Virtual Memory – Part 2 (CSE 422S)

Ken Wong
Washington University
kenw@wustl.edu
www.arl.wustl.edu/~kenw

Global Replacement Algorithms (1)

- Example Page Reference Stream (String):
  2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2

- Assume
  - Fixed number of page frames shared by all processes
  - When free frame needed
    - All resident pages are candidates for eviction
  - Alternative is a Local Replacement policy
    - Each process has a working set of pages
    - Number of pages allocated to a process can vary over time

- Goal of global replacement algorithm
  - Select a good page to be replaced when a new page must be swapped into memory

Global Replacement Algorithms (2)

- Basic Algorithms
  - Optimal (Impractical)
    - Select the P for which the time to the next reference is the longest.
  - First-In, First-Out (FIFO)
    - Select the oldest P
  - Least Recently Used (LRU)
    - Select the P that hasn’t been referenced for the longest time in the past
  - Clock
    - Approximates LRU using a clock structure

Replacement Example (1)

<table>
<thead>
<tr>
<th>Time</th>
<th>Pages</th>
<th>OPT</th>
<th>FIFO</th>
<th>LRU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2, 3</td>
<td>2, 3</td>
<td>3, 1</td>
<td>3, 2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2, 4</td>
<td>1, 1</td>
<td>1, 1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3, 5</td>
<td>4, 4</td>
<td>4, 4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5, 5</td>
<td>5, 5</td>
<td>5, 5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5, 5</td>
<td>5, 5</td>
<td>5, 5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6, 6</td>
<td>6, 6</td>
<td>6, 6</td>
</tr>
</tbody>
</table>
### Optimal and FIFO Page Replacement

- **Optimal (Impractical)**
  - **Idea**: Replace the page for which the time to the next reference is the longest.
  - Impossible to implement, but serves as a reference point

- **First-In, First-Out (FIFO)**
  - **Idea**: Replace the page that has been in memory the longest (i.e., the oldest)
  - One of the simplest algorithms, but performs poorly
  - Treat page frames of a process as a circular buffer
  - Pages are removed in RR order
  - Implement as a pointer that cycles through the page frames of a process

### LRU Page Replacement

- **Least Recently Used (LRU)**
  - **Idea**: Replace the page that hasn’t been referenced for the longest time in the past
  - Does almost as well as optimal algorithm on some reference sequences
  - Difficult to implement in hardware
    - Time stamp each page and replace the oldest one
    - Use a stack with the most recently referenced page on top

### Clock Page Replacement

- **Clock (Second Chance Algorithm)**
  - **Idea**: Faster approximation of LRU
  - Select an unmarked frame in RR order
    - Cursor cycles through page frames looking for an unmarked page as the replacement page
  - When to **unmark** the page underneath the cursor:
    - If the page is marked, then go to next page
    - If all pages are marked, the cursor will return to the first one it unmarked
  - When to **mark** a page:
    - When loaded into main memory
    - When already in main memory and it has been referenced

- **Enhanced Clock (Third Chance Algorithm)**
  - **Scan 1**: Search for \((R, M) = (0, 0)\) for replacement
    - \((1, 1) \rightarrow (0, 1); (1,0) \rightarrow (0, 0); (0,1) \rightarrow (0,0*)\) requires disk write
  - **Scan 2**: Search for \((R, M) = (0, 0)\) for replacement
Replacement Algorithm Variations

- Counting Algorithms
  - Least Frequently Used (LFU): Select page with smallest reference count
  - Most Frequently Used (MFU): Select page with largest reference count
- Use Bit (U) and Modified Bit (M)
  - U = 1 \( \rightarrow \) Page has been referenced recently
  - M = 1 \( \rightarrow \) Page has been modified and needs to be written to disk
- Page Buffering
  - Replaced page is assigned to either the free page list (if M is 0) or the modified list (otherwise)
- Cluster page writes of modified pages

The Working Set Model

- \( \text{W}(t, \Delta) \) is the working set at virtual time \( t \) with a window size of \( \Delta \) and is:
  - Defined over the page reference string for each process
  - The set of pages that have been referenced in the time interval \( [t - \Delta, t] \).
- Example:
  - \( \text{W}(4,4) = \{0, 3, 8, 9\} \)
  - \( \text{W}(2,2) = \{0, 9\} \) W(15,5) = {0, 2, 9}
- \( \text{W}(t, \Delta) \) varies over time \( t \) even with a fixed window size \( \Delta \)

The Working Set Strategy

- The Strategy
  - Monitor \( \text{W}(t, \Delta) \) for each process
  - Periodically remove pages from the resident set of a process that are not in its \( \text{W}(t, \Delta) \)
  - Schedule a process only if its working set is in main memory
- Problems
  - The past doesn’t always predict the future
  - An exact measurement of \( \text{W}(t, \Delta) \) is impractical because it requires a time-ordered queue of pages.
  - The optimal value of \( \Delta \) is unknown

