Lecture - 35
Virtual Memory

VM of process A

0 1 2

N-1

Main Memory (Cache)

Disk

Remember "block size" in cache memories?

Block \equiv Virtual Page in VM system.
Each Virtual Page is \( P = 2^p \) bytes in size.

\[ \begin{align*}
\text{eg. Virtual page size} &= 4 \text{ KB} = 4096 \text{ bytes} \\
&= 4 \times 1024 \text{ bytes} \\
&= 2^2 \times 2^{10} \text{ bytes} \\
&= 2^{12} \text{ bytes}
\end{align*} \]

Physical Page

Also \( P \) bytes in size.

\[ \text{ie. Virtual page size} = \text{Physical page size} \]

Blocks should be of the same size between 2 adjacent storage devices in the memory hierarchy.
3 types of virtual pages

- unallocated
  - pages that are not yet created by the VM system.
  - no data associated with them.
  - do not occupy any space on disk.

- cached
  - Allocated pages that are currently cached in physical memory.

- uncached
  - Allocated pages that are NOT cached in physical memory.
Virtual Pages (VPs) stored on disk:

- UA - Unallocated
- C - Cached
- UC - Uncached

Physical Pages (PPs) cached in DRAM:

- PPO
- PP1

Virtual Memory

Physical Memory

Diagram showing the mapping between virtual and physical memory, with UA, C, and UC states.
Main Memory (DRAM) as a Cache

1. Block size (Virtual Page size) = 4KB to 2MB
   - Miss penalty is large!

2. Associativity: fully associative

3. Replacement policy: More sophisticated than L1, L2, L3 caches!

4. Write policy: write-back!

PAGE TABLES!

<table>
<thead>
<tr>
<th>PTE</th>
<th>Valid</th>
<th>h-bit address field</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0x8079A2C1</td>
</tr>
</tbody>
</table>

Physical Memory

Virtual Memory (disk)
Page Table Entry

Page Hit
Read VP2!

Page Fault
Read VP3!

demand paging
VM as a tool for Memory Management

1. Simplifying linking
2. " loading
3. " sharing
4. " memory allocation.

VM as a tool for memory protection.

Virtual Address Spaces

Process : 
0  VPo
1  VP1
2  VP2

Process : 
0  VPo
1  VP1
2  VP2

Per process page table.
### Page-level memory protection

**Page tables with permission bits**

<table>
<thead>
<tr>
<th>Process i</th>
<th>SUP</th>
<th>READ</th>
<th>WRITE</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP0</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>PP6</td>
</tr>
<tr>
<td>VP1</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>PP2</td>
</tr>
<tr>
<td>VP2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>PP10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Process j</th>
<th>SUP</th>
<th>READ</th>
<th>WRITE</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP0</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>PP9</td>
</tr>
<tr>
<td>VP1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>PP6</td>
</tr>
<tr>
<td>VP2</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>PP7</td>
</tr>
</tbody>
</table>
Address Translation

N-element Virtual Address Space (VAS) \xrightarrow{\text{map.}} M-element Physical Address Space (PAS)

$$\text{MAP : VAS} \rightarrow \text{PAS \cup \emptyset}$$

$$\text{MAP}(A) = \begin{cases} 
A' & \text{if data at VA } A \text{ is present at PA } A' \text{ in PAS.} \\
\emptyset & \text{if "" "" NOT present in physical memory.}
\end{cases}$$

PTBR - Page Table Base Register (CPU Register)

Page table:

<table>
<thead>
<tr>
<th>Valid</th>
<th>PPN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Address translation with a page table

NOTE: PPO is identical to the VPO.
Page hit (handled entirely by the hardware)

CPU chip

Processor \(\xrightarrow{1}\text{VA}\) MMU

Cache/Memory

Page fault (handled by HW and OS Kernel)

PTEA - Page Table Entry Address

Page Fault Exception Handler

Processor \(\xrightarrow{1}\text{VA}\) MMU

Cache/Memory

Disk

1. The faulting instruction is restarted.
   * CPU sends VA again!