Investigation and Improvement on the Impact of TLB misses in Real-Time Systems

Takuya Ishikawa, Toshikazu Kato, Shinya Honda and Hiroaki Takada
Nagoya University, Japan
Memory protection for real-time systems is needed to ensure safety
- detects illegal memory access
- requires hardware support

Memory Management Unit (MMU)
- performs address translation with a page table
  - mapping a virtual address to a physical address
  - contains a field for access permission

Translation Lookaside Buffer (TLB)
- a page table entry is cached
  - to improve address translation speed
  - a page table is in main memory
- TLB miss: TLB does not contain a required entry
  - a required entry is obtained from a page table (page table walk) and loaded it to a TLB entry
Introduction (2/2)

- In real-time systems
  - worst case execution time (WCET) is important for schedulability analysis
  - WCET is pessimistically estimated
    - it is difficult to estimate the exact WCET
- TLB have the potential to make WCET more pessimistic
  - it is difficult to predict the occurrence of TLB misses

- Impact of TLB misses on WCETs should be studied
- predictability of WCETs with TLB should be improved
Overview of the Research

1. Impact of TLB misses on WCETs is evaluated
   - execution time distribution of a real-time task in a micro-benchmark with TLB is measured

2. Methods to improve WCETs with TLB are proposed
   - take advantage of TLB locking
     ✓ inhibit the occurrence of TLB misses related to a specified TLB entry
   - 2 types of methods: static one and dynamic one
     ✓ enable a WCET of a real-time task to be estimated
Evaluation Environment

- target processor: SH7750R (235MHz)
  - MMU with TLB which has 64 entries for instruction and data
    - page size is 4096 bytes
    - each entry has ASID (address space identifier)
      - TLB needs not to be flushed with context switch
  - TLB miss exception is handled by software (RTOS)
    - TOPPERS/HRP2 kernel is used as RTOS
      - ASID identifies a currently running application
      - each application has its own address space
      - HRP2 changes ASID with context switch
- to make it easier to evaluate an impact of TLB misses
  - cache is disabled (turned off)
  - interrupts are occurred by only a timer, and the result of evaluation does not include this interrupt execution time
Micro-Benchmark Software (1/2)

- includes automotive application software and multimedia application software
  - consideration of automotive software integration
    - ex.) both automotive control systems and automotive navigation systems are in the same electronic control unit
  - automotive software is real-time application
  - multimedia software is non-real-time application
- benchmark software is included in EEMBC benchmark*
  - automotive software is AutoBench
  - multimedia software is DENBench

Micro-Benchmark Software (2/2)

- There are 5 real-time tasks (RT_TASK1,2,...5) and 1 non-real-time task (NR_TASK)
  - each task has different ASIDs (is allocated different application)

- Execution flow
  - NR_TASK is activated firstly
  - RT_TASK1 is activated periodically
  - RT_TASK_n executes the own benchmark, and activates next RT_TASK_{n+1}

- RT_TASKs are activated 10,000 times

<table>
<thead>
<tr>
<th>task ID</th>
<th>benchmark name</th>
<th>memory size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT_TASK1</td>
<td>ttsprk01</td>
<td>58,164</td>
</tr>
<tr>
<td>RT_TASK2</td>
<td>a2time01</td>
<td>6,564</td>
</tr>
<tr>
<td>RT_TASK3</td>
<td>rspeed01</td>
<td>4,088</td>
</tr>
<tr>
<td>RT_TASK4</td>
<td>puwmod01</td>
<td>13,392</td>
</tr>
<tr>
<td>RT_TASK5</td>
<td>canrdr01</td>
<td>9,360</td>
</tr>
<tr>
<td>NR_TASK</td>
<td>djpegv2data4</td>
<td>535,332</td>
</tr>
</tbody>
</table>
Impact of TLB misses on WCETs

- evaluation metrics: \( \frac{E_H}{E_T - E_H} \) [\%]
  - \( E_H \): execution time of TLB miss exception handlers
  - \( E_T \): execution time of an evaluated task
  - a ratio of the execution time of TLB miss exception
    handlers to that of an evaluated task with no occurrence
    of TLB misses
  - evaluate the average and maximum value

- evaluated task
  - RT_TASK1 (ttsprk01)
    - constant input value, which maximizes the execution time
  - RT_TASK1 with random input value and RT_TASK1 to RT_TASK5 with
    random input value (omitted in this paper...)