Example

Reference String (RS): \( (4, 3, 0, 2, 2, 3, 1, 2, 4, 2, 4, 0, 3) \)
- Working Set (\( \Delta = 4; x \) indicates page is in memory)

\[
\begin{array}{cccccccc}\hline
\text{Page} & 0 & 1 & 2 & 3 & 4 & \text{IN} & \text{OUT} \\
\hline
\text{IN} & \times & \times & \times & \times & \times & 2 & 0 \\
\text{OUT} & 3 & 4 & 0 & 1 & 2 & 4 & 3 \\
\text{IN} & \times & \times & \times & \times & \times & 4 & 0 \\
\text{OUT} & 1 & 3 & 2 & 0 & 3 & 0 & 3 \\
\text{IN} & \times & \times & \times & \times & \times & 2 & 0 \\
\text{OUT} & 3 & 4 & 0 & 1 & 2 & 4 & 3 \\
\end{array}
\]

PFF (F = 3): expand RSS if inter-fault time < 3

\[
\begin{array}{cccccccc}\hline
\text{Page} & 0 & 1 & 2 & 3 & 4 & \text{IN} & \text{OUT} \\
\hline
\text{IN} & \times & \times & \times & \times & \times & 1 & 0 \\
\text{OUT} & 1 & 3 & 2 & 0 & 3 & 0 & 3 \\
\end{array}
\]

From Bic & Shaw
Page-Fault Frequency (PFF) Algorithm

**Idea**: Adjust resident set size (RSS) according to page fault rate

**Basic Algorithm**
- Select a threshold $F$, the minimum time between page faults
- Mark each page that is referenced with a use-bit (U) of 1
- When a page fault occurs, compute the time $F'$ since the last page fault and adjust the resident set size:
  - $F' < F$: Add a page to the resident set
  - $F' \geq F$: Discard all pages with a use-bit (U) of 0, and shrink the resident set size
- Reset all use-bits after a page fault

**Flaw**: Poor performance during expanding transition periods

Variable-Interval Sampled WS

**Idea**: Deal with transition periods by sampling more frequently and discarding unused pages when there are more page faults

**Algorithm Parameters**
- $I$: Minimum duration of the sampling interval
- $I'$: Maximum duration of the sampling interval
- $F$: Number of page faults that are allowed to occur between sampling intervals

**Algorithm**
- Suspend process and scan the U-bits when:
  1) $T = I'$
  2) $I - T < I'$ and $\#Faults = F$
- RSS can only shrink at end of sampling interval; it either remains the same or increases within each interval

Practical Systems

- Machines typically don't support LRU or WS
  - Typically, have an R-bit (referenced)
- Portable kernel code
  - May not use all features of VM hardware
- Aging - a software solution
  - Based on NFU (Not Frequently Used) algorithm
  - OS scans all pages at *each clock interrupt* (10 msec)
    - Right shift each R-bit into its 8-bit age counter
    - Then, zero (reset) each R-bit
  - Replacement page is the one with the smallest age
  - NFU
    - Adds R-bit to age counter
    - Problem is that past behavior can incorrectly effect replacement

Aging Example

<table>
<thead>
<tr>
<th>R-bits</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Tick 0</td>
<td>10000000</td>
<td>11000000</td>
<td>11000000</td>
<td>11100000</td>
<td>11110000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tick 1</td>
<td>10000000</td>
<td>10000000</td>
<td>10000000</td>
<td>11000000</td>
<td>01100000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tick 2</td>
<td>10000000</td>
<td>01000000</td>
<td>00100000</td>
<td>00100000</td>
<td>00100000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tick 3</td>
<td>10000000</td>
<td>10000000</td>
<td>10100000</td>
<td>10100000</td>
<td>10100000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: Tanenbaum, Modern Operating Systems
**WS Clock Page Replacement**

- **R-bit = 1** if page has been referenced (read/write)
- **Age counter**
- **M-bit = 1** if page has been modified (write)
- Each page falls into one of four (R,M) classes
  - (0,0): Neither recently used nor dirty (**CLEAN**)
  - (0,1): Not used but dirty (want to convert to (0,0))
  - (1,0): Recently used and clean (maybe don't replace)
  - (1,1): Recently used and dirty (might be used again soon)
- **Replacement**
  - Page is clean (R=0,M=0) and **age counter < threshold**
    - Evict and reduce WS
  - Page is (R=0,M=1) ➔ Schedule write to disk
    - Limit number of writes
    - Avoids context switch
    - Hope to find clean page later in scan

**Belady’s Anomaly**

- FIFO page replacement can produce more page faults when given more frames

**Summary**

- **Optimal:** Not implementable, but useful benchmark
- **FIFO:** Might throw out important pages
- **LRU:** Excellent but difficult to implement in HW
- **WS:** Expensive to implement
- **Aging:** Efficient approximate LRU algorithm
- **WSClock:** Good, efficient algorithm
- **Interesting References**