Interactions between WCET analysis and scheduling

Guillaume Phavorin and Pascal Richard

LIAS/Université de Poitiers

April 4th, 2014
Introduction

Context
- Into the WCET
- Into the scheduling analysis
- Reducing CRPD

Contributions
- Problematic
- Problem 1: CRPD-aware scheduling
- Problem 2: Cache-aware scheduling

Conclusion
Guillaume Phavorin and Pascal Richard

Interactions between WCET analysis and scheduling

Introduction

Context

Contributions

Conclusion

CPU

- core
- registers: 1 cycle / compiler
- instr. cache: 3 cycles / hardware
- data cache
- L2 cache: 15 cycles / hardware
- memory bus
- main memory: 200 cycles / OS

Moore's law effect

- CPU performance: 60%/yr
- Memory performance: 7%/yr

Gap grows at 50% per year
Cache

Small and fast memory (compared to the main memory).
→ to bridge the gap between the processor speed and the main memory access time.
→ by storing:
  - data that is frequently accessed (temporal locality),
  - data that will (or may) be accessed next (spatial locality).

Instruction vs data caches, shared cache, cache hierarchy...

When a block is accessed:
  - in cache: cache hit → low cost (≈ 1 to 4 clock cycles),
  - not in cache: cache miss → high cost (≈ 8 to 32 cycles).
Cache organization

Cache:

- divided into **cache lines** of equal size:
  - number of contiguous bytes transferred from the main memory to the cache.

- that may be grouped into sets:
  - **direct-mapped**: 1 line = 1 set
    - a memory block can be mapped to only **one line**.
  - **fully-associative**: only one set containing all lines
    - a memory block can be mapped **everywhere** in the cache.
  - **set-associative**: lines equally divided into several sets
    - a memory block can be mapped only to **one set** BUT everywhere in it.

Eg: 8kB direct-mapped instruction cache with a 8 bytes line size and a 4 bytes instruction size (ARM7).
Replacement policy

**Offline:**
Belady’s rule: the block whose next request is the furthest in the future is evicted. ⇒ OPTIMAL.

**Online:**
No optimal policy, as the access sequence is not known.

- LRU: Least Recently Used.

Example with a 4-way associative cache set

```
<table>
<thead>
<tr>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>b</td>
</tr>
<tr>
<td>c</td>
</tr>
<tr>
<td>d</td>
</tr>
</tbody>
</table>
```

- acess to e: cache miss

```
| e   |
| a   |
| b   |
| c   |
```

- acess to b: cache hit

```
| b   |
| e   |
| a   |
| c   |
```
Cache-Related Preemption Delay (CRPD)

\[ \tau_1 \]

\[ \tau_2 \]

adding preemption costs

Guillaume Phavorin and Pascal Richard

Interactions between WCET analysis and scheduling
Classical approaches

- Platform features (processor, cache...)
- Task code
- Task periods and deadlines
- Scheduler
- Task priorities

Timing Analysis

Task WCET

Schedulability Analysis

Yes/No

Guillaume Phavorin and Pascal Richard
Goal:

- WCET accounting for all potential preemption delays.
- Preemption costs have no longer to be considered into the scheduling analysis.

![Diagram]

**Timing Analysis** → **WCET** → **Scheduling Analysis**
Easiest way to incorporate preemption delays into the WCET:

- **every access** → considered to be a **cache miss** (as if the cache was disabled)

But very pessimistic, and cache benefits are not taking any more into account.
Magic WCET: Approach 2

- taking cache benefits into account → tighter WCET,
- upper-bounding cache effects (CRPD: Cache-Related Preemption Delays) → to achieve predictability.

→ by conducting cache analyses:

1. for WCET: representation of cache contents to identify accesses that will be "Always Hits".
2. to bound the impact of a preemption at a given program point.

\[
\text{WCET}_{w/o \ preemption} + n \cdot \text{CRPD}
\]

Problem: how to get \( n \)? → very dependant on the chosen scheduling policy and the considered task system.
CRPD incorporated into the scheduling analysis

\[ R_i = C_i + \sum_{\forall j \in hp(i)} \left\lfloor \frac{R_i}{T_j} \right\rfloor \cdot \left( C_j + \gamma_{i,j} \right) \]

- \( hp(i) \): tasks of higher priority than task \( \tau_i \).
- \( \gamma_{i,j} \): preemption cost due to each job of a higher priority preempting task \( \tau_j \) executing within the worst-case response time of task \( \tau_i \).
Using well-known scheduling policies such as RM or EDF, schedulability improvement can be achieved by:

- limiting preemptions \((\text{Buttazzo et al. 2013})\),
- selecting the best possible preemption points in the program code, based on their overhead cost \((\text{Bertogna et al. 2011})\),
- ...

