Bounding the Effects of Resource Access Protocols on Cache Behavior

Enrico Mezzetti, Marco Panunzio, Tullio Vardanega

University of Padua, Italy

Brussels, July 6, 2010
# Table of contents

1. Introduction

2. Bounds on the CRBD
   - Priority Inheritance Protocol
   - Priority Ceiling Protocol
   - Immediate Ceiling Priority Protocol

3. Conclusion
Task independence assumption

- **Simplifying assumption of independence between tasks**
  - Assumed by most schedulability analysis techniques
    - Constant (and negligible) context-switch costs

- **Broken by reality (e.g., HW acceleration features)**
  - Shared caches and complex pipelines
  - Inter-task interference effects on context switch cost
    - If HW timing is sensitive to execution history then interrupt handling and preemption may influence the execution time of preempted task
  - Even more prominent with the advent of multicore systems
Cache-Related Preemption Delay

- **Cache-aware schedulability analysis techniques**
  - Preempted task may incur additional cache misses
  - Useful cache contents may be evicted by the preempting tasks

![Diagram showing cache behavior with preempted tasks](diagram.png)
Cache-Related Preemption Delay

- **Cache-aware schedulability analysis techniques**
  - Preempted task may incur additional cache misses
  - Useful cache contents may be evicted by the preemting tasks

- **Refill penalty (CRPD)**
  - Depends on both preempted and preemting task
    - *Useful Cache Blocks (UCB)*
    - *Used Cache Blocks (UCB)*
  - Upper bound included in the *response time* of tasks
Shared Resource Access

- **Priority Inversion**
  - Higher-priority task $\tau_i$ may be **blocked** by lower-priority task $\tau_j$
  - Owing to the need to serialize access to shared resource
    - Potentially *unbounded* duration with risk of *deadlock*

- **Resource access protocols**
  - Bounded priority inversion and (possibly) deadlock avoidance
  - According to each protocol
    - Rich taxonomy (*direct*, *inheritance* and *avoidance* blocking)
    - Bounds on the number of blocking events and duration
Cache-Related Blocking Delay

**CRBD**
- Lower-priority task $\tau_j$ may evict useful cache blocks of higher-priority task $\tau_i$
  - Similar to preemption but in the opposite direction
- $\tau_i$ may incur a CRBD because of the additional cache misses
  - Blocking events may accumulate during the same activation

**CRBD vs. CRPD**
- Could be transformed into a CRPD problem
  - Critical sections → tasks that may preempt higher priority tasks
- CRBD as function of $UCB$ of $\tau_i$ and $\overline{UCB}$ of $\tau_j$ but
  - $\overline{UCB}$ limited to execution of $\tau_j$ inside critical sections
  - $UCB$ computed with respect to predefined execution points
    (for direct and avoidance blocking)
  - Transitivity
CRBD Computation

- **Classical $UCB$ and $\overline{UCB}$ sets**
  - For each task $\tau_i$ at node $n \in CFG(i)$
    - $UCB_i^n = \text{ReachingBlocks}_i(n) \cap \text{LiveBlocks}_i(n)$
    - $\overline{UCB}_i^n = \text{ReachingBlocks}_i(n)$.

- **Assumptions**
  - Total ordering between tasks: $i < j$ if $\pi(\tau_i) > \pi(\tau_j)$
  - $\tau_i$ may access a shared resource $R \in SR_i \subseteq SR_{System}$
    - $cs_{i,k}^R = k^{th}$ critical section in $\tau_i$ accessing $R$
  - Shared resources properly nested (can never overlap)

- **CRBD computation**
  - Depends on the actual type of blocking incurred
CRBD Computation (cont’d)

- **Example of $\overline{UCB}$ and $UCB$ for direct blocking**
  - High priority task $\tau_i$ is directly blocked trying to access $cs_{i,k}^R$
  - Lower-priority task $\tau_j$ is executing inside a critical section $cs_{j,h}^R$

  $UCB_{i,k}^R = UCB_i^{nR}$ where $n_R$ is the entry node of $cs_{i,k}^R$

  $\overline{UCB}_j(cs_{j,h}^R) = ReachingBlocks_j([first\_node, last\_node]_{cs_{j,h}^R})$

- **Computation of CRBD**

  \[
  CRBD = \otimes_\sigma (UCB_{i,k}^R, \overline{UCB}_j(cs_{j,h}^R)) \times \text{miss penalty}
  \]

- Where $\otimes_\sigma$ combines the information on $UCB$s and $\overline{UCB}$s
  - According to actual cache associativity and replacement policy

