

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

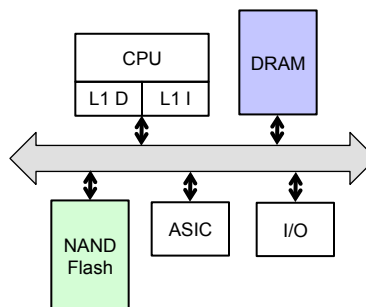
Jose Baiocchi and Bruce Childers

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



Memory Shadowing

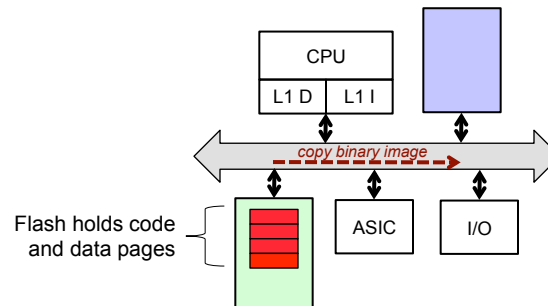
- Range of embedded systems commonly have both main memory and storage



- 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



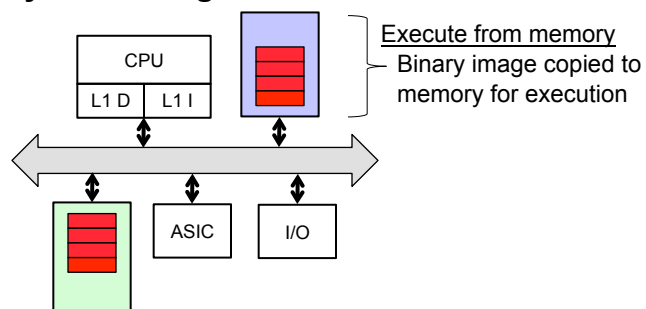
- **Flash storage:** stores program binary image
- **Main memory:** holds both code and data

