CS 423
Operating System Design:
Memory Wrap-Up

Tianyin Xu

It's an online course now.

- **The state is in a shelter-in-place state.**
- **Everything is online – so let's continue.**
  - **Siebel is closed.**
  - **DCL is closed.**
- **Stay strong and stay safe!**
  - **Find me if you have any difficulties/problems.**
  - **You can find me on SysNet slack (I'm a big "slacker")**
  - **Stay connected with your friends/family**
    - **Remote coffee/tea (buy a coffee machine)**
    - **Video games**
    - **Remote study group**
    - **Spend time on MPs ☺**

Midterm Grading

- **Sorry. We are still working on it.**
  - **We finished 75%…**
  - **We needed to physically exchange the papers in an evening..**
  - **ETA: End of this week (or beginning of next week)**

MP1 Grading

- **Grading is out (pushed to your VMs).**
- **Statistics:**

Average	**8.506944444**
Standard Dev	**2.408162551**
Minimum	**0**
Maximum	**10**

- **Regrading requests are still open.**
  - **Please go to the TA's (virtual) office hour**
  - **Do not rely on emails.**

Autograder

- **The way it works is that you send us the VM/code and we run an autograder.**
- **We can't tell you what and how autograder is testing or is implemented.**
  - **Please don't bother to interpret, it makes no sense.**
  - **We will try to remove all the side channels.**
- **Autograder cannot find all bugs and has false positives.**
  - **That's why we are taking efforts to manually regrade.**
- **Please do follow the rubrics of the PDF.**

Heads up

- **ALL course could be changed to PASS/FAIL.**
  - **Still under discussion**
- **GPA is no longer a thing.**
- **Learning is the only purpose for courses.**
  - **This is how grad school looks like.**

A Recap of Virtual Memory

- **Let's use Jamboard**
  - **Virtual and Physical address**
    - **Illusion: each process has its own memory**
  - **Translation**
    - **Page table**
    - **TLB**
    - **How translation works (TLB miss, page fault)**
    - **Huge pages**
  - **Paging**
    - **Illusion: each process has infinite amount of memory**

Page Replacement Strategies

- **Reference string**: the memory reference sequence generated by a program.
- **Paging** – moving pages to (from) disk
- **Optimal** – the best (theoretical) strategy
- **Eviction** – throwing something out
- **Pollution** – bringing in useless pages/lines

Page Replacement Strategies

- **The Principle of Optimality**
  - Replace the page that will not be used the most time in the future.
- **Random page replacement**
  - Choose a page randomly
- **FIFO - First in First Out**
  - Replace the page that has been in primary memory the longest
- **LRU - Least Recently Used**
  - Replace the page that has not been used for the longest time
- **LFU - Least Frequently Used**
  - Replace the page that is used least often
- **Second Chance**
  - An approximation to LRU.

Principle of Optimality

- Description:
  - Assume that each page can be labeled with the number of instructions that will be executed before that page is first referenced, i.e., we would know the future reference string for a program.
  - Then the optimal page algorithm would choose the page with the highest label to be removed from the memory.
- Impractical because it needs to know future references

Optimal Example

12 references,
7 faults

Page Refs	3 Page Frames			
	Fault?	Page Contents		
A	yes	A		
B	yes	B	A	
C	yes	C	B	A
D	yes	D	B	A
A	no	D	B	A
B	no	D	B	A
E	yes	E	B	A
A	no	E	B	A
B	no	E	B	A
C	yes	C	E	B
D	yes	D	C	E
E	no	D	C	E

FIFO

12 references,
9 faults

Page Refs	3 Page Frames			
	Fault?	Page Contents		
A	yes	A		
B	yes	B	A	
C	yes	C	B	A
D	yes	D	C	B
A	yes	A	D	C
B	yes	B	A	D
E	yes	E	B	A
A	no	E	B	A
B	no	E	B	A
C	yes	C	E	B
D	yes	D	C	E
E	no	D	C	E

Average Paging Behavior

As number of page frames increases, we would generally expect the number of page faults to decrease…

… that is, until Bélády's anomaly was observed!

Belady's Anomaly (FIFO)

FIFO with 4 physical pages

12 references, 10 faults

As the number of page frames increase, so does the fault rate.

