Replacement policies for multiprogramming:

Hardware support for page replacement algorithms:

- Enhanced Second Chance
- Second Chance
- LRU approximations:

Today

Benefits of demand paging:

A good page replacement algorithm can reduce the number of page faults when a switch occurs.

- Processes can share memory more effectively, reducing the costs when a context switch occurs.
- Processes can start faster because they only need to load a few pages (for code and data).
- Processes can run without being fully loaded into memory.
- Virtual address space can be larger than physical address space.

Last Class: Demand Paged Virtual Memory
**Why?**

With LRU, increasing the number of frames always decreases the number of page faults.

<table>
<thead>
<tr>
<th>Frame 1</th>
<th>Frame 2</th>
<th>Frame 3</th>
<th>Frame 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>E</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
</tbody>
</table>

**LRU:**

Adding Memory with LRU

**FIFO:**

Does adding memory always reduce the number of page faults?

<table>
<thead>
<tr>
<th>Frame 1</th>
<th>Frame 2</th>
<th>Frame 3</th>
<th>Frame 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>E</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
</tbody>
</table>

**of page frames.**

With FIFO, the contents of memory can be completely different with a different number of page faults.
Page fault still requires a search through all the pages.

Faster, since setting a single bit on each memory access.

Approximate, since it does not guarantee a total order on the pages.

- On a page fault, the lowest numbered page is kicked out.
- High order bit.
- Always keep the lowest numbered page in the list.
- Maintain more than 1 bit, say 8 bits.

Additional Reference Bits: Maintain reference bits with each page.

Hardware Requirements: Maintain reference bits with each page.

Implementation of LRU

Avail. LRU

Avail. Replacements: LRU replacements require hardware

Perfection LRU:

- All implementations and approximations of LRU require hardware

Approximations of LRU

Approximations of LRU

Implementations of LRU

Implementations of LRU
What if all bits are 1?

Will it always find a page?

Simple hardware requirements:

• A 0 reference bit.
• Page fault is faster, since we only search the pages until we find one with

shift.

Fast, since setting a single bit on each memory access, and no need for a

the algorithm.

indicates if the page was used at all since the last time it was checked by

less accurate than additional-reference-bits, since the reference bit only

Second Chance Algorithm

1. OS keeps frames in a circular list.
2. On a page fault, the OS

Use a single reference bit per page.

Second Chance Algorithm (a.k.a. Clock)
0. Page is the same as the copy on disk
1. Page is modified (different from the copy on disk)

- Hardware keeps a modify bit (in addition to the reference bit)

- OS can give preference to paging out un-modified pages
  - OS need not write the page back to disk
  - If it is cheaper to replace a page that has not been written

Enhanced Second Chance

Why not partition pages into more than two categories?

One way to view the clock algorithm is as a crude partitioning into two categories:

Clock Example
- By the third pass, all the pages will be at (0,0).
- A (0,0) page is treated as on the first pass.
- A page changed to (0,0) replace this page.
- The page is being written out. waits for the I/O to complete and then remove the
- On the second pass, a page that was originally (0,1) or (1,0) might have been
- The hand goes completely around once; the page was (0,0).
- For pages with the reference bit set, the reference bit is cleared;
- Until the I/O completes. Clear the modified bit and continue the search
- I. Page with (0,0) replace the page.
- 2. Page with (0,1) initialize an I/O to write out the page. looks the page in memory
- 3. Page with (0,1) change this page.

The OS goes around at most three times searching for the (0,0) class.

Page Replacement in Enhanced Second Chance

Enhanced Second Chance

On a page fault, the OS searches for the first page in the lowest

element class.

write it out before replacing if

write out this page, but it might not be needed anymore.

write out this page, but it might not be needed anymore.

write out this page, but it might not be needed anymore.

write out this page, but it might not be needed anymore.

write out this page, but it might not be needed anymore.

3. (1,0) recently used and unmodified - probably will be used again soon, but OS needed

2. (0,0) neither recently used nor modified - not so good to replace, since the OS must

1. (0,1) neither recently used nor modified - replace this page.
- Disadvantages: Threshold might become even more likely (why?)
- Advantages: Flexible; efficient to implement process needs

The physical memory associated with a process can grow

**Global Replacement**: Put all pages from all processes in one pool so that

**Proportional Allocation**: Allocate more page frames to large processes.

What can we do in a multiprogrammed environment to limit thrashing?

- Results in a serious and very noticeable loss of performance.
- Causes page faults
- Memory access times approach disk access times since many memory references

To avoid this situation, they are still in use

**Threshold**: The memory is over-committed and pages are continuously

---

**Replacement Policies for Multiprogramming**

**Clock Example**
process and if the working set size grows dynamically adjust its allocation.

- **Advantages:** Freshing is less likely as processes only compete with itself.
  - Only need to suspend a process until overall memory demands decrease.
  - The time it takes to handle a page fault.

- **Goal:** The system-wide mean time between page faults should be equal to each process instead.
  - When page fault frequency > 2 second threshold, take away some page frames
  - If the page fault frequency > some threshold, give it more page frames.
  - When page fault frequency < track page fault frequency of working sets are expensive to compute. To compute track page fault frequency of

  | Per-Process Replacement |

- **Disadvantages:** The OS has to figure out how many pages to give each
  - More consistent performance independent of system load.

What happens if T is too small? Too big?
- T needs to be a whole lot bigger than 2 million instructions.
- 10ms = 2 million instructions
- 1 page fault = 10ms

How does the OS pick T?
- More formally, it is the set of all pages that a process referenced in the past 1 seconds.
- Informally, the working set is the set of pages the process is using right now set size?

- How do we figure out how many pages a process needs? It’s working

- Run only groups of processes that fit in memory, and kick out the rest.

- **Per-Process Replacement:** Each process has its own pool of pages.

| Replacement Policies for Multiprogramming |
Summary of Page Replacement Algorithms

- Unix and Linux use variants of Clock, Windows NT uses FIFO.

Page Sizes

- Larger page sizes (few processes that exhibit locality of references)
- Fewer page faults (less disk i/o)
- Amortizes disk overheads over a larger page
- Simpler page tables

Reasons for Large Pages:
- Higher degree of multiprogramming possible
- More effective memory use

Reasons for Small Pages:
- Better slow down than they need to. Reducing the number of page faults is critical to CPU speed is increasing faster than disk speed. As a result, page faults result in a severe performance hit. Internal fragmentation is less of a concern with abundant memory. Physical memory is cheap. As a result, page tables could get huge with small pages.

Page sizes are growing because:

- Power page faults (per processes that exhibit locality of references)