

Review

- Page Table
 - Paging mapping hardware
- TLB: cache of page table
 - TLB miss
- Multi-level Paging
- Inverted Page Table
- Sharing and Protection

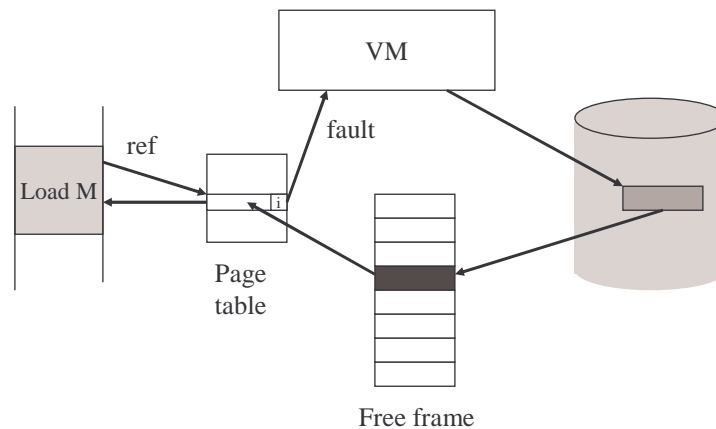
Paging Policies

- Fetch Strategies
 - When should a page be brought into primary (main) memory from secondary (disk) storage.
- Placement Strategies
 - When a page is brought into primary storage, where should it be put?
- Replacement Strategies
 - Which page now in primary storage is to be evicted from primary storage when some other page is to be brought in and there is not enough room.

Demand Paging

- Algorithm: NEVER bring a page into primary memory until its needed.
 1. Page fault
 2. Check if a valid virtual memory address. Kill job if not.
 3. If valid reference, check if its cached in memory already (perhaps for other processes.) If so, skip to 7).
 4. Find a free page frame.
 5. Map address into disk block and fetch disk block into page frame. Suspend user process.
 6. When disk read finished, add vm mapping for page frame.
 7. If necessary, restart the faulted instruction.

Demand Paging Example



Page Replacement

1. Find location of page on disk
2. Find a free page frame
 1. If there is a free page frame, use it
 2. Otherwise, select a page frame using the page replacement algorithm
 3. Write the selected modified page to the disk and update any necessary tables
3. Read the requested page from the disk.
4. Restart the faulted instruction.

Issue: Eviction

- Hopefully, kick out a less-useful page
 - Dirty pages require writing, clean pages don't
 - Hardware has a dirty bit for each page frame indicating this page has been updated or not
 - Where do you write? To "swap space"
- Goal: kick out the page that's least useful
- Problem: how do you determine usefulness?
 - Heuristic: temporal locality exists
 - Kick out pages that aren't likely to be used again

Terminology

- **Reference string**: the memory reference sequence generated by a program.
- **Paging** – moving pages from (to) 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 again the farthest 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
 - An approximation to LRU
- **NRU - Not Recently Used**
 - Replace the page that is not used recently
- **Working Set**
 - Keep in memory those pages that the process is actively using.

Principal 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.
- This algorithm provides a basis for comparison with other schemes.
- Impractical because it needs future references
- If future references are known
 - should not use demand paging
 - should use pre-paging to allow paging to be overlapped with computation.

An Example for Optimality

12 references,
7 faults

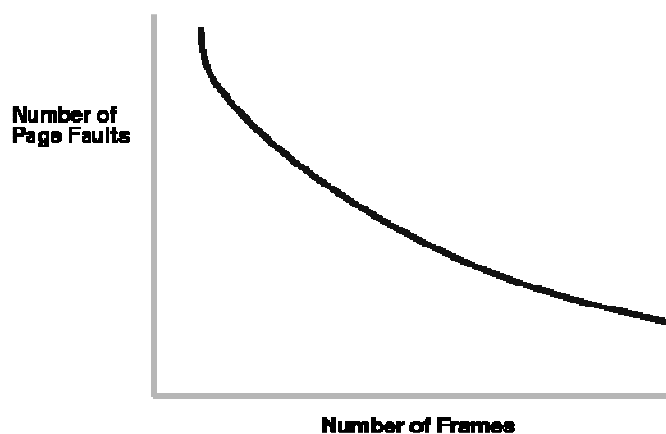
Page Refs	3 Page Frames			
	Fault?	Page Contents		
A	yes	A		
B	yes	B	A	
C	yes	C	B	A
D	yes	D	B	A
A	no	D	B	A
B	no	D	B	A
E	yes	E	B	A
A	no	E	B	A
B	no	E	B	A
C	yes	C	E	B
D	yes	D	C	E
E	no	D	C	E

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	

Expected Paging Behavior with Increasing Number of Page Frames



Belady's Anomaly (for 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		
R	yes	R	A	
C	yes	C	B	A
D	yes	D	C	B
A	no	D	C	B
B	no	D	C	B
E	yes	E	D	C
A	yes	A	E	D
R	yes	R	A	E
C	yes	C	B	A
D	yes	D	C	B
E	yes	E	D	C

LRU

12 references,
10 faults

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
B	yes	B	A
E	yes	E	B
A	no	A	E
R	no	R	A
C	yes	C	B
D	yes	D	C
E	yes	E	D

LRU and Anomalies

LRU, 4
physical pages
12 references,
8 faults

Anomalies
cannot
occur,
why?

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	no	A	D	C
B	no	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

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.