Demand Code Paging for NAND Flash in MMU-less Embedded Systems

Jose Baiocchi and Bruce Childers

University of Pittsburgh
Pittsburgh PA USA
childers@cs.pitt.edu

Memory Shadowing

- Range of embedded systems commonly have both main memory and storage

- Flash storage: stores program binary image
- Main memory: holds both code and data
Memory Shadowing

- Range of embedded systems commonly have both main memory and storage

  - Flash storage: stores program binary image
  - Main memory: holds both code and data

Execute from memory
  Binary image copied to memory for execution

Flash holds code and data pages

Bruce Childers
University of Pittsburgh
Memory Shadowing

- **Demand paging**: Bring pages in as needed with OS and hardware (MMU) support
- Low cost embedded devices
  - May lack support to detect missing code through paging mechanism
- **Full shadowing**: Copy entire binary image
  - Copy latency needs to be amortized
  - Boot-up delay to copy binary image

Reduce copy and boot-up latencies when full shadowing cannot be amortized

Our Approach

- Dynamic binary translation
  - Virtualization, security, resource management, among many other uses
- Translation steps
  1. Fetch instruction from memory
  2. Decode
  3. Possibly modify
  4. Save instruction in software-managed buffer
  5. Execute instructions from buffer

“Translation” may be from same ISA to same ISA
Dynamic Binary Translation

- Fragment Cache
  - program started, translator entered
  - Execution captured in fragment cache

- Binary Image
  - Flash (organized as pages)
  - Some Flash pages aren’t actually needed

Bruce Childers
University of Pittsburgh
Dynamic Binary Translation

Our approach: DBT changed to load pages when needed

Demand Page Loading

- Simple idea: Load Flash code pages on demand
  - DBT translates along execution path

- Change Fetch to perform Flash page load
  - Load page for requested instruction
  - Return requested instruction
  - Buffer loaded page to avoid loading again

- Challenge: How to manage buffers for translated code and loaded pages
  - Scattered page buffer
  - Unified code buffer
Scattered Page Buffer

- Two buffers: Fragment cache + Page buffer
  - Managed by DBT system as part of Fetch step
  - Buffer size set to number pages in binary image
  - Unique buffer page per Flash page

Fetch steps
1. Check whether page for requested instruction address is already loaded
2. Load missing page to pre-determined location
3. Fetch instruction from loaded page

Essentially, full shadowing with pages loaded on-demand

---

Scattered Page Buffer

- Fully shadowing without DBT
- On-demand paging with DBT using scattered page buffer
Scattered Page Buffer

1. Minimizes number Flash page reads
2. Spreads Flash reads out over time
3. Potentially sparsely populated page buffer

- Advantage: Simple one-to-one mapping
  - Flash page at fixed location – either there or not
  - Low overhead: Quick lookup and no additional data structures
  - If DBT is already used, little additional overhead

- Disadvantage: Increases memory overhead
  - Footprint: Size of SPB + FC + DBT data structures

Unified Code Buffer

- Combine fragment cache + page buffer
  - Managed as single buffer
  - Multiple caching units: fragments and pages
  - Fixed constraint on total size

- Management more complex
  - Single buffer with regions for fragments (translated code) and pages (untranslated code)
  - Overflow between regions
  - Which region gets priority
Unified Code Buffer

- Page locality, eviction policy (LRU/FIFO), UCB capacity determine how well scheme works

- Advantage: Constrain total DBT footprint
  - UCB + DBT structures ≤ Full shadow size
  - 75% of full shadow size works well
  - DBT data structures are 1 word per 3 instructions

- Disadvantage: Performance overhead may be worse
  - May need to reload previously seen pages
  - Manage data structures, e.g., LRU information
Experimental Methodology

- When is DBT-based demand code paging needed
- Does UCB perform as well as SPB while mitigating memory footprint of DBT system
- Strata DBT for SimpleScalar/PISA
- Simulated SoC with 400 MHz ARM-like processor
- NAND Flash card Kingston 1GB CF, 0.6MB/sec
- MiBench with large input data sets (show selected)

- FS (baseline): Fully shadowed binary
- SPB: Scattered page buffer
- UCB-75%: FIFO; size set to 75% of FS

<table>
<thead>
<tr>
<th>Program</th>
<th>FS</th>
<th>SPB</th>
<th>UCB-75-FIFO</th>
<th>UCB-75-LRU</th>
</tr>
</thead>
<tbody>
<tr>
<td>fft</td>
<td>92</td>
<td>80</td>
<td>124</td>
<td>120</td>
</tr>
<tr>
<td>ghostscript</td>
<td>2047</td>
<td>971</td>
<td>971</td>
<td>971</td>
</tr>
<tr>
<td>lame</td>
<td>470</td>
<td>391</td>
<td>534</td>
<td>529</td>
</tr>
<tr>
<td>jpeg.dec</td>
<td>277</td>
<td>168</td>
<td>187</td>
<td>183</td>
</tr>
<tr>
<td>pgp.enc</td>
<td>524</td>
<td>290</td>
<td>292</td>
<td>291</td>
</tr>
<tr>
<td>susan.cor</td>
<td>149</td>
<td>88</td>
<td>91</td>
<td>89</td>
</tr>
</tbody>
</table>

Absolute number of page reads with full shadowing (FS), scattered page buffer (SPB) and unified code buffer (UCB) with FIFO and LRU and sized to 75% of binary image.
### NAND Page Reads