- average execution time of each TLB miss
  exception handler
  - 16.9 microseconds
Result of Evaluation

- execution time distribution and metrics value

![Graph showing execution time distribution and metrics value]

- the impact of TLB misses on the WCET can be considerable
Improvement on WCET with TLB Locking

- **TLB locking**
  - makes a specified TLB entry not to be replaced
    - inhibits the occurrence of TLB misses
  - assigns a normal entry of TLB as a locked entry
    - after locked entry is released, that can be used as normal one
  - implemented in an RTOS (on SH7750R processors)
    - some of ARM processors implement in hardware

- **Proposed TLB locking methods**
  - reduce TLB misses while a real-time task is running
  - improve the WCET of a real-time task

<table>
<thead>
<tr>
<th>ASID</th>
<th>virtual page</th>
<th>physical page</th>
<th>attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x80100</td>
<td>0x80100</td>
<td>RX, Non-Shared</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0xc0200</td>
<td>0xc0200</td>
<td>RW, Non-Shared</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0xc0f00</td>
<td>0xc0f00</td>
<td>RW, Shared</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0x80108</td>
<td>0x80108</td>
<td>RX, Non-Shared</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Static Locking Method

- locks the target TLB entries during system initialization
  - these entries remain locked while system is running
- TLB locking increase only the execution time of system initialization
- response time of an entire system can be increased
  - TLB entries for NR_TASK are reduced, and TLB misses can be increased while NR_TASK is running

![Diagram showing RT_TASK and NR_TASK with TLB entries]

- Normal Entry
- Locked Entry for RT_TASK

used only RT_TASK
Dynamic Locking Method

- locks the target TLB entries when a job of a RT_TASK is activated, and releases these entries when a job ends
- released entries can be used as normal entries
- TLB misses while NR_TASK is running can be less than static locking method
- execution time of RT_TASK can be increased, because of TLB locking and releasing
  - prepare a table for locked TLB entries instead of a page table walk

<table>
<thead>
<tr>
<th>Time</th>
<th>RT_TASK</th>
<th>NR_TASK</th>
<th>TLB</th>
</tr>
</thead>
<tbody>
<tr>
<td>lock TLB entries</td>
<td>release locked entries</td>
<td>NR_TASK can use all TLB entries</td>
<td></td>
</tr>
</tbody>
</table>
TLB entry invalidation

- An entry which indicates the same virtual page as a locked entry shall be invalidated before locking.
- Different TLB entries indicate the same virtual page: exception.

- Static locking method: invalidate all TLB entries.
  - Invalidation has little impact on subsequent TLB misses.
  - Invalidate all entries by one memory access (to memory mapped MMU register).

- Dynamic locking method: invalidate not all TLB entries simply.
  - Invalidation has much impact on entire TLB misses.

![Diagram showing TLB entries and locking methods](image)
Dynamic Invalidation Method I

- invalidates only TLB entries related to locked entries
- does not invalidate unrelated entries
- invalidation processes are increased by the number of locked entries

Iterate this process for each locked entry.
Dynamic Invalidation Method II

- invalidate all TLB entries
  - saves all TLB entries to memory before invalidation
  - restores all TLB entries from memory when locked entries are released
  - TLB entries for NR_TASK are not expired
  - saving and restoring all entries can be increase the response time

![Diagram of TLB process]
Evaluation for TLB Locking

- apply proposed methods to the micro-benchmark
- evaluation items
  1. execution time distribution of RT_TASK and metrics value
  2. execution time of each dynamic locking method
  3. execution efficiency of the entire micro-benchmark
     ✓ measure response time, execution time and the number of TLB misses of NR_TASK

- apply TLB locking to
  ✓ RT_TASK1 with the constant input
     → 16 pages are locked
  ✓ RT_TASK1 to RT_TASK5 (all RT_TASKs) with the random inputs
     → 49 pages are locked
Result of Execution Time of RT_TASK1

- execution time distribution and metrics value

- the metrics value are 0 % in all cases
  all TLB locking methods can get rid of TLB misses for RT_TASK
Result of Execution Time of Locking

- Dynamic method I is effective if locked pages are less
- Dynamic method II is effective if locked pages are more
  - Threshold is 52

### Graph Details
- **Execution time** in microseconds
- **Number of pages**
- **TLB miss exception**
- **Dynamic method I**
- **Dynamic method II**

**Key Points**
- Save and restore all entries: 268 microseconds
- Gradients: increase 9.6 microseconds/page
- Gradients: increase 4.2 microseconds/page
- Gradients: increase 17 microseconds/page
Result for NR_TASK

- **static locking method**
  - difference is about 0.1 % if the number of locked page is less
    - ✓ decrease response time because decrease execution time of RT_TASK1
  - increase response time, execution time and TLB misses if the number of locked page is more

- **dynamic locking method**
  - method II has less impact on NR_TASK than method I
    - saving and restoring TLB entries are effective for execution of NR_TASK
Conclusion

● evaluate the impact of TLB misses on WCETs
  ● TLB misses can have a considerable impact on WCETs of real-time tasks

● propose the TLB locking methods
  ● improve WCETs of real-time tasks
    ✓ lock entries related to real-time tasks
  ● static method and 2 dynamic methods are proposed
    ✓ evaluation shows features of methods

● Future Works
  ● the impact of TLB misses and proposed methods should be evaluated with other environments
    ✓ different task sets, hardware TLB management, enabled cache, ...
  ● methods to select locked entries should be considered
    ✓ for lack of lockable TLB entries
Page Table Walk

• implementation in hardware
  • a TLB miss is handled by hardware
    ✓ overhead is less
  • format of page table is defined by hardware
    ✓ hierarchical page table
  • TLB controls by software are limited
    ✓ ex) TLB flush is enabled
• adopted by ARM, SPARC V8 architecture

