WCET Analysis for Preemptive Scheduling

Sebastian Altmeyer\textsuperscript{1}  Gernot Gebhard\textsuperscript{2}

\textsuperscript{1}Saarland University
\textsuperscript{2}AbsInt GmbH

WCET Workshop, July 2008
Overview

1. Preemptive Scheduling
   Targeted System
   Preemptive vs. Non-preemptive Scheduling

2. Influence Preemption Costs
   Cache Set Classification
   Cost Function
   Optimization

3. WCET Analysis for Preemptive Scheduling
Targeted System and Notation

- set of $n$ tasks $\tau_1 \ldots \tau_n$
- scheduled preemptively
- combined data/instruction cache
- $k$-way LRU or direct-mapped caches (for the sake of simplicity)

- task-to-task relation: $\tau_i \vdash \tau_j \Leftrightarrow$ task $\tau_i$ can preempt task $\tau_j$ (for instance, given by priorities, data dependencies, etc.)
- set of data fragments $D_i = \{d_{i,1}, \ldots, d_{i,l}\}$ for each task (continuous data block such as arrays, instruction block, etc.)
Targeted System and Notation

- set of $n$ tasks $\tau_1 \ldots \tau_n$
- scheduled preemptively
- combined data/instruction cache
- $k$-way LRU or direct-mapped caches
  (for the sake of simplicity)

- task-to-task relation: $\tau_i \vdash \tau_j \iff$ task $\tau_i$ can preempt task $\tau_j$
  (for instance, given by priorities, data dependencies, etc.)
- set of data fragments $D_i = \{d_{i,1}, \ldots, d_{i,l}\}$ for each task
  (continuous data block such as arrays, instruction block, etc.)
Targeted System and Notation

• set of $n$ tasks $\tau_1 \ldots \tau_n$
• scheduled preemptively
• combined data/instruction cache
• $k$-way LRU or direct-mapped caches
  (for the sake of simplicity)

• task-to-task relation: $\tau_i \vdash \tau_j \Leftrightarrow$ task $\tau_i$ can preempt task $\tau_j$
  (for instance, given by priorities, data dependencies, etc.)
• set of data fragments $D_i = \{d_{i,1}, \ldots, d_{i,l}\}$ for each task
  (continuous data block such as arrays, instruction block, etc.)
Preemptive vs. Non-preemptive Scheduling

Non-preemptive scheduling

- tasks are running to completion
- (nearly) no inter-task cache-interference
- timing analysis feasible

Preemptive scheduling

- tasks may be preempted
- strong inter-task interference
- timing analysis much more complex (due to cache interference)

- some task-sets only schedulable using preemptive scheduling
Preemptive vs. Non-preemptive Scheduling - Example

Non-preemptive scheduling:
- unknown cache states only at the beginning
- tasks are running to completion

Preemptive scheduling:
- possible preemptions at unknown points
- unknown cache states at the beginning and after preemption
- preemptions task changes cache state of preempted task

additional costs to reload cache entries
Influence of the Memory Layout

Evicted cache-entries determined by the memory layout (i.e. the arrangement of code and data in the memory)

Example:

- direct mapped cache of size $n$
- 3 tasks ($\tau_1, \tau_2, \tau_3$) of size $n/2$
- $\tau_1$ can preempt the other two ($\tau_1 \triangleright \tau_2$ and $\tau_1 \triangleright \tau_3$)

![Diagram of memory layout](image)

- Layout 1: bad performance
- Layout 2: good performance

WCET Analysis for Preemptive Scheduling
Sebastian Altmeyer, Gernot Gebhard
WCET Workshop, July 2008
Cache-Set Classification

all data fragments $d_{i,j}$ on all cache sets $s$ are classified as follows

$$cl(d_{i,j}, s) =$$

**persistent:** $d_{i,j}$ does not occupy $s$ or at most $k$ data fragments of tasks that can preempt task $\tau_i$ occupy cache set $s$ $\Rightarrow$ even if task $\tau_i$ is preempted, $d_{i,j}$ on cache set $s$ still cached

**endangered:** $d_{i,j}$ occupies $s$ and at least $k + 1$ data fragments of tasks that can preempt task $\tau_i$ occupy cache set $s$ $\Rightarrow$ if task $\tau_i$ is preempted, $d_{i,j}$ on cache set $s$ could be evicted
Changing the Memory Layout

Different memory layouts lead to different preemption costs. We need

- metric to compare different memory layouts,
- optimization method.
Metric on Memory Layouts

costs of memory layout $C_L$ determined by all endangered data fragments over all cache sets

$$C_L = \sum_{d_{i,j}} \sum_{\text{cache set } s} W(d_{i,j}) \cdot \text{confl}(d_{i,j}, s)$$

with

$$\text{confl}(d_{i,j}, s) = \begin{cases} 
1 & \text{if } \text{cl}(d_{i,j}, s) = \text{endangered} \\
0 & \text{if } \text{cl}(d_{i,j}, s) = \text{persistent}
\end{cases}$$

• weight function $W$ used to increase precision
Metric on Memory Layouts (cont’d)

Data fragments do not contribute equally to the preemption costs (for instance, straight-line code vs. loops)

- weight function only approximates preemption costs
- weight data fragments according to their uses
- evaluation and testing of different weight function still future work
Optimization

- restriction to hole-free layouts
  ⇒ layout represented as a permutation
- finding optimal layout (still) NP-complete
  ⇒ find local instead of global optimum

Hill-climbing:

1. start with random layout \( L \)
2. search for a better layout \( L' \) in the set of neighbors of \( L \)
3. if \( L' \) exists, goto 1 with \( L := L' \)
4. restart searching with next best layout at most \( P \) times

- step 4 is used to jump over local hills
- parameter \( P \) determines how often
WCET Analysis for Preemptive Scheduling

- cache-set classification is new input to the analysis
- between cache analysis and low-level analysis
In case a cache-entry is classified as:

- **persistent**: analysis behaves as usual (even in case of preemption, cache-entry still valid)
- **endangered**: depends on the cache-analysis:
  - **hit**: cache-hit or cache-miss
  - **miss**: surely a cache-miss
  - **unknown**: cache-hit or cache-miss
Structure of the Approach

- Tasks
- Schedule
- Structural Analysis
  - cost function
- Optimization
  - classification
- WCET Analysis
- Memory Layout
- WCET$_{preempt}$
Conclusions...

- optimization and analysis of the memory layout
- classification into endangered and persistent cache-entries
- straight-forward extension of the WCET analysis

... and Future Work

- implement and evaluate the approach
- evaluate (and improve) metric on the memory layouts
- extend by information about preemption points
Conclusions...

- optimization and analysis of the memory layout
- classification into endangered and persistent cache-entries
- straight-forward extension of the WCET analysis

... and Future Work

- implement and evaluate the approach
- evaluate (and improve) metric on the memory layouts
- extend by information about preemption points
Thanks for your attention!