Page Refs	4 Page Frames				
	Fault?	Page Contents			
A	yes	A			
B	yes	B	A		
C	yes	C	B	A	
D	yes	D	C	B	A
A	no	D	C	B	A
B	no	D	C	B	A
E	yes	E	D	C	B
A	yes	A	E	D	C
B	yes	B	A	E	D
C	yes	C	B	A	E
D	yes	D	C	B	A
E	yes	E	D	C	B

FIFO w/ Page #'s 3 vs. 4

Page Refs	3 Page Frames			
	Fault?	Page Contents		
A	yes	A		
B	yes	B	A	
C	yes	C	B	A
D	yes	D	C	B
A	yes	A	D	C
B	yes	B	A	D
E	yes	E	B	A
A	no	E	B	A
B	no	E	B	A
C	yes	C	E	B
D	yes	D	C	E
E	no	D	C	E

Page Refs	4 Page Frames				
	Fault?	Page Contents			
A	yes	A			
B	yes	B	A		
C	yes	C	B	A	
D	yes	D	C	B	A
A	no	D	C	B	A
B	no	D	C	B	A
E	yes	E	D	C	B
A	yes	A	E	D	C
B	yes	B	A	E	D
C	yes	C	B	A	E
D	yes	D	C	B	A
E	yes	E	D	C	B

Belady's Anomaly (FIFO)

- Why??? **Increasing the number of page frames affects the order in which items are removed.**
- For certain memory access patterns, this can actually increase the page fault rate!
- Belay's Anomaly is *reference string dependent;* intuition about increasing page count still holds in general case.

FIFO w/ Page #'s 3 vs. 4

Page Refs	3 Page Frames			
	Fault?	Page Contents		
A	yes	A		
B	yes	B	A	
C	yes	C	B	A
D	yes	D	C	B
A	yes	A	D	C
B	yes	B	A	D
E	yes	E	B	A
A	no	E	B	A
B	no	E	B	A
C	yes	C	E	B
D	yes	D	C	E
E	no	D	C	E

Page Refs	4 Page Frames				
	Fault?	Page Contents			
A	yes	A			
B	yes	B	A		
C	yes	C	B	A	
D	yes	D	C	B	A
A	no	D	C	B	A
B	no	D	C	B	A
E	yes	E	D	C	B
A	yes	A	E	D	C
B	yes	B	A	E	D
C	yes	C	B	A	E
D	yes	D	C	B	A
E	yes	E	D	C	B

FIFO w/ Page #'s 3 vs. 4

Page Refs	3 Page Frames			
	Fault?	Page Contents		
A	yes	A		
B	yes	B	A	
C	yes	C	B	
D	yes	D	C	
A	yes	A	D	C
B	yes	B	A	D
E	yes	E	B	A
A	no	E	B	A
B	no	E	B	A
C	yes	C	E	B
D	yes	D	C	E
E	no	D	C	E

Page Refs	4 Page Frames			
	Fault?	Page Contents		
		A		
		B	A	
		C	B	A
A	no	D	C	B	A
B	no	D	C	B	A
E	yes	E	D	C	B
A	yes	A	E	D	C
B	yes	B	A	E	D
C	yes	C	B	A	E
D	yes	D	C	B	A
E	yes	E	D	C	B

$\text{Cache}_3 \nsubseteq \text{Cache}_4$!!!!

LRU

12 references,
10 faults

Page Refs	3 Page Frames			
	Fault?	Page Contents		
A	yes	A		
B	yes	B	A	
C	yes	C	B	A
D	yes	D	C	B
A	yes	A	D	C
B	yes	B	A	D
E	yes	E	B	A
A	no	A	E	B
B	no	B	A	E
C	yes	C	B	A
D	yes	D	C	B
E	yes	E	D	C

Least Recently Used (LRU) Issues

- How to track "recency"?
  - use time
    - record time of reference with page table entry
    - use counter as clock
    - search for smallest time.
  - use stack
    - remove reference of page from stack (linked list)
    - push it on top of stack
- both approaches require large processing overhead, more space, and hardware support.