• implementation in software
  • a TLB miss is handled by software
    ✓ overhead is more
  • format of page table is freedom, and TLB can be controlled by software
    ✓ flexible
• adopted by SH, MIPS architecture
Page Table in HRP2 on SH7750R

<table>
<thead>
<tr>
<th>address space</th>
<th>page table</th>
</tr>
</thead>
<tbody>
<tr>
<td>application 1</td>
<td>application 2</td>
</tr>
<tr>
<td>.data_app1</td>
<td>RW, Non-Shared</td>
</tr>
<tr>
<td>.bss_app1</td>
<td></td>
</tr>
<tr>
<td>.data_app2</td>
<td>RX, Non-Shared</td>
</tr>
<tr>
<td>.bss_app2</td>
<td></td>
</tr>
<tr>
<td>.text_app1</td>
<td>RX, Shared</td>
</tr>
<tr>
<td>.rodata_app1</td>
<td></td>
</tr>
<tr>
<td>.text_shared</td>
<td>RX, Shared</td>
</tr>
<tr>
<td>.rodata_shared</td>
<td></td>
</tr>
</tbody>
</table>
MMU in SH7750R

- paging virtual address space (32 bit)
  - page size is 1KB, 4KB, 64KB or 1MB

- TLB
  - instruction TLB (ITLB) : 4 entries, fully associative
    - upper level of UTLB, controlled by hardware
  - shared TLB (UTLB): 64 entries, fully associative
    - software page table walk

- UTLB controlled by
  - LDTLB instruction
    - replaced entry is specified by MMUCR.URC
      - each memory access increments
  - access to memory mapped TLB
    - UTLB is mapped to memory
TLB locking in SH7750R

- TLB locking process
  - invalidate the same virtual page entry in TLB
    - stop CPU exception
  - modify MMUCR
    - MMUCR.URB configures area where URC is ranged
  - load page table entry to UTLB
    - set PTEH, PTEL and MMUCR.URC and issue LDTLB instruction

- data and code for TLB locking are in TLB-invalidated area
  - special virtual address space

- interrupt and exception is disabled while TLB locking

![Diagram of TLB locking](image)
Address Space of SH7750R
Static Locking Process

critical section

entry

- disable int
- TLB flush
- save ASID
- complete TLB locking?
  - NO
    - invalidate all TLB entries
  - YES
    - set a page table entry to PTEH and PTEL
    - set MMUCR.URB and MMUCR.URC
    - LDTLB instruction
- restore ASID and set MMUCR.URC to 0
- enable int
- exit
Dynamic Locking Process I

**lock**

- **entry**
  - disable int
  - save ASID
  - complete TLB lock ?
    - YES: invalidate TLB entry, set a page table entry to PTEH and PTEL, restore ASID and set MMUCR.URC to 0, enable int, exit
    - NO: invalidate TLB entry, set MMUCR.URB and MMUCR.URC, LDTLB instruction
  - exit

**release**

- **entry**
  - disable int
  - set MMUCR.URB to 0
  - enable int
  - exit

Each entry for locked page is invalidated.
Release all locked entries.
Dynamic Locking Process II

- lock

  entry
  → disable int
  → save all TLB entries
  → TLB flush
  → save ASID
  → complete TLB lock ?
    → NO
      → set a page table entry to PTEH and PTEL
    → YES
      → restore ASID and set MMUCR.URC to 0
      → enable int
      → exit

- release

  entry
  → disable int
  → set MMUCR.URB to 0
  → restore all TLB entries
  → enable int
  → exit

Invalidate all TLB entries

Saving TLB

Restoring TLB

set MMUCR.URB and MMUCR.URC

LDTLB instruction

save all TLB entries

TLB flush

entry

YES

NO

save ASID

restore ASID and set MMUCR.URC to 0

enable int

exit
TLB miss overhead

- measure execution time from entry of TLB miss exception handler to end

![Bar Chart](image)
Evaluation for Impact of TLB miss

- **RT_TASK1 with random inputs**
  
  - Execution time is several hundreds microseconds
  
  - Average: 11.01%, Maximum: 76.28%

- **RT_TASK1 to RT_TASK5 (entire RT_TASKs)**
  
  - Average: 10.21%, Maximum: 36.91%
Result of Evaluation for TLB Locking

<table>
<thead>
<tr>
<th>method</th>
<th>WCET</th>
<th>overhead</th>
<th>execution efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>less pages</td>
<td>more pages</td>
</tr>
<tr>
<td>static locking</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>dynamic locking I</td>
<td>○</td>
<td>○</td>
<td>△</td>
</tr>
<tr>
<td>dynamic locking II</td>
<td>○</td>
<td>×</td>
<td>△</td>
</tr>
</tbody>
</table>

• Conclusion in this evaluation
  • TLB locking is efficient for improvement on WCETs
  • static locking method is effective if the number of locked entries is less
  • dynamic locking method II is effective if the number of locked entries is more
Related Works

- **Cache miss and cache lock** [1][2][3][4][5]
- **Page fault** [6]
- **TLB miss** [7]