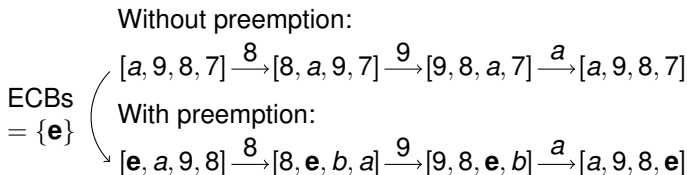
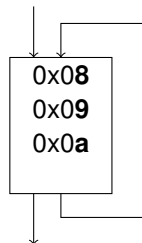
Without preemption:
 $[a, 9, 8, 7] \xrightarrow{8} [8, a, 9, 7] \xrightarrow{9} [9, 8, a, 7] \xrightarrow{a} [a, 9, 8, 7]$

ECBs = {**e**}

With preemption:
 $[e, a, 9, 8] \xrightarrow{8} [8, e, b, a] \xrightarrow{9} [9, 8, e, b] \xrightarrow{a} [a, 9, 8, e]$

Some of the UCBs are guaranteed to *remain useful* under preemption!

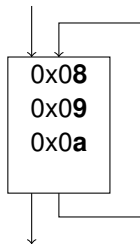
Analysis of Preempted and Preempting Task: “Deeper” Combination



Some of the UCBs are guaranteed to *remain useful* under preemption!

- $\text{CRPD}_{\text{UCB\&ECB}} = \min(\text{CRPD}_{\text{UCB}}, \text{CRPD}_{\text{ECB}}) = \min(3, 4) = 3$
- **Yet: actual number of additional misses: 0**

Analysis of Preempted and Preempting Task: “Deeper” Combination



ECBs
= {e}

Without preemption:

$[a, 9, 8, 7] \xrightarrow{8} [8, a, 9, 7] \xrightarrow{9} [9, 8, a, 7] \xrightarrow{a} [a, 9, 8, 7]$

With preemption:

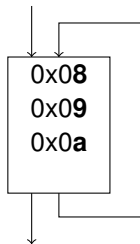
$[e, a, 9, 8] \xrightarrow{8} [8, e, b, a] \xrightarrow{9} [9, 8, e, b] \xrightarrow{a} [a, 9, 8, e]$

Some of the UCBs are guaranteed to *remain useful* under preemption!

- $CRPD_{UCB\&ECB} = \min(CRPD_{UCB}, CRPD_{ECB}) = \min(3, 4) = 3$
- **Yet: actual number of additional misses: 0**

Why?

Analysis of Preempted and Preempting Task: “Deeper” Combination



ECBs
= {e}

Without preemption:

$[a, 9, 8, 7] \xrightarrow{8} [8, a, 9, 7] \xrightarrow{9} [9, 8, a, 7] \xrightarrow{a} [a, 9, 8, 7]$

With preemption:

$[e, a, 9, 8] \xrightarrow{8} [8, e, b, a] \xrightarrow{9} [9, 8, e, b] \xrightarrow{a} [a, 9, 8, e]$

Some of the UCBs are guaranteed to *remain useful* under preemption!

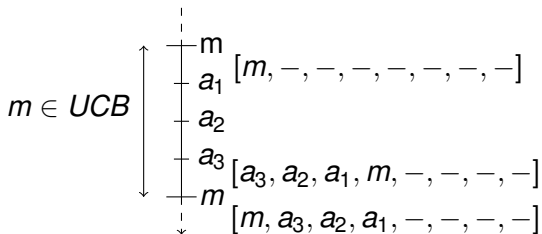
- $CRPD_{UCB\&ECB} = \min(CRPD_{UCB}, CRPD_{ECB}) = \min(3, 4) = 3$
- **Yet: actual number of additional misses: 0**

Why?