\(\Rightarrow\) reduce CRPD.

But, scheduling decisions are independent from any cache-related parameter.
All previous strategies → use of "classical" scheduling policies (RM, EDF...):

- CRPD added to achieve better predictability,
- **but** scheduling decisions are independant from any cache-related parameter.

Would it not be better to take scheduling decisions to reduce CRPD?

- Taking delays due to the use of caches into account in the definition of scheduling algorithms.

  - Task model modified → addition of cache-related parameters:
    1. representing the sequence of accessed block,
    2. representing preemption cost.
CRPD-aware scheduling

Scheduling decisions taken based on preemption costs → to minimize the general overhead.

Task defined by $\tau_i(C_i, D_i, T_i, \gamma)$

- $C_i$: WCET without preemption cost estimated when $\tau_i$ is executed fully non preemptively,
- $\gamma$: CRPD for one preemption → the same for all program points and all tasks.
\( \tau_1(1, 3), \tau_2(7, 12), \text{CRPD: } \gamma = 0.5. \)

- **Fixed-Job Priority Scheduling:**

  ![Fixed-Job Priority Scheduling Diagram]

  \( \Rightarrow \) Fixed-Task and Fixed-Job Priority schedulers are not optimal.

- **CRPD-aware scheduling:**

  ![CRPD-aware Scheduling Diagram]
Simplified scheduling with CRPD problem:

- **INSTANCE:**
  - a finite set of $n$ tasks $\tau_i(C_i, D_i, T_i)$,
  - a positive number $\gamma$ representing the Cache-Related Preemption Delay incurred by $\tau_i$, $1 \leq i \leq n$ at every resume point after a preemption.

- **QUESTION:**
  - Is there a uniprocessor preemptive schedule meeting the deadlines?

⇒ the scheduling problem with CRPD is **NP-hard**.
Cache-aware scheduling

Scheduling with information about cache state and block reuse by the different tasks.

- eg: tasks using the same data or a common external library.

Job defined by $J_i(C_i, D_i, S_i)$:

- $C_i$: WCET considering that all requested memory blocks are hits in the cache,
- $D_i$: relative deadline of the job,
- $S_i$: string denoting the sequence of memory blocks used during the job execution (no if-then-else structure).

Hypotheses:

- a single cache line,
- hit cost = 0, miss cost = $BRT$ (Block Reload Time),
- job preemption → only before requesting the next block,
- synchronous jobs.
Cache size = 1, exec.(hit) = 1, exec.(miss) = 1.5, $S_1 = cbabd$, $S_2 = ebaf$.

- **Fixed-Job Priority Scheduling** ($prio(J_1) > prio(J_2)$):

  \[
  \begin{array}{cccccccccccc}
  J_1 & c & b & a & b & d \\
  J_2 & e & b & a & f \\
  \text{Cache} & Miss & Miss & Miss & Miss & Miss & Miss & Miss \\
  0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13
  \end{array}
  \]

  $\Rightarrow$ **Fixed-Task and Fixed-Job Priority schedulers are not optimal.**

- **Cache-aware scheduling**:

  \[
  \begin{array}{cccccccccccc}
  J_1 & c & b & a & b & d \\
  J_2 & e & b & a & f \\
  \text{Cache} & Miss & Miss & Hit & Miss & Hit & Miss & Miss \\
  0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13
  \end{array}
  \]
Simplified scheduling with cache memory problem:

- **INSTANCE:**
  - a finite alphabet $\Sigma \rightarrow$ representing all accessed blocks,
  - a finite set of $n$ jobs $J_i(C_i, D, S_i)$ with a common deadline $D$,

- **QUESTION:**
  - Is there a uniprocessor preemptive schedule meeting the overall deadline $D$ for every job $J_i$?

$\Rightarrow$ the scheduling problem with cache memory is **NP-hard**.
Improving WCET ⇒ make scheduling more complex.
Many questions:
- which task(s) model(s)?
- which parameters for the timing analysis (WCET, CRPD...)?
- ...

Guillaume Phavorin and Pascal Richard
Interactions between WCET analysis and scheduling
Thank you for your attention!