Bruce Childers
University of Pittsburgh

3

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

Bruce Childers
University of Pittsburgh

4

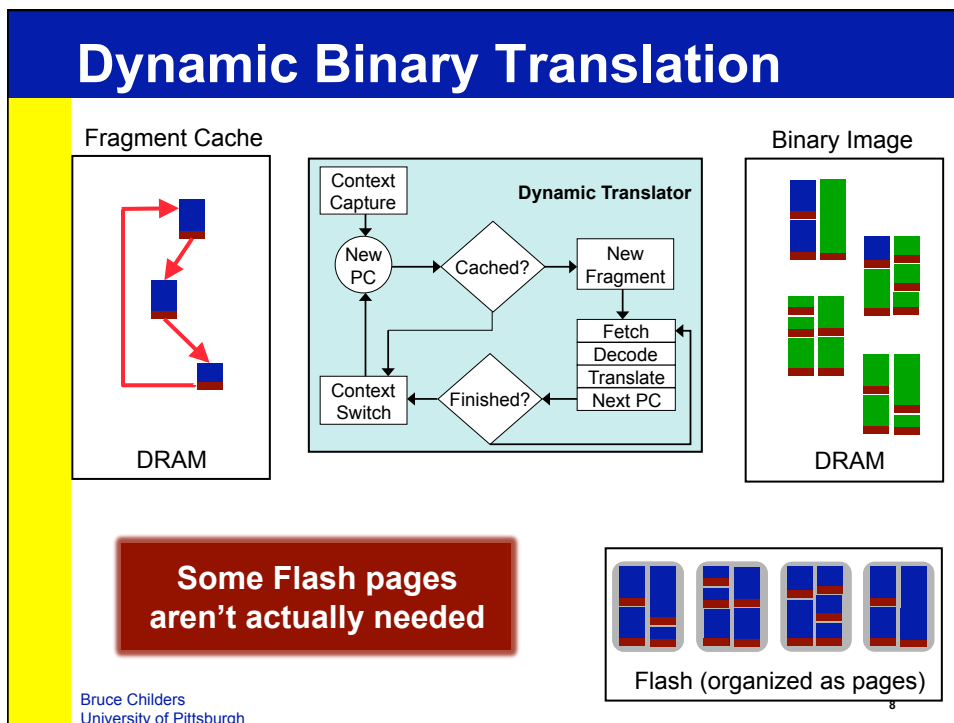
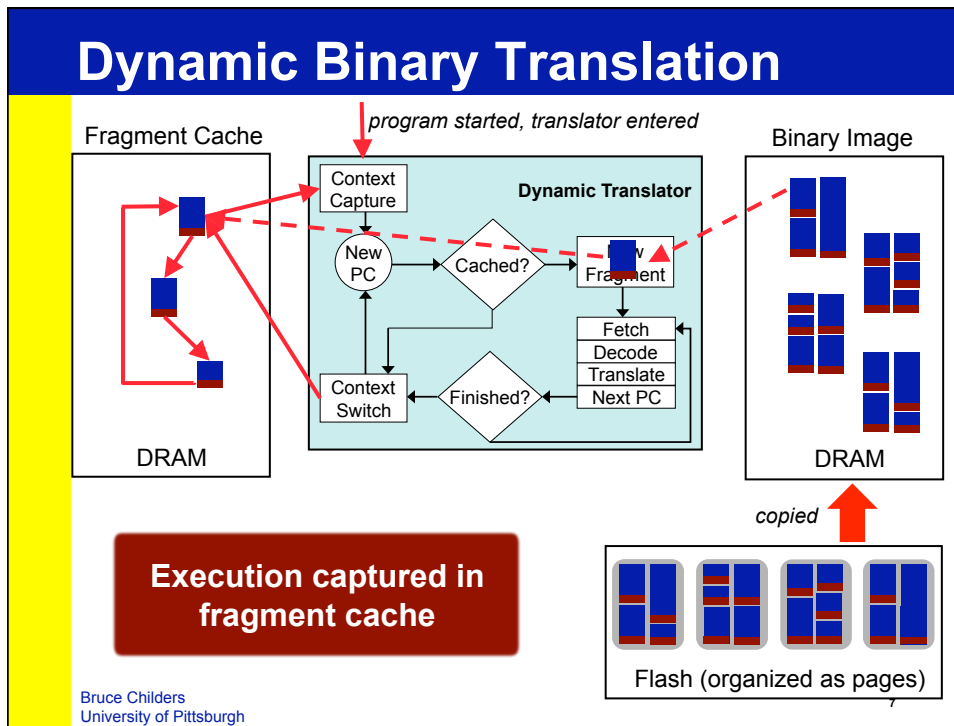
Memory Shadowing

- **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

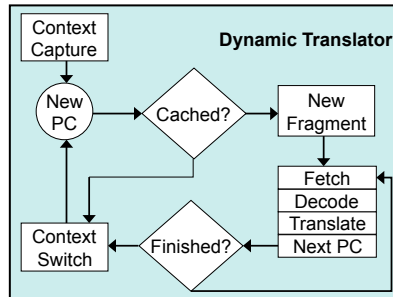
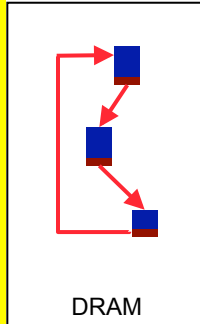
Our Approach

