CSE 121 Winter 2005
Homework 3

Due: Thursday, March 3 2005

Lottery Scheduler

1. One tricky aspect of implementing a scheduler is priority inversion - where one process is waiting on a lock held by another lower-priority process, and its effective priority is therefore limited by the priority of the lower process holding the lock. How would an implementation of lottery scheduler address this problem?

2. This question will run through a basic example of a lottery scheduler. Assume we start with two processes - A and B and a base currency of 1000 tickets. 500 Tickets are assigned to process A, and 500 tickets are assigned to process B. A and B both have their own currency of 100 tickets each (which is backed by 500 tickets of base currency for both processes). Assume A has two threads associated with it: $T1$ and $T2$. $T1$ is assigned 25 tickets of A's currency and $T2$ is assigned 75 tickets of As currency. Process B also has two threads associated with it: $T3$ and $T4$, where $T3$ has 20 of B's tickets and $T4$ is assigned 80 tickets of B's currency. Assume that we are starting a new time quantum τ_i and at the beginning of this time quantum thread $T1$ is current blocked awaiting I/O.

   (a) For time quantum τ_i, for each thread $T_j \in T1, T2, T3, T4$, what is the probability that T_j will be chosen to run?

   (b) Assume that for time quantum τ_i, $T3$ was chosen to run, but $T3$ only ran for half of the quantum. For time quantum τ_{i+1}, for each thread $T_j \in T1, T2, T3, T4$ what is the probability that T_j will be chosen to run? Assume that $T1$ is still blocked.

3. Describe why the lottery scheduler is a well-suited scheduler for the Monte-Carlo algorithm. For contrast, how would one go about mimicking this behavior using a traditional priority-based scheduler?

Virtual Memory

4. When porting Unix to the VAX, why did the designers implement a free page pool rather than have all pages on the system in the loop of the clock algorithm? Give an example of a type of access that would benefit from having this free page pool.

5. The paper mentions that the VAX lacks page reference bits. Why are page reference bits traditionally necessary for the clock algorithm? Briefly explain what mechanism was used to get around this.

GMS

6. In terms of data structures described in the GMS paper, describe the steps taken to locate a page currently stored in Global memory. More specifically, Node A is requesting page 1, and page 1 is not found in its local cache, but node B has a copy of page 1 in its global cache. Also, assume that node C stores the GCD entry for page 1. Be sure to say which node's data structure is accessed and not simply name data structures, and explicitly state any communication between nodes.

7. Assume that we are designing a GMS system but want to make some optimizations. For efficiency we decide not to calculate the $MinAge$ parameter at every Epoch. Instead we will choose a constant offset from the current time as our $MinAge$ (i.e. $currTime + x$ for some predetermined value of x) and use this for every Epoch. As a result, every node can determine $MinAge$ at the beginning of each Epoch in constant time and without communicating with any of the other nodes in the cluster. Do you think this is a good idea? Explain why or why not.

μ-Kernels

8. This question will highlight the advantages of a tagged TLB for address space switches.
   (a) Assuming an architecture with tagged TLBs, describe why the following process context switches would not incur significant overhead: Process A is a 'large' process with a large working set, process B has a small working set. Process A is running then context switches to process B. After some amount of time, process A gets context switched back in and runs.
   (b) Now, describe a situation where a context switch would incur a large overhead.

9. The L4 μ-kernel paper attempts to address the problem of address-space context switches being a costly factor in a μ-kernel architecture. Assuming an architecture without tagged TLBs, describe an approach to combat the problem of large address-space context switches that a Unix-like environment would benefit from.

10. In the Mach kernel, what are the requirements for a user-level memory manager (i.e. what does it need to do and how would one go about writing one) and how would a user-level memory manager interact with the kernel. What are the limitations of a user-level memory manager? What is the action taken if a user-level memory manager fails to page-out when requested?