- Minimal number of ECBs to evict a UCB is 2, but $|ECB| = 1$
- **A single ECB is not sufficient to evict any of the UCBs.**

Combining UCB and ECB: Notion of Resilience

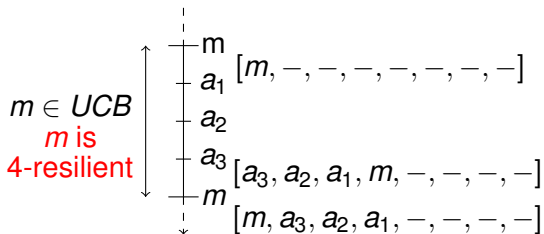
Determining the maximal number of ECBs, such that no additional cache miss may occur:



Combining UCB and ECB:

Notion of Resilience

Determining the maximal number of ECBs, such that no additional cache miss may occur:



Definition (Resilience)

A memory block m is called *l -resilient* at program point P , if all possible next accesses to m

- that would be hits without preemption,
- would still be hits in case of a preemption at P with l accesses.

Definition (Resilience)

A memory block m is called l -resilient at program point P , if all possible next accesses to m

- that would be hits without preemption,
- would still be hits in case of a preemption at P with l accesses.

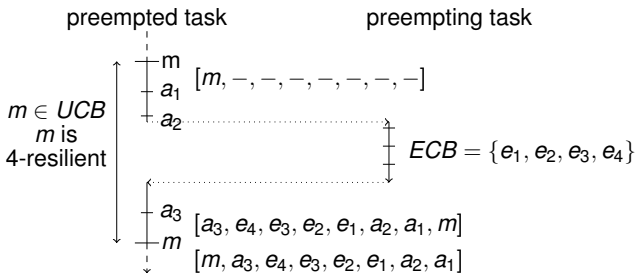
- No UCB is k -resilient, i.e., no UCB remains useful after a preemption with k (= associativity) many ECBs.
- Each $(l + 1)$ -resilient UCB is also l -resilient.
- Each UCB is at least 0-resilient.

Resilience Analysis

Definition (Resilience)

A memory block m is called l -resilient at program point P , if all possible next accesses to m

- that would be hits without preemption,
- would still be hits in case of a preemption at P with l accesses.

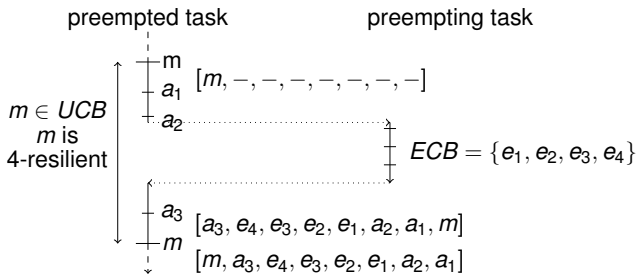


Resilience Analysis

Definition (Resilience)

A memory block m is called l -resilient at program point P , if all possible next accesses to m

- that would be hits without preemption,
- would still be hits in case of a preemption at P with l accesses.



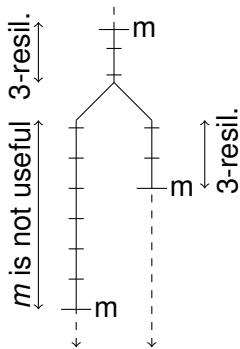
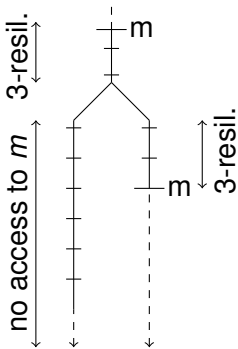
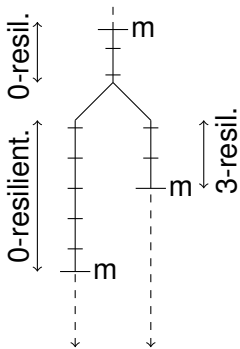
In general: if $|ECB| \leq l$ then the UCB is not evicted

Resilience Analysis

Definition (Resilience)

A memory block m is called l -resilient at program point P , if all possible next accesses to m

- that would be hits without preemption,
- would still be hits in case of a preemption at P with l accesses.