- **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**

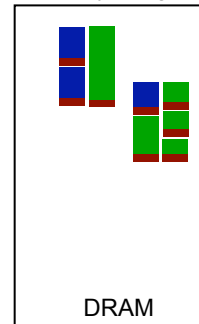


Dynamic Binary Translation

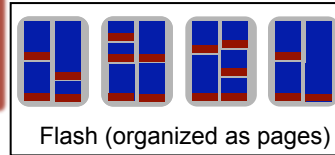
Fragment Cache



Binary Image



Our approach: DBT changed to load pages when needed



Bruce Childers
University of Pittsburgh

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

Bruce Childers
University of Pittsburgh

10

Scattered Page Buffer

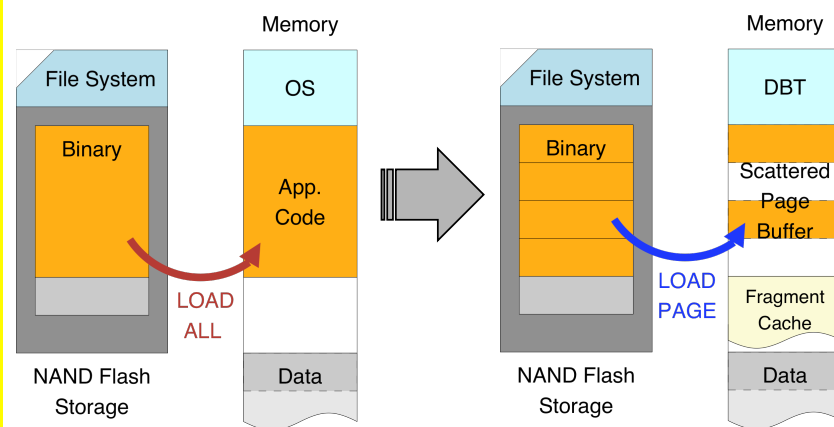
- Two buffers: Fragment cache + Page buffer
 - Managed by DBT system as part of Fetch step
 - Buffer size set to number pages in binary image
 - Unique buffer page per Flash page

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

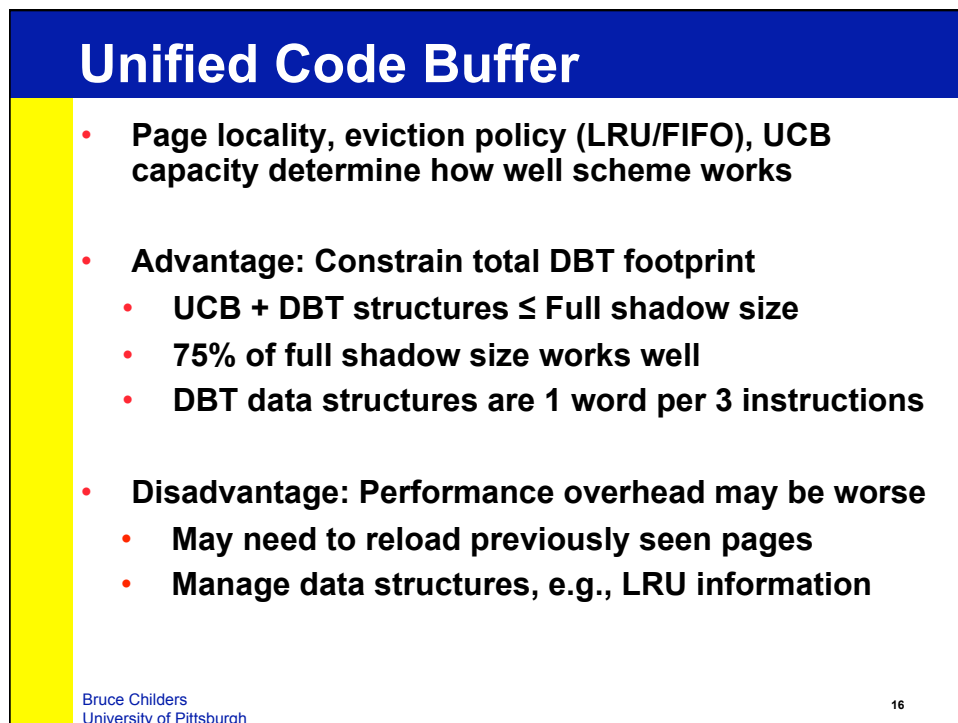
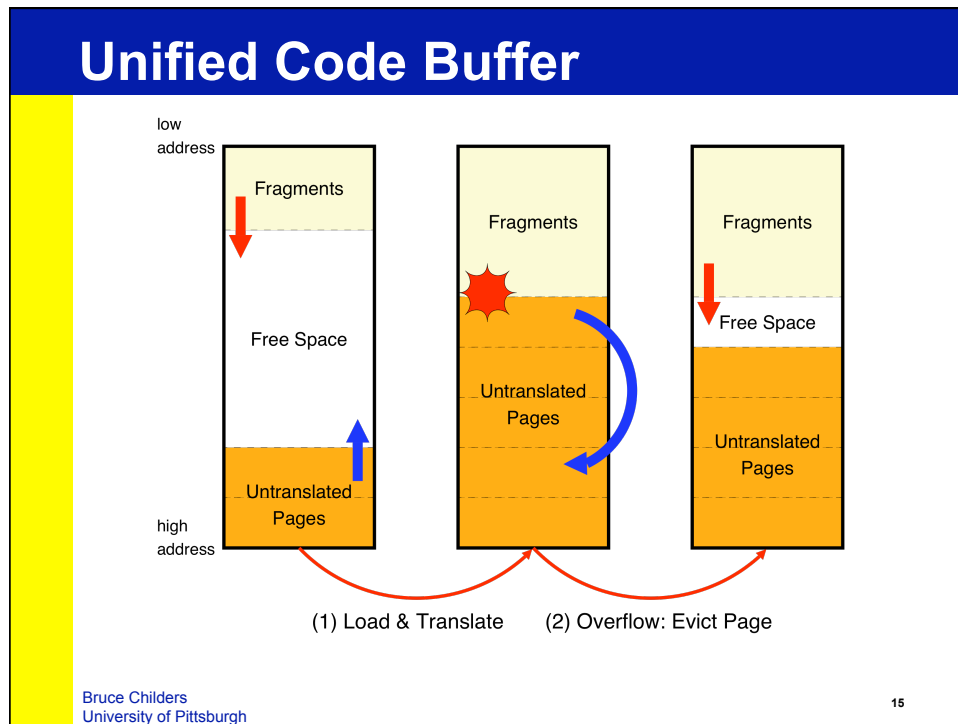
On-demand paging with DBT using scattered page buffer