Second Chance

- Only one reference bit in the page table entry.
  - 0 initially
  - 1 When a page is referenced
- Pages are kept in FIFO order using a circular list.
- Choose "victim" to evict
  - Select head of FIFO
  - If page has reference bit set, reset bit and select next page in FIFO list.
  - keep processing until you reach page with zero reference bit and page that one out.
- System V uses a variant of second chance

Second Chance Example

12 references
9 faults

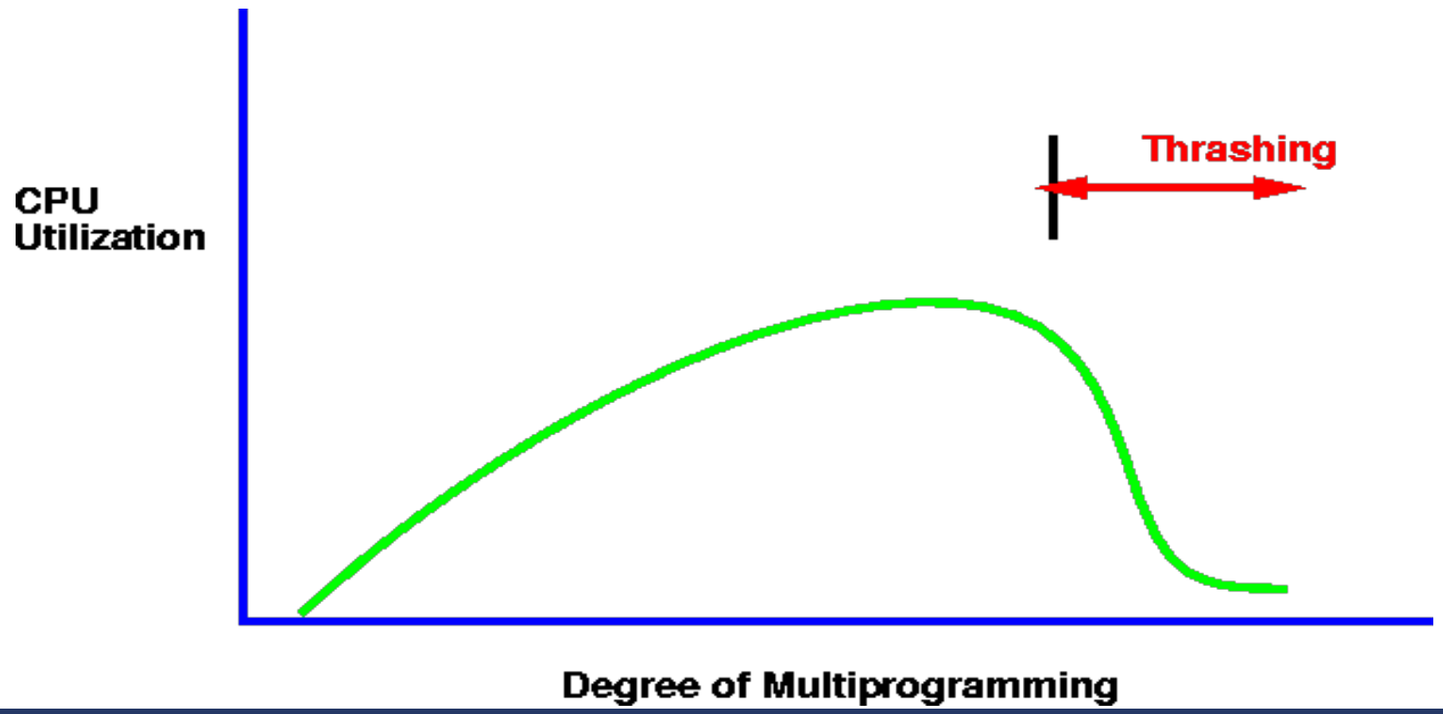
Page Refs	3 Page Frames			
	Fault?	Page Contents		
A	yes	A•		
B	yes	B•	A•	
C	yes	C•	B•	A•
D	yes	D•	C	B
A	yes	A•	D•	C
B	yes	B•	A•	D•
E	yes	E•	B	A
A	no	E•	B	A•
B	no	E•	B•	A•
C	yes	C•	E	B
D	yes	D•	C•	E
E	no	D•	C•	E•

Thrashing

- Computations have locality.
- As page frames decrease, the page frames available are not large enough to contain the locality of the process.
- The processes start faulting heavily.
- Pages that are read in, are used and immediately paged out.

