Sean Marino

CSC 4103 OS

Chen

21 March 2017

Homework 3

1) Explain the difference between internal and external fragmentation. Explain the fragmentation issues with paging.

External fragmentation is the free space holes that are generated in memory or disk space. Internal fragmentation is any space that is wasted within each allocated block because of rounding up from the actual requested allocation. The size difference is memory internal to a partition, but not being used. Expanding beyond initial allocation can cause paging to become heavily fragmented. If a hard drive isn't defragmented, the performance suffers because of insufficient contagious space.

2) What is the effect of allowing two entries in a page table to point to the same page frame in memory? Explain how this effect could be used to decrease the amount of time needed to fork a child process.

User's are able to share data this way. Sharing large programs can save a lot of memory space.
Forking large amount of memory could be effected by having different page tables point ot he same memory location.

3) On a system with paging, a process cannot access memory that it does not own; why? How could the operating system allow access to other memory? Why should it or should it not?

If the page is not available in the page table, there is no way for a process to refer to its page. The process is limited to accessing only those pages that are allocated to the process. In order to allow access, an OS needs to allow entries to non-process memory to be added to the processes page table. If two or more processes need to exchange data, this is useful.

4) Compare paging with segmentation with respect to the amount of memory required by the address translation structures in order to convert virtual addresses to physical addresses.

Segmentation only needs two registers for maintaining the base of the segment and the extent of the segment, while paging needs one entry per page which proves the page address for each page.

5) What is the copy-on-write feature and under what circumstances is it beneficial to use this feature? What is the hardware support required to implement this feature?

The copy-on-write lets both parent and child processes share the same pages in memory. This allows more efficient process creation as only modified pages are copied. In each memory access, the page table needs to check whether the page is protected, and if it is a trap occurs so that the OS can solve the problem.

6) Under what circumstances do page faults occur? Describe the actions taken by the OS when a page fault occurs. Explain why page fault is critical to system performance.

If a page that is not in the main memory is accessed a page fault occurs. The program is aborted if it is invalid. If valid, a free frame is located. If there is no free frame, a victim page is freed from the frame. System performance is degraded because of page faults and can cause thrashing. Thrashing occurs if a process is too busy swapping pages in and out which is critical to a systems performance.

7) Consider the parameter Delta used to refine the working-set window in the working-set model. What is the effect of setting Delta to a small value on the page fault frequency and the number of active (non-suspended) processes currently executing in the system? What is the effect when Delta is set to a very high value?

When delta is set to a small value, the set of pages for a process may be underestimated which may allow a process to be scheduled even though all of its required pages are not resident. If this is the circumstance, page faults could occur. If delta was set to a large value, the result is overestimated and could prevent many processes from being scheduled. Is a process is scheduled, it is unlikely to generate any page faults because the resident set is overestimated.

8) What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to estimate this problem?

Thrashing is caused if a process is too busy swapping pages in and out, which will force the system to continuously page fault. By detecting the level of CPU utilization as compared to the level of multiprogramming. The problem can be resolved by reducing the level of multiprogramming.

9) Consider a paging system with the single-level page table stored in memory

a) If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?

400 nanoseconds

b) If we add associative registers, and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers take zero time if the entry is there.)

EMAT = .75 * 200 + .25 * 400 = **250 nanoseconds**

10) Consider the following page reference string:

1,2,3,4,2,1,5,6,2,1,2,3

Assuming the memory has 3 empty frame, illustrate the memory content for each page access and give the number of page faults that would occur for the LRU, FIFO, and Optimal replacement algorithms.

LRU

1	2	3	4	2	1	5	6	2	1	2	3
1	1	1	4		4	5	5	5	1		1
	2	2	2		2	2	6	6	6		3
		3	3		1	1	1	2	2		2

10 Page Faults
2 Hits

FIFO

1	2	3	4	2	1	5	6	2	1	2	3
1	1	1	4		4	4	6	6	6		3
	2	2	2		1	1	1	2	2		2
		3	3		3	5	5	5	1		1

10 Page Faults
2 Hits

Optimal

1	2	3	4	2	1	5	6	2	1	2	3
1	1	1	1			1	1				3
	2	2	2			2	2				2
		3	4			5	6				6

7 Page Faults
5 Hits