Page Tables

Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
The Problem

- **Space overhead of page tables**
  - A 32-bit address space with 4KB pages (4 bytes/PTE): \(2^{20} \times 4 = 4\text{MB} \) (per process)
  - A 64-bit address space with 8KB pages (8 bytes/PTE): \(2^{51} \times 8 = 2^{54} = 32\text{PB} \) (per process)

- **How can we reduce this overhead?**
  - Observation: Many invalid PTEs
  - Only need to map the portion of the address space actually being used which is a tiny fraction of entire address space
(Typical) Linear Page Table

Virtual address space

- code
- data
- heap
- heap

Physical memory

Stack

Huge waste!
Paging with Segmentation (I)

- A segment represents a region of valid address space
  - Segmentation:
    - Divide virtual address space into segments
    - Each segment can have variable length
  - Paging:
    - Divide each segment into fixed-sized pages
    - Each segment has a page table
    - Each segment tracks base (physical address) and limit of the page table for that segment

- Virtual address divided into three portions

<table>
<thead>
<tr>
<th>Seg #</th>
<th>Page number</th>
<th>Page offset</th>
</tr>
</thead>
</table>

SWE3004: Operating Systems | Spring 2016 | Jin-Soo Kim (jinsookim@skku.edu)
Paging with Segmentation (2)

- Example: Multics address translation
Paging with Segmentation (3)

- **Pros**
  - Can decrease the size of page tables
  - Segments can grow without any reshuffling
  - Can run process when some pages are swapped to disk
  - Increases flexibility of sharing: share either single page or entire segment

- **Cons**
  - Page tables potentially can be large
    - e.g. A large but sparse-used heaps will have a lot of waste
  - External fragmentation due to page tables
    - Each page table should be allocated contiguously
Multi-level Page Tables (1)
Multi-level Page Tables (2)

- Allow each page table to be allocated non-contiguously
- Virtual addresses have 3 parts

  - Outer page table: outer page number $\rightarrow$ secondary page table
  - Secondary page table: secondary page # $\rightarrow$ page frame #
### Multi-level Page Tables (3)

- **Example**
  - 32-bit address space, 4KB pages, 4 bytes/PTE
  - Want every page table fit into a page

<table>
<thead>
<tr>
<th>Outer page #</th>
<th>Secondary page #</th>
<th>Page offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>12</td>
</tr>
</tbody>
</table>

- **Outer page table**
- **Secondary page table**

- **Physical memory**
  - Page frame N
  - Page frame 0
  - Page frame 1
  - Page frame 2
  - Page frame 3
  - Page frame 4
  - Page frame 5
  - Page frame 6
Multi-level Page Tables (4)

- Address translation in Alpha AXP architecture
  - Three-level page tables
  - 64-bit address space divided into 3 segments (coded in bits 63/62)
    - seg0 (0x): user code
    - seg1 (11): user stack
    - kseg (10): kernel
  - Alpha 21064
    - Page size: 8KB
    - Virtual address: 43 bits
    - Each page table is one page long
Multi-level Page Tables (5)

- Address translation in Intel 64 architecture
  - 48-bit “linear” address $\rightarrow$ 52-bit physical address (4KB page)
Multi-level Page Tables (6)

- **Pros**
  - Compact while supporting sparse-address space
    - Page-table space in proportion to the amount of address space used
  - Easier to manage physical memory
    - Each page table usually fits within a page
  - Easier for hardware to walk through page tables
  - No external fragmentation

- **Cons**
  - More memory accesses on a TLB miss
  - More complex than a simple linear page-table lookup
Inverted Page Tables (1)

- Reverse mapping from PFN $\rightarrow$ <VPN, PID>
  - One entry for each page frame in physical memory
  - Entry consists of the virtual page number with information about the process that owns that page
  - Need to search through the table to find match
  - Use hashing to limit the search to one, or at most a few, page-table entries

- Pros & Cons
  - Decrease memory needed to store page tables: No need to have per-process page tables
  - Increase time needed to search the table on a TLB miss
Inverted Page Tables (2)
Paging Page Tables

- Store page tables in virtual address space
  - Cold (unused) page table pages can be paged out to disk
  - But, now addressing page tables requires translation
  - Outer page table is usually pinned into physical memory
  - Outer page table points to the virtual addresses (in the kernel address space) of secondary page tables
  - Need to handle nested page faults

- What if we page kernel code and data too?
(cf.) IA-32 VM Architecture