- **Bounds on the CRBD**

  - Leveraging on bounds warranted by resource access protocols
    - Bounds on blocking events $\triangleright$ bounds on cache interference
CRBD under PIP

- **Priority Inheritance**
  - A task inherits the priority of the highest-priority task it is blocking
  - Lowered to the highest inherited priority value upon release
    - Bounded priority inversion
    - Does not prevent deadlocks
    - Direct and inheritance blocking

- **Bound on blocking events**
  - Given $\beta_{i,j}^*$ set of *outermost* critical sections of $\tau_j$ that can block $\tau_i$
  - $\tau_i$ can be blocked by $\tau_j$ for at most the duration of one $cs \in \beta_{i,j}^*$
    - By either direct or inheritance blocking
  - Computing UCB of $\tau_i$ in case of inheritance blocking
    - Consider any possible node in $CFG(\tau_i)$ ($\sim$ CRPD)
CRBD under PIP (cont’d)

- **CRBD bound**
  - Direct blocking
    \[
    CRBD_{i,j}^{base} \leq \max_{R \in SR_i, k \in [1, |cs_i^R|]} \{ \otimes_{\sigma} (UCB_{i,k}^R, \overline{UCB}_j(cs)) \} \times \text{miss penalty}
    \]
  - Inheritance blocking
    \[
    CRBD_{i,j}^{inherit} \leq \max_{cs \in \hat{\beta}_{i,j}} \{ \otimes_{\sigma} (UCB_{i}^n, \overline{UCB}_j(cs)) \} \times \text{miss penalty}
    \]
    \[
    \hat{\beta}_{i,j} = \{ cs | cs \in \beta_{i,j}^* \land cs \text{ can block } \tau_i \text{ by inheritance blocking} \}
    \]

- Then CRBD possibly incurred by \( \tau_i \):

\[
CRBD_i \leq \sum_{j > i} \max (CRBD_{i,j}^{base}, CRBD_{i,j}^{inherit})
\]

- PIP also bounds the number of blocking semaphores
CRBD under PCP

- **Priority Ceiling**
  - Each resource $R$ is statically assigned a *ceiling priority* $\text{ceil}(R)$
  - $\tau_i$ can access $R$ if $\pi(i) > \text{ceil}(S)$ $\forall S \in SR$ currently locked
  - Otherwise the task that blocks $\tau_i$ inherits the ceiling priority of the resource it is locking
    - Bounded priority inversion
    - Prevents deadlock
    - Avoids transitive blocking
    - Introduces avoidance blocking

- **CRBD bound**
  - Exploits the $\beta_{i,j}^*$ and $\hat{\beta}_{i,j}$ sets defined for PIP
  - Task $\tau_i$ can be blocked at most once per activation
    - By either direct, inheritance or avoidance blocking

$$\text{CRBD}_i \leq \max_{j > i} \{ \max (\text{CRBD}_{i,j}^{\text{base}}, \text{CRBD}_{i,j}^{\text{inherit}}) \}$$
CRBD under ICPP

- **Immediate Ceiling Priority**
  - *Ceiling priorities* are statically assigned as in PCP
  - A task always inherits the ceiling priority of the resource it is locking
  - All tasks with priority lower than or equal to the ceiling priority cannot be scheduled until the resource has been released
    - Bounded priority inversion
    - Prevents deadlock
    - Avoids transitive blocking
    - Adds avoidance blocking

- **Bound on blocking events**
  - Task $\tau_i$ can be blocked at most once per activation
    - By either direct, inheritance or avoidance blocking
  - If blocking occurs, it is always before execution
    - No CRBD
Conclusion

- **Included in RTA iterative equation**

\[ w_{i}^{n+1} = C_i + B_i + \beta_i + \sum_{j \in hp(i)} \left\lfloor \frac{w_{i}^{n}}{T_j} \right\rfloor \times (C_j + \gamma_j) \]

- However worst-case \( B_i \) and \( \beta_i \) not necessarily occur altogether
- Advanced approaches to CRPD like *Resilience Analysis*
  - Would require a combined computation of \( \gamma_i \) and \( \beta_i \)

- **CRBD bounds**
  - The actual CRBD effect may be small
    - Simple test ➤ 8 out of 38 misses due to direct blocking
    ➤ 3 out of 12 misses due to inheritance blocking
  - Still important for schedulability analysis that seeks accuracy
  - CRBD as a selection criterion for resource access protocol
    - The use of ICPP is free from CRBD