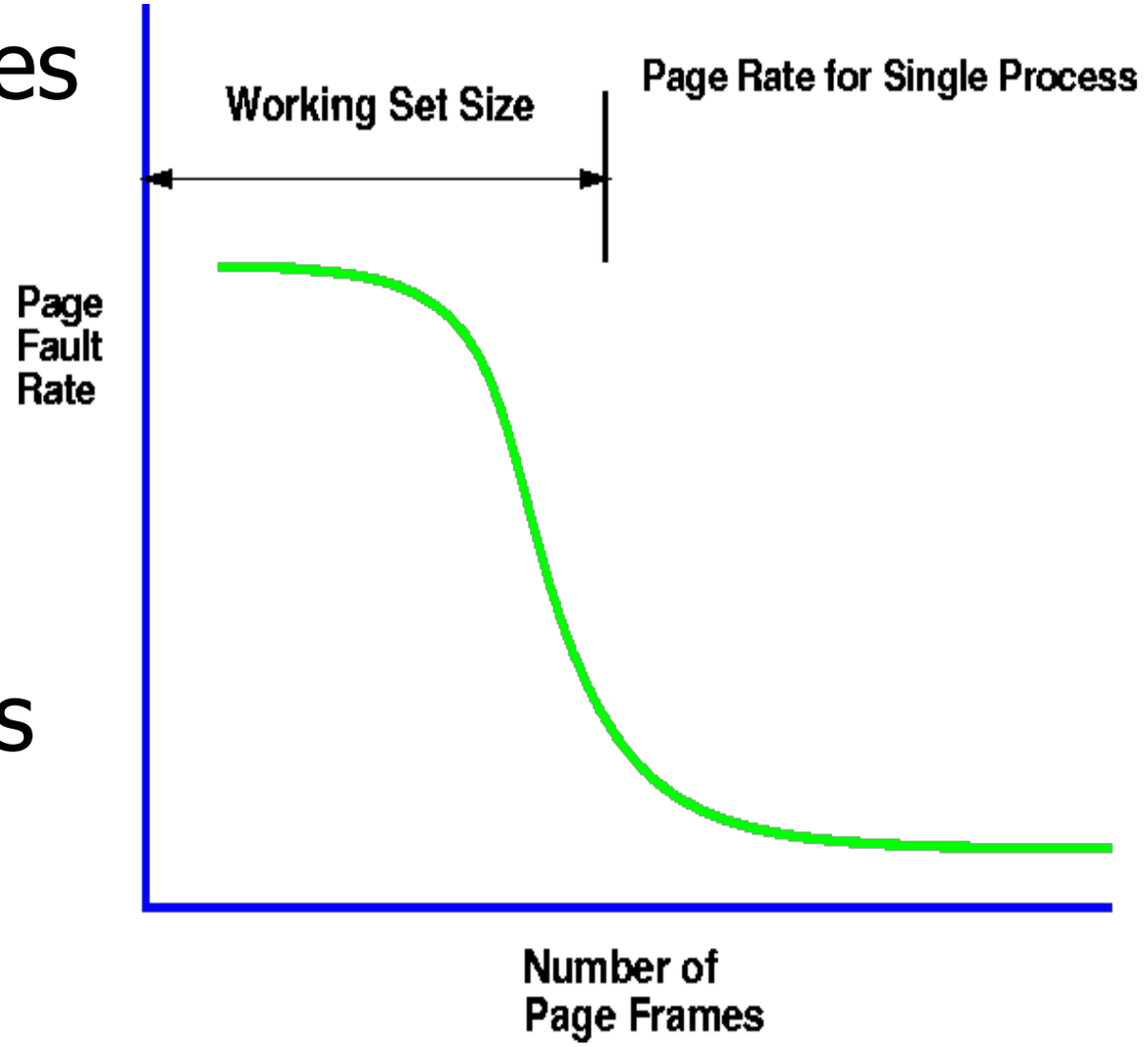
Thrashing & CPU Utilization

- As the page rate goes up, processes get suspended on page out queues for the disk.
- the system may try to optimize performance by starting new jobs.
- starting new jobs will reduce the number of page frames available to each process, increasing the page fault requests.
- system throughput plunges.



Working Set

- the working set model assumes locality.
- **the principle of locality states that a program clusters its access to data and text temporally**.
- As the number of page frames increases above some threshold, the page fault rate will drop dramatically.



Question

Why not use very large pages to reduce page faults?

Paging Terminology

- **Reference string**: the memory reference sequence generated by a program.
- **Paging** – moving pages to (from) disk
- **Optimal** – the best (theoretical) strategy
- **Eviction** – throwing something out
- **Pollution** – bringing in useless pages/lines

Page Size Considerations

- Small pages
  - Reason:
    - Locality of reference tends to be small (256)
    - Less fragmentation
  - Problem: require large page tables
- Large pages
  - Reason
    - Small page table
    - I/O transfers have high seek time, so better to transfer more data per seek
  - Problem: Internal fragmentation, needless caching