Review: Address Translation

- Map from virtual address to physical address
- Page Tables, PTE
  - va->pa, attributes
  - forward mapped, inverted, hashed, clustered
- Translation Lookaside Buffer
  - hardware cache of most recent va->pa translation
  - misses handled in hardware or software
- Implications of larger address space
  - page table size
  - possibly more TLB misses
- OS Structure
  - microkernels -> lots of IPC -> more TLB misses
Cache Memory 102

- Block 7 placed in 4 block cache:
  - Fully associative, direct mapped, 2-way set associative
  - S.A. Mapping = Block Number Modulo Number Sets
  - DM = 1-way Set Assoc
- Cache Frame
  - location in cache
- Bit-selection

Cache Indexing

- Tag on each block
  - No need to check index or block offset
- Increasing associativity shrinks index, expands tag

Fully Associative: No index
Direct-Mapped: Large index
Address Translation and Caches

- Where is the TLB wrt the cache?
- What are the consequences?

- Most of today’s systems have more than 1 cache
  - Digital 21164 has 3 levels
  - 2 levels on chip (8KB-data, 8KB-inst, 96KB-unified)
  - one level off chip (2-4MB)
- Does the OS need to worry about this?

Definition:
page coloring = careful selection of va->pa mapping

TLBs and Caches

Conventional Organization

Virtually Addressed Cache
Translate only on miss
Alias (Synonym) Problem

Overlap $ access
with VA translation:
requires $ index to remain invariant
across translation
Virtual Caches

- Send virtual address to cache. Called *Virtually Addressed Cache* or just *Virtual Cache* vs. *Physical Cache* or *Real Cache*
- Avoid address translation before accessing cache
  - faster hit time to cache
- Context Switches?
  - Just like the TLB (flush or pid)
  - Cost is time to flush + “compulsory” misses from empty cache
  - Add *process identifier tag* that identifies process as well as address within process: can’t get a hit if wrong process
- I/O must interact with cache

I/O and Virtual Caches

I/O is accomplished with physical addresses

DMA
- flush pages from cache
- need pa->va reverse translation
- coherent DMA
Aliases and Virtual Caches

- **aliases** (sometimes called **synonyms**); Two different virtual addresses map to same physical address
- But, but... the virtual address is used to index the cache
- Could have data in two different locations in the cache

Index with Physical Portion of Address

- If index is physical part of address, can start tag access in parallel with translation so that can compare to physical tag

<table>
<thead>
<tr>
<th>Page Address</th>
<th>Page Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Tag</td>
<td>Index</td>
</tr>
</tbody>
</table>

- Limits cache to page size: what if want bigger caches and use same trick?
  - Higher associativity
  - Page coloring

© Alvin R. Lebeck 1999
Page Coloring for Aliases

• HW that guarantees that every cache frame holds unique physical address
• OS guarantee: lower n bits of virtual & physical page numbers must have same value; if direct-mapped, then aliases map to same cache frame
  – one form of page coloring

Virtual Memory and Physically Indexed Caches

• Notion of bin
  – region of cache that may contain cache blocks from a page
• Random vs careful mapping
• Selection of physical page frame dictates cache index
• Overall goal is to minimize cache misses
Careful Page Mapping

[Kessler92, Bershad94]

- Select a page frame such that cache conflict misses are reduced
  - only choose from available pages (no replacement induced)
- static
  - “smart” selection of page frame at page fault time
- dynamic
  - move pages around

Page Coloring

- Make physical index match virtual index
- Behaves like virtual index cache
  - no conflicts for sequential pages
- Possibly many conflicts between processes
  - address spaces all have same structure (stack, code, heap)
  - modify to xor PID with address (MIPS used variant of this)
- Simple implementation
- Pick arbitrary page if necessary
**Bin Hopping**

- Allocate sequentially mapped pages (time) to sequential bins (space)
- Can exploit temporal locality
  - pages mapped close in time will be accessed close in time
- Search from last allocated bin until bin with available page frame
- Separate search list per process
- Simple implementation

**Best Bin**

- Keep track of two counters per bin
  - used: # of pages allocated to this bin for this address space
  - free: # of available pages in the system for this bin
- Bin selection is based on low values of used and high values of free
- Low used value
  - reduce conflicts within the address space
- High free value
  - reduce conflicts between address spaces
Hierarchical

- Best bin could be linear in # of bins
- Build a tree
  - internal nodes contain sum of child <used,free> values
- Independent of cache size
  - simply stop at a particular level in the tree

Benefit of Static Page Coloring

- Reduces cache misses by 10% to 20%
- Multiprogramming
  - want to distribute mapping to avoid inter-address space conflicts
Dynamic Page Coloring

- Cache Miss Lookaside (CML) buffer [Bershad94]
  - proposed hardware device
- Monitor # of misses per page
- If # of misses >> # of cache blocks in page
  - must be conflict misses
  - interrupt processor
  - move a page (recolor)
- Cost of moving page << benefit

Outline

- Page Coloring
- Page Size
A Case for Large Pages

- Page table size is inversely proportional to the page size
  - memory saved
- Fast cache hit time easy when cache <= page size (VA caches);
  - bigger page makes it feasible as cache size grows
- Transferring larger pages to or from secondary storage, possibly over a network, is more efficient
- Number of TLB entries are restricted by clock cycle time,
  - larger page size maps more memory
  - reduces TLB misses

A Case for Small Pages

- Fragmentation
  - large pages can waste storage
  - data must be contiguous within page
- Quicker process start for small processes(??)
Superpages

• Hybrid solution: multiple page sizes
  – 8KB, 16KB, 32KB, 64KB pages
  – 4KB, 64KB, 256KB, 1MB, 4MB, 16MB pages

• Need to identify candidate superpages
  – Kernel
  – Frame buffers
  – Database buffer pools

• Application/compiler hints

• Detecting superpages
  – static, at page fault time
  – dynamically create superpages

• Page Table & TLB modifications