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



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

Bruce Childers
University of Pittsburgh

17

NAND Page Reads

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

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.

Bruce Childers
University of Pittsburgh

18

NAND Page Reads

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

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

NAND Page Reads

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

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

NAND Page Reads

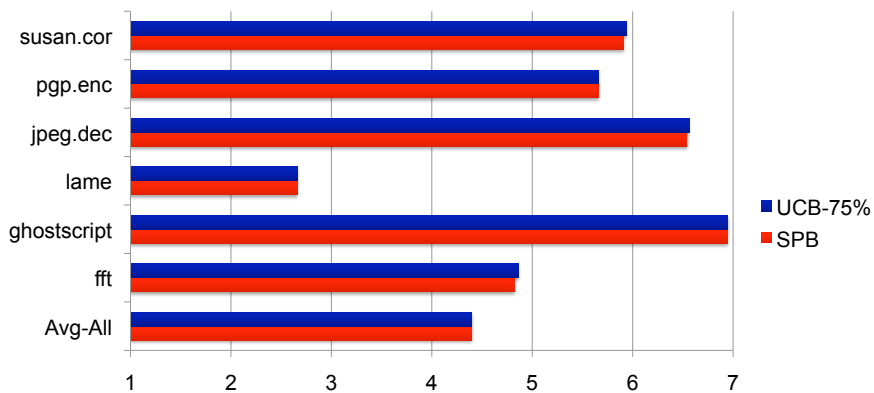
Program	FS	SPB	UCB-75-FIFO	UCB-75-LRU
fft	92	80	124	120
ghostscript	2047	971	971	971
lame	470	391	534	529
jpeg.dec	277	168	187	183
pgp.enc	524	290	292	291
susan.cor	149	88	91	89