CRPD (Combining UCB and ECB by using Resilience)

$$\underbrace{UCB}_{\text{useful}} \setminus \underbrace{\{m \mid m \text{ is } |ECB|\text{-resilient}\}}_{\text{remain useful}}$$

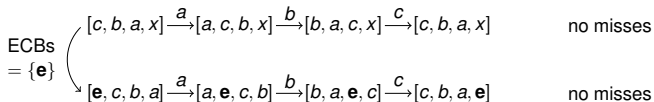
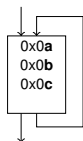
blocks contributing to CRPD

CRPD (Combining UCB and ECB by using Resilience)

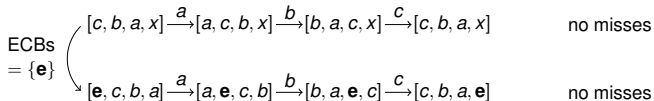
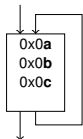
$$CRPD \leq BRT \times \left| \underbrace{UCB}_{\text{useful}} \setminus \underbrace{\{m \mid m \text{ is } |ECB|\text{-resilient}\}}_{\text{remain useful}} \right|$$

blocks contributing to CRPD

Bounding the CRPD using Resilience: Example

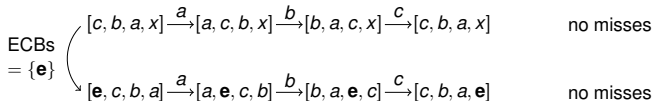
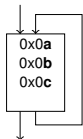


Bounding the CRPD using Resilience: Example



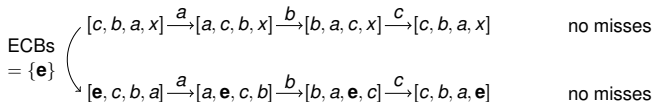
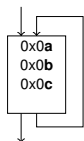
- - ▶ $|ECB| = 1$
 - ▶ $a, b,$ and c are 1-resilient
 - ▶ $CRPD_{UCB\&ECB}^{res} = BRT \times |UCB \setminus \{m \mid m \text{ is } |ECB|\text{-resilient}\}| = 0$

Bounding the CRPD using Resilience: Example



- - ▶ $|ECB| = 1$
 - ▶ $a, b,$ and c are 1-resilient
 - ▶ $CRPD_{UCB\&ECB}^{res} = BRT \times |UCB \setminus \{m \mid m \text{ is } |ECB|\text{-resilient}\}| = 0$
- Instead of: $CRPD_{UCB\&ECB} = \min(CRPD_{UCB}, CRPD_{ECB}) = 3 \times BRT$

Bounding the CRPD using Resilience: Example



- - ▶ $|ECB| = 1$
 - ▶ $a, b,$ and c are 1-resilient
 - ▶ $CRPD_{UCB\&ECB}^{res} = BRT \times |UCB \setminus \{m \mid m \text{ is } |ECB|\text{-resilient}\}| = 0$
- Instead of: $CRPD_{UCB\&ECB} = \min(CRPD_{UCB}, CRPD_{ECB}) = 3 \times BRT$

1 Why Preemptive Scheduling?

2 CRPD Computation

- Analysis of the Preempted Task
- Analysis of the Preempting Task
- Analysis of the Preempted and the Preempting Task
- **Simplifying or Eliminating the Problem**
- Policies other than LRU

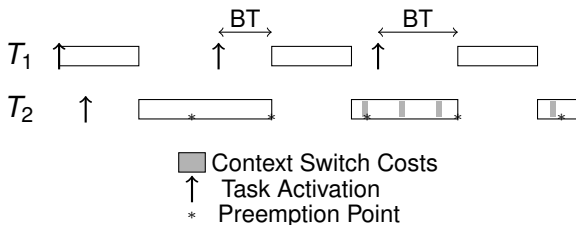
3 Accounting for CRPD within Response-Time Analysis

4 Summary

Deferred Preemption

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

Intervals between preemption points \equiv critical sections.



Where to place preemptions points, s.t.

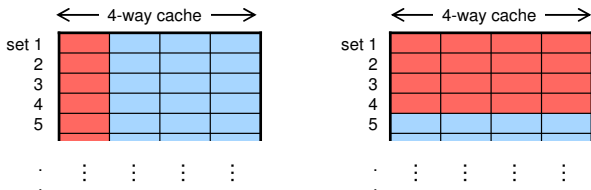
- CRPD is minimized, and
- *Maximum Blocking Time* is minimized?

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

Cache Partitioning

Additional cache misses are due to interference on the cache.

