Controlling Memory Allocation to a Process

- Process \( P_i \) is allocated \( alloc_i \) number of page frames
- **Fixed memory allocation**
  - Fixes \( alloc \) statically; uses local page replacement
- **Variable memory allocation**
  - Uses global or local page replacement
  - If local replacement is used, VM manager periodically determines correct \( alloc \) value for a process
- May use working set model

**Definition 12.4 Working Set**

- \( WS_i(t,\Delta) \)-working set for process \( i \) at time \( t \) for window size \( \Delta \)
- \( WSS_i(t,\Delta) \)-size of the working set \( WS_i(t,\Delta) \)
- Sets \( alloc \) to \( WSS_i(t,\Delta) \)

Working Set Memory Allocator

- If \{ \( proc_k \) \} is the set of processes in memory, the degree of multiprogramming should be decreased if
  \[ \sum_k WSS_k > \#\text{frames} \]
- The degree of multiprogramming should be increased if
  \[ \sum_k WSS_k < \#\text{frames} \]
  and there exists a process \( g \) such that
  \[ WSS_g \leq (\#\text{frames} - \sum_k WSS_k) \]
- Performance of a working set memory allocator depends on the value of \( \Delta \)
  - \( \Delta \) too large \( \rightarrow \) overcommitment of memory to processes
  - \( \Delta \) too small \( \rightarrow \) undercommitment of memory to processes
Implementation of a Working Set Memory Allocator

• Expensive to determine $WS_i(t, \Delta)$ and $alloc_i$ at every time instant $t$
  – Solution: Determine working sets periodically
    • Working sets determined at end of an interval are used to decide values of $alloc$ for use during the next interval

<table>
<thead>
<tr>
<th>Process</th>
<th>$t=100$</th>
<th>$t=200$</th>
<th>$t=300$</th>
<th>$t=400$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WSS alloc</td>
<td>WSS alloc</td>
<td>WSS alloc</td>
<td>WSS alloc</td>
</tr>
<tr>
<td>$P_1$</td>
<td>14 14 12 14</td>
<td>14 14 13 13</td>
<td>14 14 13 13</td>
<td></td>
</tr>
<tr>
<td>$P_2$</td>
<td>20 20 24 24</td>
<td>11 11 25 25</td>
<td>11 11 25 25</td>
<td></td>
</tr>
<tr>
<td>$P_3$</td>
<td>18 18 18 18</td>
<td>70 70 18 18</td>
<td>70 70 18 18</td>
<td></td>
</tr>
<tr>
<td>$P_4$</td>
<td>10 0 10 0</td>
<td>10 0 10 0</td>
<td>10 0 10 0</td>
<td></td>
</tr>
</tbody>
</table>

Figure 12.21 Operation of a working set memory allocator.

Summary

• Basic actions in virtual memory using paging: *address translation* and *demand loading of pages*
  – Implemented jointly by
    • Memory Management Unit (MMU): Hardware
    • Virtual memory manager: Software
• Which page should VM manager remove from memory to make space for a new page?
  – *Page replacement algorithms* exploit *locality of reference*
    • *LRU* has *stack property*, but is expensive
    • *NRU* algorithms are used in practice
      – E.g., *clock algorithms*
• How much memory should VM manager allocate?
  – Use *working set model* to avoid *thrashing*