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

Bruce Childers
 University of Pittsburgh

21

Improvement in Boot Time

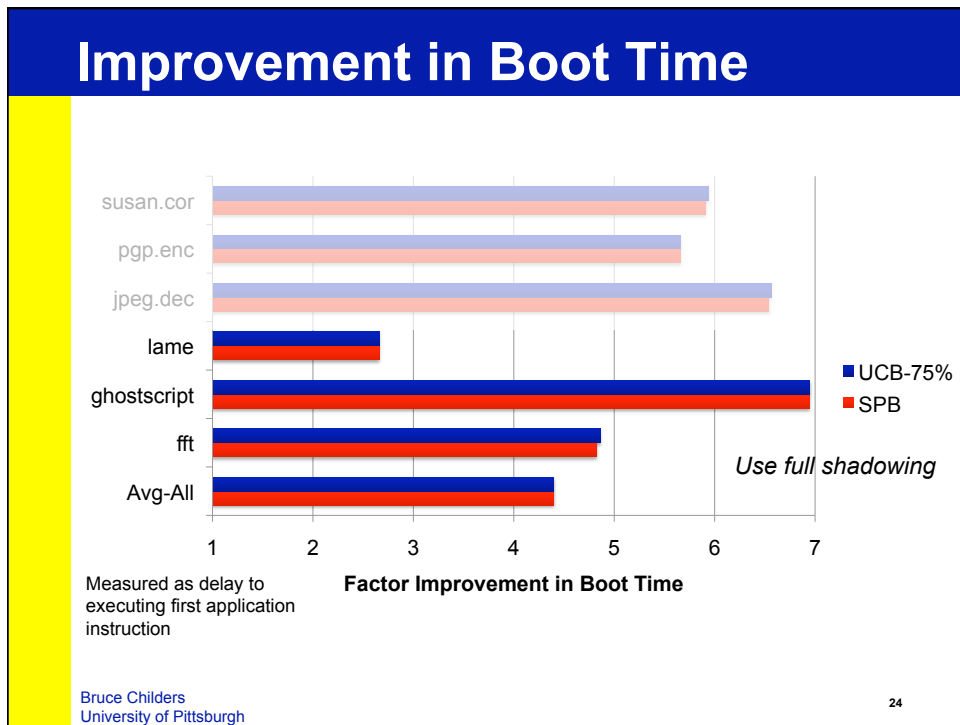
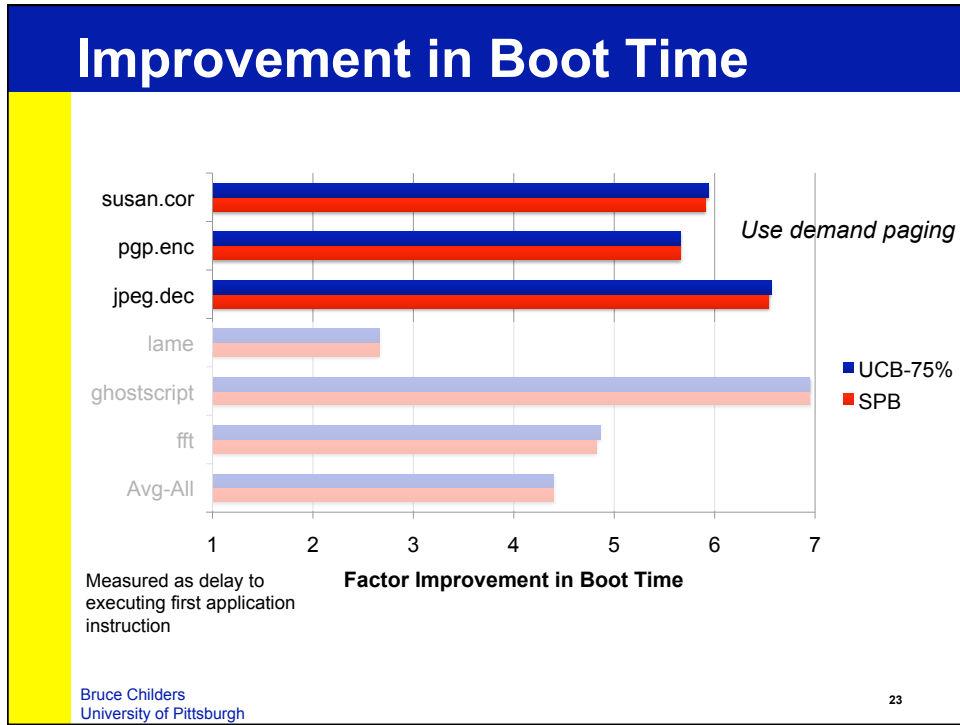