⇒ 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

1 Why Preemptive Scheduling?

2 CRPD Computation

- Analysis of the Preempted Task
- Analysis of the Preempting Task
- Analysis of the Preempted and the Preempting Task
- Simplifying or Eliminating the Problem
- Policies other than LRU

3 Accounting for CRPD within Response-Time Analysis

4 Summary

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]:

$$\begin{array}{l}
 \text{ECBs} \\
 = \{\mathbf{x}\}
 \end{array}
 \left(
 \begin{array}{l}
 [b, a] \xrightarrow{a} [b, a] \xrightarrow{e^*} [e, b] \xrightarrow{b} [e, b] \xrightarrow{c^*} [c, e] \xrightarrow{e} [c, e] \quad 2 \text{ misses} \\
 [x, b] \xrightarrow{a^*} [a, x] \xrightarrow{e^*} [e, a] \xrightarrow{b^*} [b, e] \xrightarrow{c^*} [c, b] \xrightarrow{e^*} [e, c] \quad 5 \text{ misses}
 \end{array}
 \right)$$

- $|\text{UCB}(s)| = 2$
- $|\text{ECB}(s)| = 1$
- associativity $k = 2$
- **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).

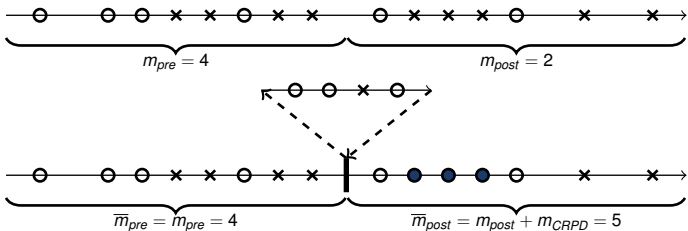
\implies Performing WCET and CRPD analyses assuming LRU($1 + \log_2 n$) replacement should give correct bounds for PLRU(n).

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

Applying Relative Competitiveness: A sequence of memory accesses

■ Notation:

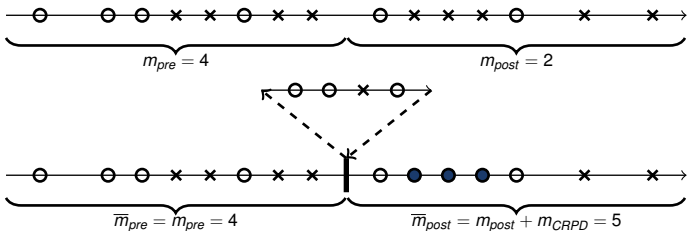
- ▶ m = number of misses
- ▶ \bar{m} = number of misses in the case of preemption



Applying Relative Competitiveness: A sequence of memory accesses

■ Notation:

- ▶ m = number of misses
- ▶ \bar{m} = number of misses in the case of preemption



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

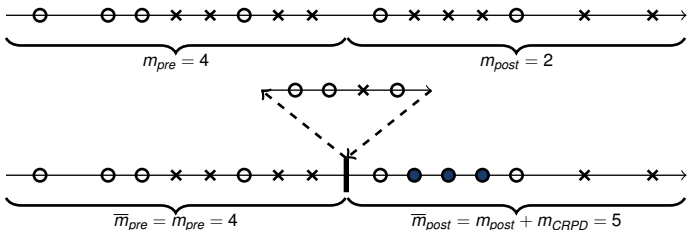
$$\bar{m}^{P(t)} = \bar{m}_{pre}^{P(t)} + \bar{m}_{post}^{P(t)}$$

Applying Relative Competitiveness:

A sequence of memory accesses

■ Notation:

- ▶ m = number of misses
- ▶ \bar{m} = number of misses in the case of preemption



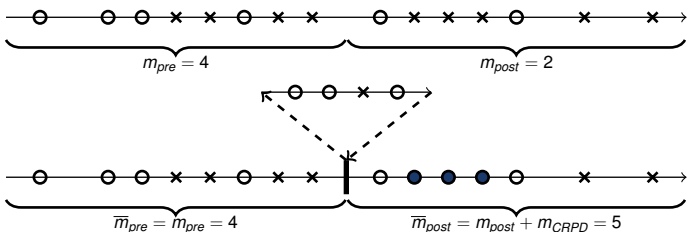
- Assume $P(t)$ is (k, c) -miss-competitive rel. to $LRU(s)$. Then:

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

Applying Relative Competitiveness: A sequence of memory accesses

■ Notation:

- ▶ m = number of misses
- ▶ \bar{m} = number of misses in the case of preemption



