CSE 421/521 – Operating Systems **Sassi Kesavan**
Fall 2018 - Homework Assignment #3 **UBIT: sassikes**

Problem 1:

Explain the difference between internal and external fragmentation. Suggest ways to prevent each.

Difference between Internal Fragmentation and External Fragmentation

Fragmentation occurs when memory is allocated and returned to the system. As this occurs, free memory is broken up into small chunks, often too small to be useful. External fragmentation occurs when there is sufficient total free memory to satisfy a memory request, yet the memory is not contiguous, so it cannot be assigned. Some contiguous allocation schemes may assign a process more memory than it actually requested (i.e. they may assign memory in fixed-block sizes). Internal fragmentation occurs when a process is assigned more memory than it has requested and the wasted memory fragment is internal to a process.

Ways to Avoid

External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic. The internal fragmentation can be reduced by effectively assigning the smallest partition but large enough for the process.

Problem 3:

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.

Paging requires more memory overhead to maintain the translation structures. Segmentation requires just two registers per segment: one to maintain the base of the segment and the other to maintain the extent of the segment. Paging on the other hand requires one entry per page, and this entry provides the physical address in which the page is located.

Problem 5:

Why are segmentation and paging sometimes combined into one scheme?

Segmentation and paging are often combined in order to improve upon each other. Segmented paging is helpful when the page table becomes very large. A large contiguous section of the page table that is unused can be collapsed into a single-segment table entry with a page-table address of zero. Paged segmentation handles the case of having very long segments that require a lot of time for allocation. By paging the segments, we reduce wasted memory due to external fragmentation as well as simplify the allocation.

Problem 6:

A certain computer provides its users with a virtual-memory space of 2^{32} bytes. The computer has 2^{18} bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4096 bytes. A user process generates the virtual address 11123456. Explain how the system establishes the corresponding physical location. Distinguish between software and hardware operations.

The virtual address in binary form is0001 0001 0001 0010 0011 0100 0101 0110Since the page size is 2^12, the page table size is 2^20. Therefore, the low-order 12 bits "0100 0101 0110" are used as the displacement into the page, while the remaining 20 bits "0001 0001 0001 0010 0011" are used as the displacement in the page table.

Problem 8:

Discuss situations under which the least frequently used (LFU) page-replacement algorithm generates fewer page faults than the least recently used page replacement (LRU) algorithm. Also discuss under what circumstance does the opposite holds.

In the LFU algorithm, if any of the page that is least frequently used do not occur in the future then this algorithm will perform well.

In the LRU algorithm, if any of the page that is least recently used do not occur in the future then this algorithm will perform well.

For example,

Consider the following sequence of memory accesses in a system that can hold four pages in memory: 1 1 2 3 4 5 1. When page 5 is accessed, the least frequently used page-replacement algorithm would replace a page other than 1, and therefore would not incur a page fault when page 1 is accessed again. On the other hand, for the sequence "1 2 3 4 5 2," the least recently used algorithm performs better.

Problem 9:

Consider a demand-paging system with the following time-measured utilizations:

• CPU utilization 20%

• Paging disk 97.7%

• Other I/O devices 5%

Which (if any) of the following will (probably) improve CPU utilization? Explain your answer

a. Install a faster CPU->NO, a faster CPU reduces the CPU utilization further since the CPU will spend more time waiting for a process to enter in the ready queue.

b. Install a bigger paging disk->NO, the size of the paging disk does not affect the amount of memory that is needed to reduce the page faults.

c. Increase the degree of multiprogramming-> NO, If you increase the degree of multiprogramming the utilization will the more that it is in the current.

d. Decrease the degree of multiprogramming-> YES, by suspending some of the processes, the other processes will have more frames in order to bring their pages in them, hence reducing the page faults.

e. Install more main memory->Likely, more pages can remain resident and do not require paging to or from the disks (i.e. would depend on the page replacement algorithm you are using and the page reference sequence).

f. Install a faster hard disk or multiple controllers with multiple hard disks->Likely, faster response and more throughput to the disks, the CPU will get more data more quickly

g. Add prepaging to the page fetch algorithms-> Likely, the CPU will get more data faster, so it will be more in use.

h. Increase the page size-> Likely, Increasing the page size will result in fewer page faults

Problem 2:

Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)? Which algorithm makes the most efficient use of memory?

Problem 4:

Consider a paging system with the page table stored in memory. a) If a memory reference takes 200 nanoseconds, how long does a paged memory reference take? 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 takes zero time, if the entry is there.)

Problem 7:

Assume we have a demand-paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified, and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 200 nanoseconds?

Problem 10:

A page-replacement algorithm should minimize the number of page faults. We can do this minimization by distributing heavily used pages evenly over all of memory, rather than having them compete for a small number of page frames. We can associate with each page frame a counter of the number of pages that are associated with that frame. Then, to replace a page, we search for the page frame with the smallest counter.

a. Define a page-replacement algorithm using this basic idea. Specifically address the problems of (1) what the initial value of the counters is, (2) when counters are increased, (3) when counters are decreased, and (4) how the page to be replaced is selected.

b. How many page faults occur for your algorithm for the following reference string, for four page frames?

1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2.

c. What is the minimum number of page faults for an optimal page replacement strategy for the reference string in part b with four page frames?