Real-Time Executions of Program Codes in NAND Flash Memory

2009. 10. 20

Seoul National University
Chang-Gun Lee
Increasing Market of Flash Memory

Mobile embedded devices → shock resistance
Data and Codes are dramatically increasing → high volume
Flash Memory: Non-volatile, shock resistance, high volume

Traditional Embedded Systems

Soft Real-time Embedded Systems

Hard Real-time Embedded Systems

NAND: Data Storage
NOR: Code Storage
Why NAND is attractive for codes?

Increasing SW Complexity → Huge program codes

Soft Real-Time Embedded Systems (e.g., Multimedia Smart Phones)
- more than 5M source code lines in a smart phone
- cf. 5M source code lines in banking system

Hard Real-Time Embedded Systems (e.g., Automotive)

Memory-hierarchy for embedded systems

NAND-based low-cost, large-size Code execution technology
What is a big challenge of NAND for codes?

NAND Physical Characteristics

- Page based sequential Read (No Random Access)
  - Read: 130us/page
- Page based write
  - 300us/page
- Block based Erase
  - 2 ms/block
- No overwrite before erase

How to guarantee Program’s real-time execution with smallest RAM?
• RT-PLRU
  – Soft real-time
  – Single task
• mRT-PLRU
  – Extension to multiple tasks
• HRT-PLRU
  – Extension to hard real-time
RT-PLRU: Soft real-time single task

• Two Important Goals
  – Developer-transparency
  – Probabilistic guarantee of real-time with minimum DRAM

• Solution approach
  – Kernel-level auto-discovery of apps. temporal intension
  – Kernel-level auto-tracing of page reference sequences
  – Kernel-level auto-configuration (optimal) of pinning and LRU (RT-PLRU)

(a) LRU only

(b) Pinning only

(c) Pinning + LRU
Overall Design Flow (RT-PLRU)

kernel-level auto-tracing

(2,3,1,2)

Video Decoding  Sleeping  Video Decoding  Sleeping  Video Decoding  Sleeping

Time

production with RT-PLRU

(2,3,1,2)

kernel-level auto configuration

single instance

probabilistic extension for multiple instances

Prototype with sample movie
Comparison of required DRAM sizes
Implementations

(a) shadowing (The Lord Of The Rings 1)

(b) RT-PLRU (The Lord Of The Rings 1)

(c) shadowing (Starwars Ep2)

(d) RT-PLRU (Starwars Ep2)
• RT-PLRU
  – Soft real-time
  – Single task
• mRT-PLRU
  – Extension to multiple tasks
• HRT-PLRU
  – Extension to hard real-time
mRT-PLRU: Soft real-time Multiple tasks

• Problems to answer

Minimize this size
Such that
All three tasks “probabilistically” guarantee their deadlines
Step 1: Per-task analysis

- kernel-level auto-tracing

Prototype with sample content

<table>
<thead>
<tr>
<th>Video Decoding</th>
<th>Sleeping</th>
<th>Video Decoding</th>
<th>Sleeping</th>
<th>Video Decoding</th>
</tr>
</thead>
</table>

Optimal PLRU for a given size RAM

RAM size vs. opt U

Collection of optimal (pinning, LRU) combinations as a function of $S_k$
Step 2: Convex optimization

Convex approximation

Probabilistically meet deadlines for both tasks
How much RAM saved?

![Bar graph showing the total number of required SRAM pages for different required deadline meet probabilities and cache replacement policies.](chart)

- shadowing
- shared LRU
- even-partitioned LRU
- even-partitioned PLRU
- mHT-PLRU

<table>
<thead>
<tr>
<th>Required Deadline Meet Probability (%)</th>
<th>Shadowing</th>
<th>Shared LRU</th>
<th>Even-Partitioned LRU</th>
<th>Even-Partitioned PLRU</th>
<th>mHT-PLRU</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>203</td>
<td>196</td>
<td>96</td>
<td>124</td>
<td>105</td>
</tr>
<tr>
<td>95</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>204</td>
<td>202</td>
<td>126</td>
<td>117</td>
<td></td>
</tr>
</tbody>
</table>
Really work?

(a) shadowing ("Content A")  
(b) mRT-PLRU ("Content A")  
(c) shadowing ("Content B")  
(d) mRT-PLRU ("Content B")
More than two tasks?
- RT-PLRU
  - Soft real-time
  - Single task
- mRT-PLRU
  - Extension to multiple tasks
- HRT-PLRU
  - Extension to hard real-time
HRT-PLRU: Hard real-time Multiple tasks

- Problems to answer

Minimize this size
Such that
All three tasks “deterministically” guarantee their deadlines
Per-task analysis and Convex optimization

Deterministically meet deadlines for both tasks

WCET_1(S_1)  

WCET_2(S_2)
Step 1: Per-task analysis

- **WCET for a PLRU combination**

\[
WCET = \text{max} \left( \left( \sum_{i=1}^{11} e_i x_i \right) + \left( \sum_{j \in \text{PageTransition}} d_j^{\text{miss}} \right) \cdot \text{PageFaultDelay} \right)
\]

ILP can solve this!
Step 1: Per-task analysis

- RAM size vs. opt PLRU in terms of WCET

(a) $WCET_i(S_{LRU}^{i}, S_{Pinning}^{i})$ table

(b) RAM size $S_i$ vs. WCET relation
Step 2: Convex optimization

Schedulable!

\[
\frac{\text{WCET}_1}{p_1} + \frac{\text{WCET}_1}{p_2} \leq n(2^n-1)
\]
How much RAM saved?
Conclusion

• RT-PLRU for
  – Soft real-time single task $\rightarrow$ RT-PLRU
  – Soft real-time multiple tasks $\rightarrow$ mRT-PLRU
  – Hard real-time multiple tasks $\rightarrow$ HRT-PLRU
• It provides a potential to use NAND for code executions of real-time applications
• More study needed for practical applications
  – Trade-off between RAM cost and energy consumption
  – System bus conflict problems
  - etc.