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

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

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

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

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

- 1 Why Preemptive Scheduling?
- 2 CRPD Computation
 - Analysis of the Preempted Task
 - Analysis of the Preempting Task
 - Analysis of the Preempted and the Preempting Task
 - Simplifying or Eliminating the Problem
 - Policies other than LRU
- 3 Accounting for CRPD within Response-Time Analysis
- 4 Summary

Recap: Rate-Monotonic Schedulability Condition

The time-demand function $W_i(t)$ of the task τ_i is defined as follows:

$$W_i(t) = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j.$$

Theorem

A system \mathcal{T} of periodic, independent, preemptable tasks is schedulable on one processor by algorithm A if it holds that:

$$\forall \tau_i \in \mathcal{T} \exists t \text{ with } 0 < t \leq D_i \text{ and } W_i(t) \leq t$$

This condition is also necessary for synchronous, periodic task sets and also sporadic task sets.

Note that this holds for implicit-deadline and constrained-deadline task sets.

Recap: Response-Time Analysis for RM

Want to determine whether there is a t such that $W_i(t) \leq t \leq D_i$.

Compute smallest such t (if there is any) by fixed-point iteration:

$n := 0$

$R_i^0 := C_i$

do

$n := n + 1$

$R_i^n := C_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_j^{n-1}}{T_j} \right\rceil C_j$

while $R_i^n < D_i$ and $R_i^{n-1} < R_i^n$

A set of tasks is schedulable if $R_i^n < D_i$ for every task i .

How to incorporate CRPD?

Introduce additional term $\gamma_{i,j}$ in the time-demand function:

$$W_i(t) = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil (C_j + \gamma_{i,j}).$$

Fixed-point iteration can be appropriately adapted.

How to incorporate CRPD?

Introduce additional term $\gamma_{i,j}$ in the time-demand function:

$$W_i(t) = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil (C_j + \gamma_{i,j}).$$

Fixed-point iteration can be appropriately adapted.

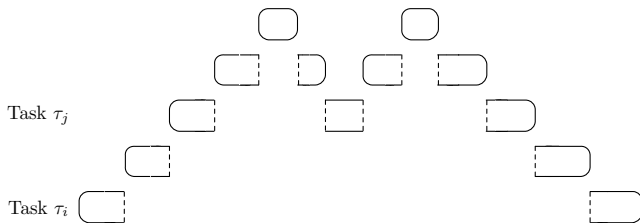
What is the meaning of $\gamma_{i,j}$?

Which value should it take?

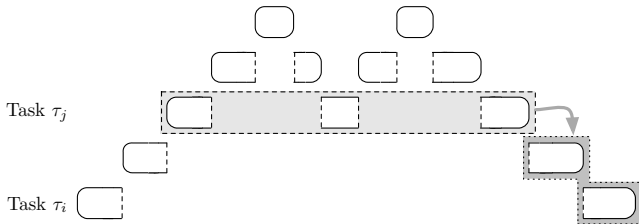
Without nested preemptions:

$\gamma_{i,j} = \text{CRPD}$ due to one preemption of task i by task j .

But what if there are nested preemptions?



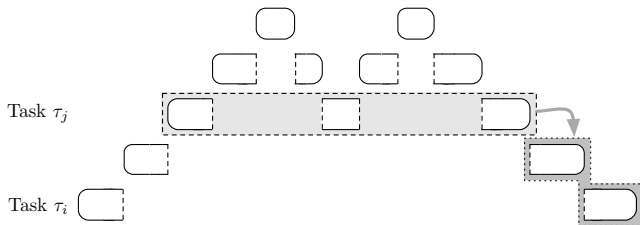
First Interpretation of $\gamma_{i,j}$: “Effect of preempting task”



Under this interpretation, $\gamma_{i,j}$ needs to bound the effect of j 's execution on all tasks of lower priority up to task i .

