LINJE: E, DI, EI, I, li m fl

DATUM: 8 januari 2001

TID: 14.00 – 18.00

Hjälpmedel: Engelsk-Svensk-Engelsk ordbok
Miniräknare

Antal uppg.: 5
Antal poäng: 30

Betygsgränser: U, 3, 4 och 5:
Underkänd: - 15 p
3: 16 - 20 p
4: 21 - 25 p
5: 26 - 30 p

Betygsgränser för U, G och VG
Underkänd: - 15 p
G: 16 - 23 p
VG: 24 - 30 p

Skriv namn, klass och personnummer på samtliga lösa blad. I övrigt så gäller även följande:
1) Endast ett svar per blad. Använd endast framsidan (delfrågor kan vara på samma sida).
2) Sortera inlämnade svar med avseende på uppgiftsnummer i stigande ordning.

Poängavdrag kommer att göras om detta inte åtföljs!


Dina svar skall tydligt visa lösningsmetod. Enbart rätt svar kommer inte att ge poäng. I det fall du är osäker på frågeställning, skriv ner din tolkning och lös uppgiften utifrån din tolkning.

Lycka till!

jörgen
Q1: Memory Management

Suppose we see a dynamic memory management scheme that requires 8 bytes per block in space overhead and requires 50 microseconds to respond to a request for memory. Suppose that the average request size is 256 Kbytes, and you get an average of 30 memory requests a second.

(i) What is the average space and time overhead for dynamic memory management? (2 points)

(ii) Compute the average space and time overhead for dynamic memory management in a system where the average request is 40 bytes and you get an average of 2000 memory requests a second. (2 points)

(iii) The first figures are more typical of the memory management demands on an operating system and the second are more typical of the memory management demands of a program doing a lot of dynamic memory allocation. What does this say about how each one should be designed? (2 points)

(6 points)

Q2: Scheduling

Consider a variant of round-robin called progressive round-robin. In this algorithm, each process has its own quantum. This starts out at 50ms and increases by 50ms each time it goes through the round-robin queue. Hence, long jobs keep getting longer and longer time slices. Discuss the advantages and disadvantages of this variant over ordinary round-robin.

(3 points)

Q3: Swapping

a) Three things take up time in a swapping system: (1) swapping out the old job, (2) swapping in the new job, and (3) running the new job. When the new job is ready to be swapped out again, it becomes the old job, and the cycle starts over again. If we can overlap these activities, we can speed up the system. There are three levels of overlap: (i) no overlap; (ii) you can overlap running a program with either swapping or swapping out (but not both); and (iii) you can overlap all three things.

Suppose we have a computer system where each job is 200 Kbytes long, the CPU can execute 1M instructions per second, the I/O channels can transfer 1 Mbyte per second, and a job runs exactly 300K instructions before being swapped. (K=1024; M=K*K). We only consider the CPU time spent on running the jobs to be useful. The effective computation rate is the ratio of useful computation time to total time. Hence, if the CPU was busy only half of the time, the effective computation rate would be 0.5. (Assume that processes are in the system, i.e. no cost of initialisation)

(i) For each of the three overlap cases above, compute the effective computation rate. (3 points)

(ii) Now assume that you can speed up the CPU or the I/O channels by 100 percent. For each of the three overlap cases, motivate which one would be better to speed up (you can only speed up one, not both) and how that speedup will affect the effective computation rate. If your results seem anomalous, explain why! (6 points)

(9 points)

b) Generalize the formula for computing the effective computation rate such that it includes any cost due to initialisation of the system. Do this generalization for scenario (ii) and (iii), i.e. where there is an overlap of the activities.

(2 points)
Q4: Two-Phase Locking

Consider the following variation of two-phase locking. Each time a process fails to get a lock and has to release all its locks and start over, it increases its age by 1. As soon as a process reaches age 10, then all other processes that are in phase 1 have to release all their locks and wait (they do not start trying to get locks again). Then this process (the one with age 10) starts trying to get locks, but when it cannot get a lock it just waits for it. When this process completes phase 1 and enters phase 2, then all the other processes in phase 1 can start trying to get locks again.

Does this algorithm prevent deadlock? Does it prevent starvation? Explain your reasoning for each answer!
(4 points)

Q5: Paging

a) Suppose you have a two-level paging system where the first level uses 10 bits, the second level uses 10 bits, and the page offset uses 12 bits. If you have a program that uses 18 Mbytes of memory, how many page frames will it use for the program and the page tables if everything is in memory? What is the memory overhead due to paging in this case? (Assume that processes are in the system, i.e. no cost of initialisation)
(3 points)

b) Given the two-level paging system given in (a), suppose that it takes 10 ms to service a page fault and a regular memory access takes 200 nanoseconds. An average instruction takes 500 nanoseconds. The translation lookaside buffer (TLB) hits 90 percent of the time, and the user page fault rate is one page fault per 100,000 instructions.

You are asked to
(i) compute the effective access time given that the TLB is not used;
(ii) compute the average time for memory fetch when TLB is enabled, and assuming that page faults are not occurring (TLB is active); and finally
(iii) compute the average instruction time for all instructions, including the ones that cause page faults.
(3 points)