Lecture 15: memory management continued

- clock algorithm
- thrashing
- (time permitting) Disk
Rand, PIFO fast, easy, dumb
LRU, LFU require storage (timestamp/count per page) estimating whether something will be needed soon.

OPT perfect

requires HW support: do something on every access (TLB)

Instead, add a "use" bit:
- set when page accessed
- cleared by OS.
Clock algorithm:

- Use bit: ✓ (yes, page accessed)
- X (cleared, not accessed)

To evict: take next page, skipping over pages marked "in use".

- Bits are set when page used (automatic TLB update)
- Clear bits "a little while" before we read then
  - Clearing hand, set a fixed # of frames in the future, clear use bit when it passes a frame.

Simple, decent locality.

Effect:

- Page doesn't get evicted if it is used between when it is cleared and when eviction hand arrives.

\[ \Delta t \text{ (distance)} \]

\( \Delta t \) is large: in limit, all pages (or most) will be marked invalid; evicting takes a long time.

\( \Delta t \) is tiny (e.g., 1):
- Use bits always clear; FIFO
Thrashing
- page replacement / swapping work or it
- don't swap often.
- lots of swapping: terrible
- too much context for some frames:
  spend all time swapping, no useful progress, thrashing.
- thrashing caused by launching more processes than you have memory for.
- need to know how much memory a process is using (not allocated)
  - working set: set of pages that process has accessed recently (within some \( \Delta t \))
- compute total size of working sets of all processes, if bigger than # frames of physical memory: thrashing.
- if thrashing is detected: suspend a process until total working set size goes down
- which process? similar to CPU scheduling
Disk

Very different from memory:
- Big
  memory: GB
  disk: TB, PB

- Slow
  memory: ns
  disk: ms

- Persistent
  memory is volatile: gone when you end program
  disk is persistent: think about what to do when suddenly shut down.

- Sequential access is much faster than random access.
Disk (magnetic)

- Spindle
- Tracks
- Platters
- Read/Write head
- Arm

4 tracks form a "cylinder"

Sectors (8*4 KB data)
DOPS: SSTF not actually optimal.

Optimal algorithm: Shortest seek time first - at any point, consider seek time to all requests, service the closest one.

SSTF can be unfair
Faster than SCAN:

Elevator algorithm: put a direction service all requests in that direction, then round service all requests in other direction.

Still unfair to beds near end of the elevator.

SCAN algorithm (trades efficiency for fairness)

C-SCAN algorithm

C-LOOK

like elevator algorithm, only go as far as next request instead of to end of disk.