First Interpretation of $\gamma_{i,j}$:

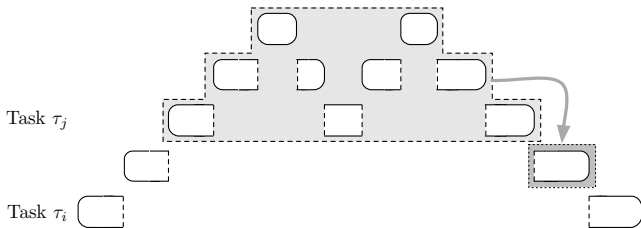
“Effect of preempting task”



For *direct-mapped caches* there are three such approaches:

- 1 “ECB-Only”: $\gamma_{i,j} = BRT \cdot |ECB_j|$
- 2 “UCB-Union”: $\gamma_{i,j} = BRT \cdot \left| \bigcup_{k \in \text{aff}(i,j)} UCB_k \right|$, where
 $\text{aff}(i,j) = \text{hep}(i) \cap \text{lp}(j)$.
- 3 Combination of the above: $\gamma_{i,j} = BRT \cdot \left| \bigcup_{k \in \text{aff}(i,j)} UCB_k \cap ECB_j \right|$

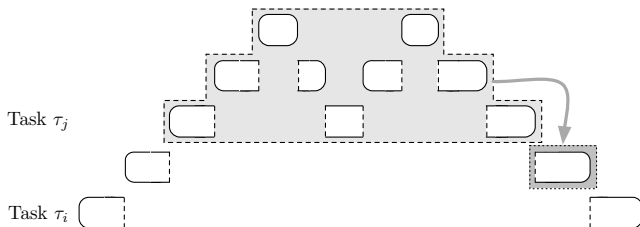
Second Interpretation of $\gamma_{i,j}$: “Effect of immediately preempted task”



Under this interpretation, $\gamma_{i,j}$ needs to bound the effect of j 's execution and that of the execution of task's preempting j itself on the task that j immediately preempted.

Second Interpretation of $\gamma_{i,j}$:

“Effect of immediately preempted task”



For *direct-mapped caches* there are three such approaches:

- 1 “UCB-Only”: $\gamma_{i,j} = BRT \cdot \max_{k \in \text{aff}(i,j)} |UCB_k|$
- 2 “ECB-Union”: $\gamma_{i,j} = BRT \cdot \left| \bigcup_{h \in \text{hp}(j) \cup \{j\}} ECB_h \right|$.
- 3 Combination of the above:

$$\gamma_{i,j} = BRT \cdot \max_{k \in \text{aff}(i,j)} \left| UCB_k \cap \bigcup_{h \in \text{hp}(j) \cup \{j\}} ECB_h \right|$$

- Soundly and precisely handle set-associative caches in response-time analysis and use *resilience* information.
- Empirical analysis of sources of imprecision: is overestimation due to characterization of preempting tasks, of preempted task, or other imprecisions?
- What is a *sound penalty* for a preemption-induced cache reload?
 - main memory latency is *not*, even for simple in-order pipelines
 - on the other hand, often part of the latency can be hidden

- 1 Why Preemptive Scheduling?
- 2 CRPD Computation
 - Analysis of the Preempted Task
 - Analysis of the Preempting Task
 - Analysis of the Preempted and the Preempting Task
 - Simplifying or Eliminating the Problem
 - Policies other than LRU
- 3 Accounting for CRPD within Response-Time Analysis
- 4 Summary

- Preemptive Scheduling desirable, but not for free:
 - Need to bound CRPD
- For LRU, the CRPD can be bounded by analyzing
 - ▶ the preempted task: UCB analysis
 - ▶ the preempting task: ECB analysis
 - ★ Sound approach rather imprecise
 - ★ Need to couple more tightly with analysis of preempted task
 - ▶ both, the preempted and the preempting task
 - ★ “Shallow” combination
 - ★ “Deeper” combination: Resilience analysis
- Approaches do not carry over to FIFO, PLRU, etc. immediately
 - ▶ First approach: relative competitiveness
- Can be accounted for in response-time analysis in different ways
 - ▶ Has so far only been investigated in detail for direct-mapped caches



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.



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).



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



Staschulat, J. and Ernst, R. (2007).

Scalable precision cache analysis for real-time software.

Trans. on Embedded Computing Sys., 6(4):25.



Tan, Y. and Mooney, V. (2004).

Integrated intra- and inter-task cache analysis for preemptive multi-tasking real-time systems.

In *Proceedings of the 8th International Workshop SCOPES 2004*, in: *Lecture Notes on Computer Science, LNCS3199*, pages 182–199. Press.



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.