<table>
<thead>
<tr>
<th>Program</th>
<th>FS</th>
<th>SPB</th>
<th>UCB-75-FIFO</th>
<th>UCB-75-LRU</th>
</tr>
</thead>
<tbody>
<tr>
<td>fft</td>
<td>92</td>
<td>80</td>
<td>124</td>
<td>120</td>
</tr>
<tr>
<td>ghostscript</td>
<td>2047</td>
<td>971</td>
<td>971</td>
<td>971</td>
</tr>
<tr>
<td>lame</td>
<td>470</td>
<td>391</td>
<td>534</td>
<td>529</td>
</tr>
<tr>
<td>jpeg.dec</td>
<td>277</td>
<td>168</td>
<td>187</td>
<td>183</td>
</tr>
<tr>
<td>pgp.enc</td>
<td>524</td>
<td>290</td>
<td>292</td>
<td>291</td>
</tr>
<tr>
<td>susan.cor</td>
<td>149</td>
<td>88</td>
<td>91</td>
<td>89</td>
</tr>
</tbody>
</table>

Use full shadowing: small reduction, or page reads are well amortized

Bruce Childers  
University of Pittsburgh

### NAND Page Reads

<table>
<thead>
<tr>
<th>Program</th>
<th>FS</th>
<th>SPB</th>
<th>UCB-75-FIFO</th>
<th>UCB-75-LRU</th>
</tr>
</thead>
<tbody>
<tr>
<td>fft</td>
<td>92</td>
<td>80</td>
<td>124</td>
<td>120</td>
</tr>
<tr>
<td>ghostscript</td>
<td>2047</td>
<td>971</td>
<td>971</td>
<td>971</td>
</tr>
<tr>
<td>lame</td>
<td>470</td>
<td>391</td>
<td>534</td>
<td>529</td>
</tr>
<tr>
<td>jpeg.dec</td>
<td>277</td>
<td>168</td>
<td>187</td>
<td>183</td>
</tr>
<tr>
<td>pgp.enc</td>
<td>524</td>
<td>290</td>
<td>292</td>
<td>291</td>
</tr>
<tr>
<td>susan.cor</td>
<td>149</td>
<td>88</td>
<td>91</td>
<td>89</td>
</tr>
</tbody>
</table>

Use demand paging: large reduction and/or page reads are not amortized

Bruce Childers  
University of Pittsburgh
NAND Page Reads

<table>
<thead>
<tr>
<th>Program</th>
<th>FS</th>
<th>SPB</th>
<th>UCB-75-FIFO</th>
<th>UCB-75-LRU</th>
</tr>
</thead>
<tbody>
<tr>
<td>fft</td>
<td>92</td>
<td>80</td>
<td>124</td>
<td>120</td>
</tr>
<tr>
<td>ghostscript</td>
<td>2047</td>
<td>971</td>
<td>971</td>
<td>971</td>
</tr>
<tr>
<td>lame</td>
<td>470</td>
<td>391</td>
<td>534</td>
<td>529</td>
</tr>
<tr>
<td>jpeg.dec</td>
<td>277</td>
<td>168</td>
<td>187</td>
<td>183</td>
</tr>
<tr>
<td>pgp.enc</td>
<td>524</td>
<td>290</td>
<td>292</td>
<td>291</td>
</tr>
<tr>
<td>susan.cor</td>
<td>149</td>
<td>88</td>
<td>91</td>
<td>89</td>
</tr>
</tbody>
</table>

FIFO is nearly as good, yet is much simpler with less management overhead cost. Remaining results use FIFO.

Improvement in Boot Time

<table>
<thead>
<tr>
<th>Program</th>
<th>Factor Improvement in Boot Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>susan.cor</td>
<td></td>
</tr>
<tr>
<td>pgp.enc</td>
<td></td>
</tr>
<tr>
<td>jpeg.dec</td>
<td></td>
</tr>
<tr>
<td>lame</td>
<td></td>
</tr>
<tr>
<td>ghostscript</td>
<td></td>
</tr>
<tr>
<td>fft</td>
<td></td>
</tr>
<tr>
<td>Avg-All</td>
<td></td>
</tr>
</tbody>
</table>

Measured as delay to executing first application instruction.
**Improvement in Boot Time**

- **susan.cor**
- **pgp.enc**
- **jpeg.dec**
- **lame**
- **ghostscript**
- **fft**
- **Avg-All**

**UCB-75%**

**SPB**

Use demand paging

Measured as delay to executing first application instruction

**Improvement in Boot Time**

- **susan.cor**
- **pgp.enc**
- **jpeg.dec**
- **lame**
- **ghostscript**
- **fft**
- **Avg-All**

**UCB-75%**

**SPB**

Use full shadowing

Measured as delay to executing first application instruction

**Bruce Childers**
University of Pittsburgh
Improvement in Boot Time

Measured as delay to executing first application instruction

Significantly improved boot time even when shadowing is preferred

Bruce Childers
University of Pittsburgh

Improvement in Performance

Bruce Childers
University of Pittsburgh
Improvement in Performance

Bruce Childers
University of Pittsburgh

Use demand paging

Performance Speedup

Improvement in Performance

Bruce Childers
University of Pittsburgh

Use full shadowing

Performance Speedup
Improvement in Performance

- Susan.cor
- Pgp.enc
- Jpeg.dec
- Lame
- Ghostscript
- Fft
- Avg-Shadow
- Avg-Demand
- Avg-All

~5% loss - use full shadowing
~11% (UCB)
~14% (SPB) gain
use DBT demand paging

Performance Speedup

UCB-75%
SPB

UCB-75% nearly as good yet memory size about same as full binary shadowing
Conclusion

• Dynamic binary translation can serve as basis for on-demand code paging

• Challenge is how to manage combine memory resources to effectively hold pages and translated instructions
  • UCB most effective: Constrains footprint, yet does well when full shadowing shouldn’t be used

• Boot time and performance (UCB-75%)
  • Boot time 4.75x (average) faster
  • Performance 1.11 (average) speedup