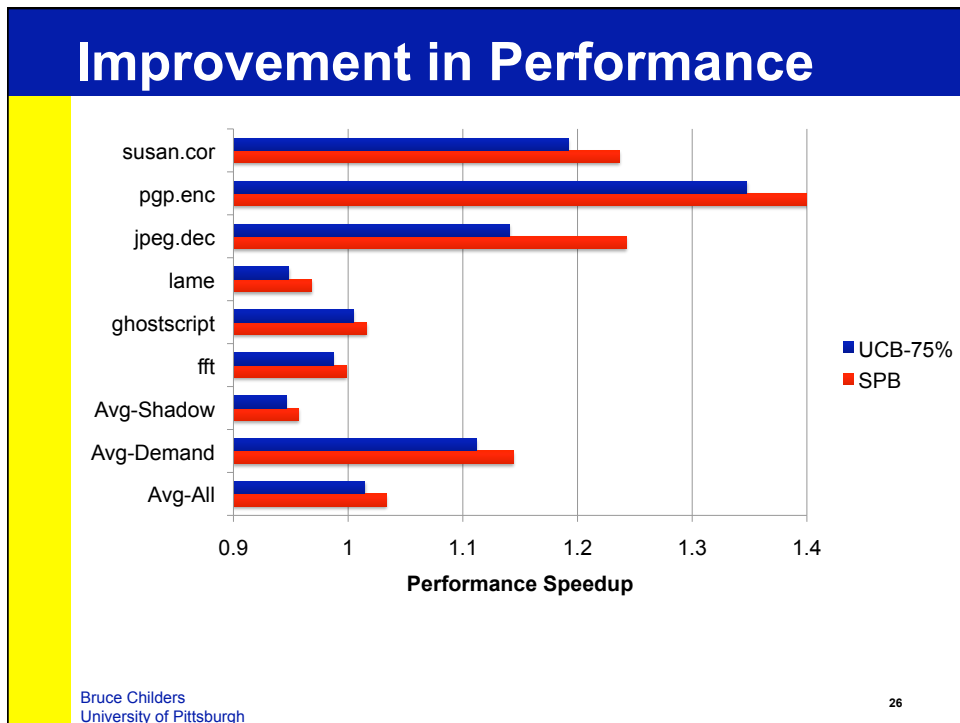
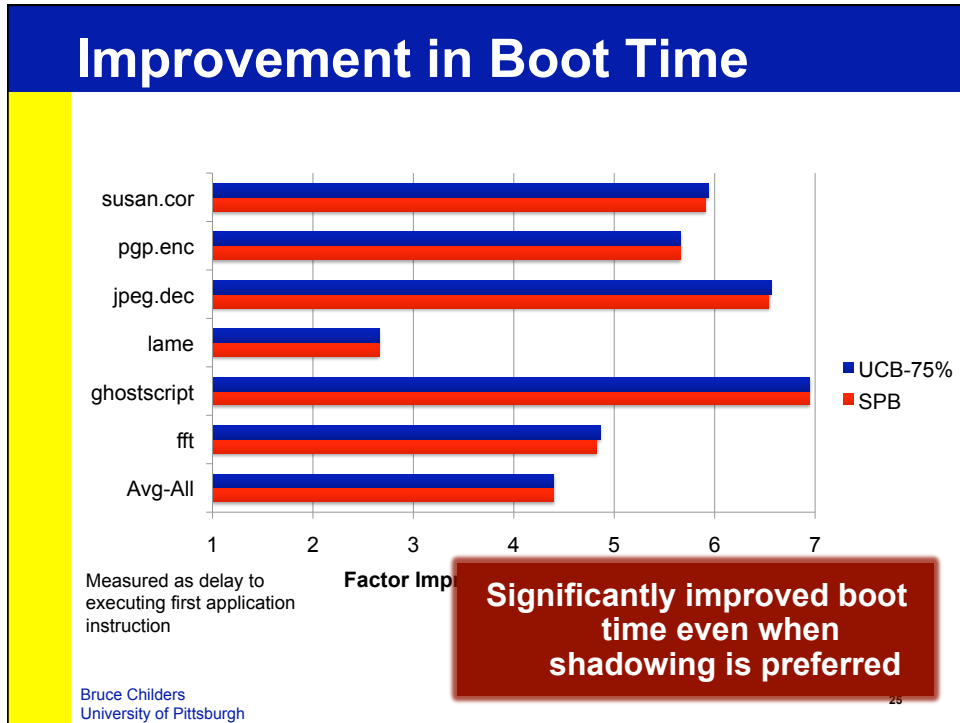
Measured as delay to executing first application instruction

