CS323 Operating Systems
Memory Management V

Yuanyuan Zhou

Lecture 18

3/3/2003

Content of this lecture

- Administrative announcements
- Page replacement algorithms
- Multi-program memory management
- Summary

Administrative

- Quiz2
- MP2
- Midterm1
  - Close-notes, close-book
  - No calculator

Review

- Inverted page table
- Multi-level page table
- Demand paging
- Page Replacement
  - Optimal

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 reach page with zero reference bit and page that one out.
- system v, r4 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*

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*

Page Classes

1.(0,0) neither referenced nor dirtied
2.(0,1) not referenced (recently) but dirtied
3.(1,0) referenced but clean
4.(1,1) referenced and dirtied

- select a page from lowest class
- if conflict, use random or FIFO.

Ad Hoc Techniques

- Separate Page Out from Page In.
  - Keep a pool of free frames.
  - When a page is to be replaced, use a free frame.
  - Read the faulting page and restart the faulting process while page out is occurring.
- Write dirty pages to disk whenever the paging device is free and reset the dirty bit. This allows page replacement algorithms to replace clean pages.
- Cache paged out pages in primary memory.
  - Page out dirty pages as before.
  - Return pages to a free pool but remember which page frame they are.
  - If system needs to map page in again, reuse page.
  - If system needs to page in data, choose any page in pool.
  - System V, R4 implements this strategy

Frame Allocation: Minimum

- How are the page frames allocated to individual virtual memories in a multi programmed environment?
- The simple case would be to allocate a minimum number of frames per process.
  - most instructions require two operands
  - include an extra page for paging out and one for paging in.
  - moves and indirection instructions might require more.

Equal Allocation

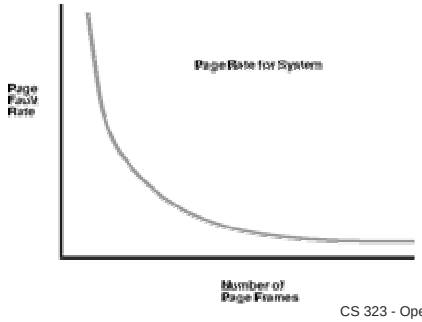
- allocate an equal number of frames per job
  - but jobs use memory unequally
  - high priority jobs have same number of page frames and low priority jobs
  - degree of multiprogramming might vary

Proportional Allocation

- allocate a number of frames per job proportional to job size
  - Challenge: how do you determine job size: by run command parameters or dynamically?

Page Fault Rate Curve

As page frames per VM space decrease, the page fault rate increases.

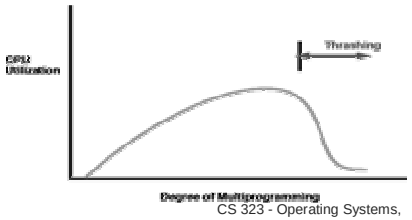


Thrashing

- 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.

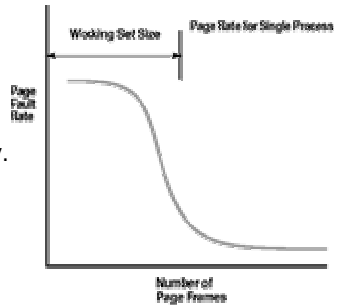
Thrashing and CPU Utilization

- 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*.



Working Set in Action

- Algorithm
  - if # free page frames > working set of some suspended process, then activate process and map in all its working set
  - if working set size of some process increases and no page frame is free, suspend process and release all its pages
- moving window over reference string used for determination.
- keeping track of working set.

Working Set Example

Window size is Δ

12 references, 8 faults

Page Refs	Δ = 4 References				
	Fault?	Page Contents			
A	yes	A			
B	yes	A	B		
C	yes	A	B	C	
D	yes	A	B	C	D
A	no	A	B	C	D
B	no	A	B	C	D
E	yes	A	B	D	E
A	no	A	B	E	
B	no	A	B	E	
C	yes	A	B	C	E
D	yes	A	B	C	D
E	yes	B	C	D	E

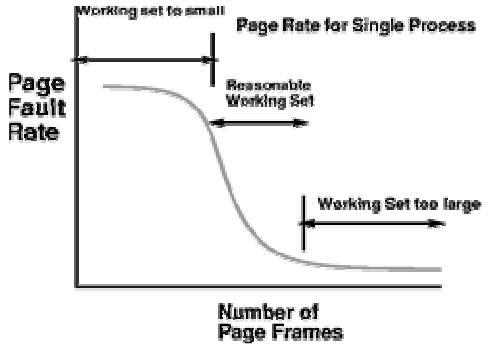
Working Set Solution

- Approximate working set model using timer and reference bit.
- Set timer to interrupt after approximately x references, τ.
- Remove pages that have not been referenced and reset reference bit.

Page Fault Frequency Version of Working Set

- Assume that if the working set is correct there will not be many page faults.
- If page fault rate increases beyond assumed knee of curve, then increase number of page frames available to process.
- If page fault rate decreases below foot of knee of curve, then decrease number of page frames available to process.

Page Fault Frequency Version of Working Set



WSClock Page Replacement Algorithm

- Carr and Hennessey, 1981
- Used very widely (Linux)
- Circular list as in the clock algorithm.
  - Initially, list is empty.
  - As pages are loaded into memory, pages are added to the list
  - Each entry contains *Time of last use* as well as reference and dirty bits.
- Algorithm:
  - At each page fault, the page pointed to by the clock hand is examined.
  - (Step 1) If reference bit is 1, then the page has been reference during the current tick, so it is in working set and not ideal candidate to remove.
  - If reference bit is 0, then check if it is in working set window (i.e., if Current Time minus Time of last use is less than the working set window size time).
  - If page is not in the working set and the page is clean, then replace it.
  - If the page is not in working set and page is dirty, request write to disk, and move on to the next page in the circular list.
  - Repeat (Step 1).

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

Summary

- protection hardware
- example systems
- fragmentation
- paging policies
- random
- optimal
- FIFO

Reminder

- Quiz2
- Mp2