Demand Paging

Code pages are stored in a memory-mapped file on the backing store
- some are currently in memory—most are not

Data and stack pages are also stored in a memory-mapped file

OS determines what portion of VAS is mapped in memory
- physical memory serves as cash for memory-mapped file on backing store

Demand Paging: Touching Valid but not Present Address

1. TLB Miss (HW managed)
2. Page Table walk
3. Page fault (Present bit P not set in Page Table)
4. Exception to kernel to run page-fault handler
5. Convert VA to file offset
6. Allocate page frame (evict page if needed)
7. Initiate disk block read into page frame
8. Disk interrupt when transfer completes
9. Set P to 1 and update PFN for page's PTE
10. Resume process at faulting instruction
11. TLB miss
12. Page Table walk—success!
13. TLB updated
14. Execute instruction

Allocating a Page Frame

When free frames fall below Low Watermark, do until they climb above High Watermark:
- Select “victim” page VP to evict (a policy question)
- Find all PTEs referring to frame VP maps to
  - if page frame was shared
  - Set P bit in each such PTE to 0
- Remove any TLB entries that included VP’s victim frame
  - the PTE they are caching is now invalid!
- Write changes to page back to disk

Transferring pages in bulk allows to reduce transfer time
Page Replacement

- Local vs Global replacement
  - **Local**: victim chosen from frames of process experiencing page fault
    - fixed allocation per process
  - **Global**: victim chosen from frames allocated to any process
    - variable allocation per process
- Many replacement policies
  - Random, FIFO, LRU, Clock, Working set, etc.
- Goal: minimizing number of page faults

How do we pick a victim?

- We want:
  - low fault-rate for pages
  - page faults as inexpensive as possible
- We need:
  - a way to compare the relative performance of different page replacement algorithms
  - some absolute notion of what a “good” page replacement algorithm should accomplish

Comparing Page Replacement Algorithms

- Record a trace of the pages accessed by a process
  - E.g. 3,1,4,2,5,2,1,2,3,4 (or c,a,d,b,e,b,a,b,c,b)
- Simulate behavior of page replacement algorithm on trace
- Record number of page faults generated

Optimal Page Replacement

- Replace page needed furthest in future

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Requests</strong></td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>e</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>b</td>
</tr>
<tr>
<td><strong>Page Frames</strong></td>
<td>0</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
</tr>
<tr>
<td><strong>Faults</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Time page needed next</strong></td>
<td>a = 7</td>
<td>b = 6</td>
<td>c = 9</td>
<td>d = 10</td>
<td>e = ∞</td>
<td>a = 11</td>
<td>b = 12</td>
<td>c = 13</td>
<td>d = 15</td>
<td>e = 15</td>
<td>a = 7</td>
</tr>
</tbody>
</table>
FIFO Replacement

- Replace pages in the order they come into memory

Assume:
- a @ -3
- b @ -2
- c @ -1
- d @ 0

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requests</td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>e</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>Frames</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page Frames</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Faults</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frames</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>c</td>
<td>c</td>
</tr>
</tbody>
</table>

For example...

FIFO Replacement

- 3 frames - 9 page faults!

Belady’s Anomaly

- 4 frames - 10 page faults!
Yes, but only for stack page replacement policies.

- set of pages in memory with n frames is a subset of set of pages in memory with n+1 frames.

Locality of Reference

- If a process accesses a memory location, then it is likely that
  - the same memory location is going to be accessed again in the near future (temporal locality)
  - nearby memory locations are going to be accessed in the future (spatial locality)

- 90% of the execution of a program is sequential
- Most iterative constructs consist of a relatively small number of instructions.

LRU: Least Recently Used

- Replace page not referenced for the longest time.

Implementing LRU

- Maintain a "stack" of recently used pages.

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requests</td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>e</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>Page Frames</td>
<td>0</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
</tr>
<tr>
<td>3</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>Faults</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
No-Locality Workload

- Workload references 100 unique pages over time
- 10,000 references
- Next page chosen at random

What do you notice?

80%-20% Workload

- 10,000 references, but with some locality
  - 80% of references to 20% of the pages
  - 20% of references to the remaining 80% of pages.

What do you notice?

Sequential-in-a-loop Workload

- 10,000 references
- We access 50 pages in sequence, then repeat, in a loop.

What do you notice?

Implementing LRU

- Add a (64-bit) timestamp to each page table entry
  - HW counter incremented on each instruction
  - Page table entry timestamped with counter when referenced
  - Replace page with lowest timestamp
Implementing LRU

- Add a (64-bit) timestamp to each page table entry
- HW counter incremented on each instruction
- Page table entry timestamped with counter when referenced
- Replace page with lowest timestamp

Approximate LRU through aging:
- Keep a k-bit tag in each table entry
- At every "tick":
  i) Shift tag right one bit
  ii) Copy Referenced (R) bit in tag
  iii) Reset Referenced bits to 0
- If needed, evict page with lowest tag

The Clock Algorithm

- Organize pages in memory as a circular list
- When page is referenced, set its reference bit R to 1
- On page fault, look at page the hand points:
  i) If R = 0:
     - Evict the page
     - Set R bit of newly loaded page to 1
  ii) Else (R = 1):
     - Clear R
     - Advance hand

The Second Chance Algorithm

- Dirty pages get "second chance" before eviction
- Synchronously replacing dirty pages is expensive!

Clock Page Replacement

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requests</td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>e</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>Page Frames</td>
<td>0</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>Faults</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Page table entries for resident pages:

- 1a, 1b, 1c, 1d

Hand clock:

- 0

[Start asynchronous transfer of dirty page to disk]
### Second Chance Page Replacement

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requests</td>
<td>c</td>
<td>a</td>
<td>w</td>
<td>d</td>
<td>b</td>
<td>w</td>
<td>e</td>
<td>b</td>
<td>a</td>
<td>w</td>
<td>b</td>
</tr>
<tr>
<td>Page Frames</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>Page Frames</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>Page Frames</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
</tr>
<tr>
<td>Page Frames</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>c</td>
</tr>
</tbody>
</table>

**Faults**

- Time 0: X
- Time 1: X
- Time 2: X

---

**Page table entries for resident pages**

- Page 01: a
- Page 02: a
- Page 03: b
- Page 04: c
- Page 05: d

**Hand clock:**

- Page 01: 01
- Page 02: 01
- Page 03: 01
- Page 04: 01
- Page 05: 01

**Async copy:**

- Page 01: 01
- Page 02: 01
- Page 03: 01
- Page 04: 01
- Page 05: 01

---

**Notes:**

- The table above illustrates a page replacement strategy called Second Chance Page Replacement. It shows how requests for different pages are handled over time, along with the page frames and page faults.

---

**Further Exploration:**

- This strategy is an improvement over basic page replacement algorithms by giving pages a second chance to be reused if they are not recently accessed.