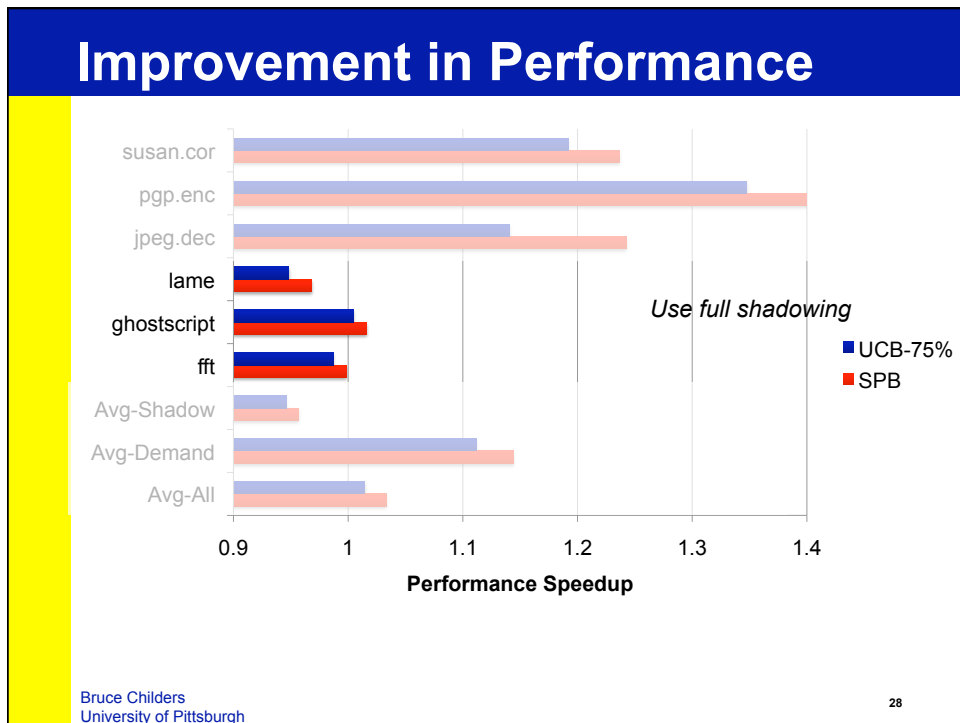
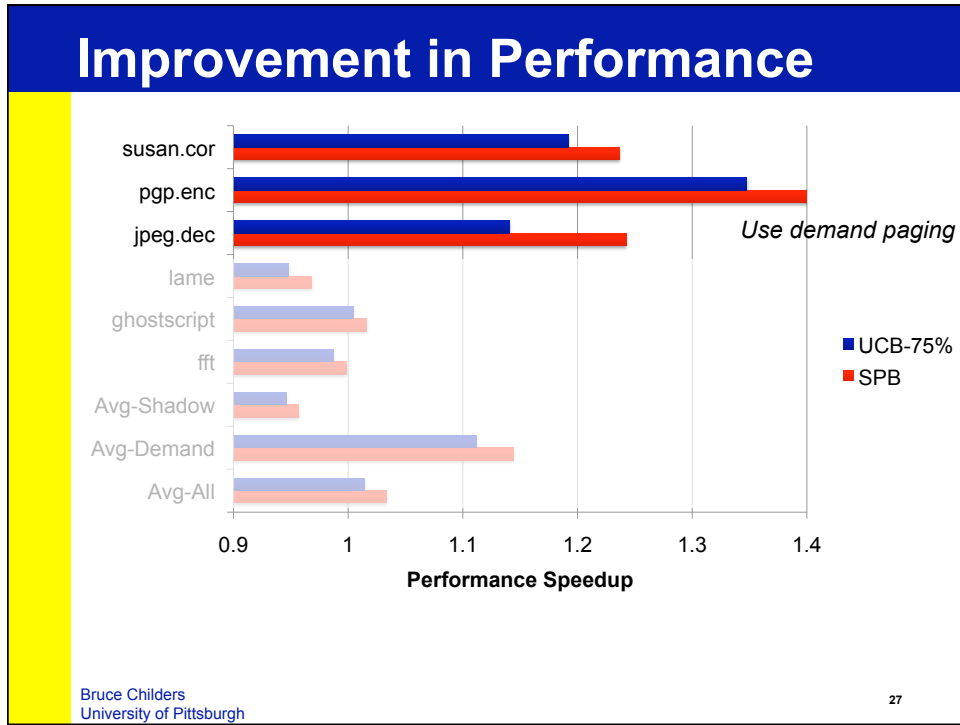
Factor Improvement in Boot Time

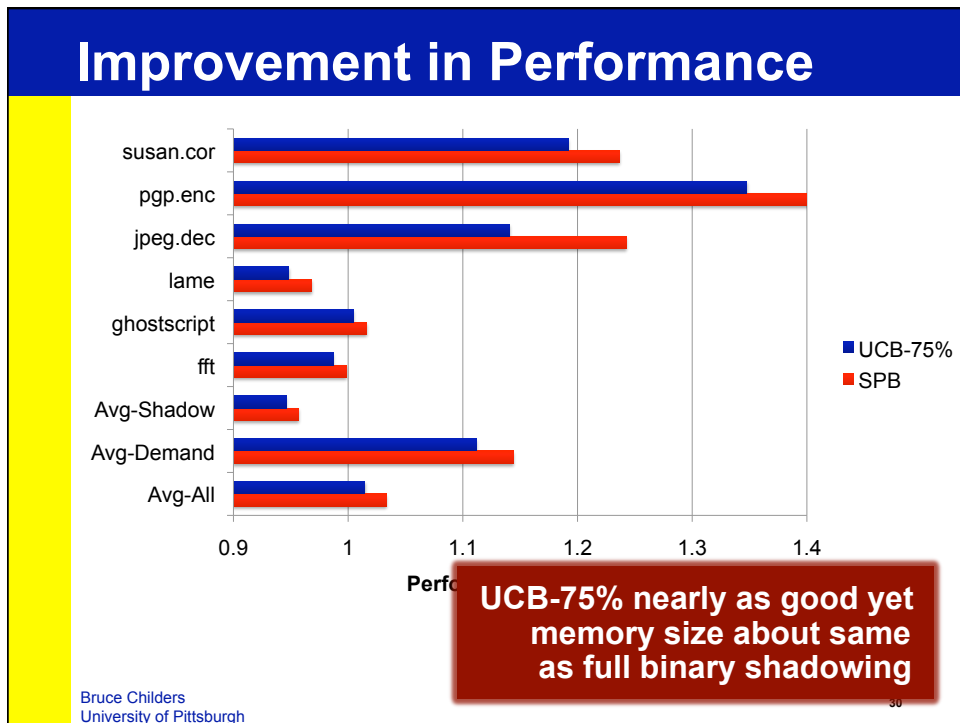
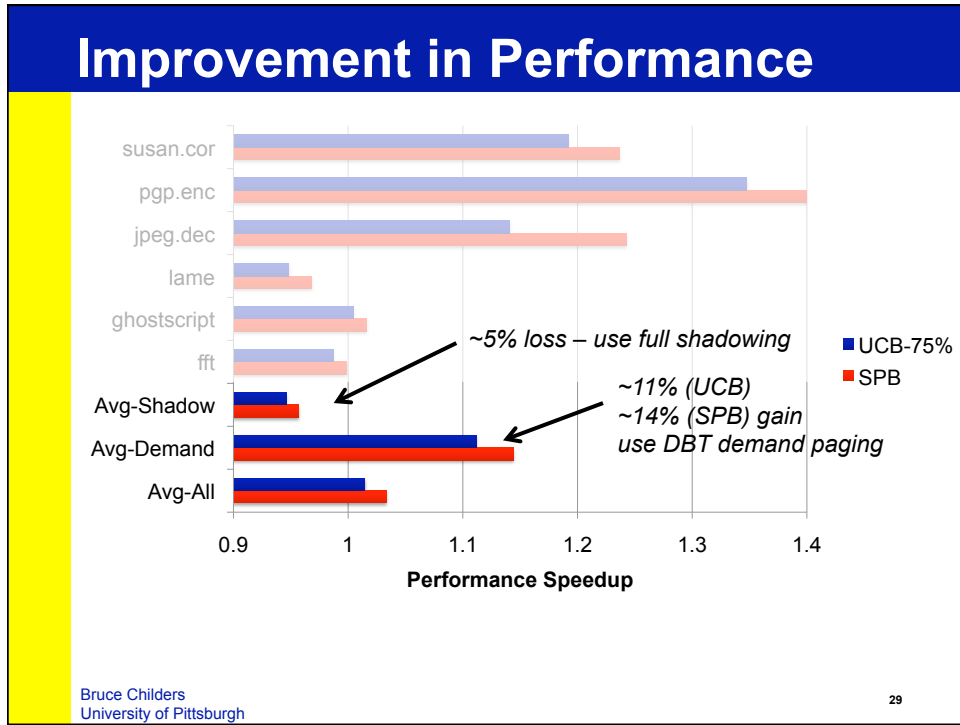
Bruce Childers
 University of Pittsburgh

22









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**