Homework #5 (100 Points)

Due 03:00pm on Thursday 03/17/2015 through gsubmit.

Feel free to make assumptions, if you feel that such assumptions are justified or necessary. Please state your assumptions clearly. Unreasonable assumptions lead to unreasonable grades!

---

1. Consider an F-SCAN CPU scheduler, whereby the CPU uses a Shortest-Job-First algorithm on a high-priority buffer, while new arrivals are queued in a low-priority buffer. When all requests in the high-priority buffer are finished, the buffer priorities are switched. Consider a Poisson arrival process with rate 100 requests per second and a uniform service time ranging between 1 milliseconds and 20 milliseconds. Compute the expected value of the maximum number of out-of-order executions of jobs in this system. Said differently, for a request R arriving at time *t* to this system, what is the expected value of the number of requests arriving after *t* which could be served before R in the worst case?

2. In this problem you will use simulation to compare the performance of the following schedulers with respect to CPU-bound versus I/O-bound jobs:
   - FCFS
   - Shortest Remaining Time Next
   - Round Robin with a quantum of 100 msec
   - Highest Slowdown Next applied preemptively using a timeout of 100msec

   The following are parameters/assumptions you should use in your simulation:

   - The arrival process for I/O bound jobs is Poisson with rate = 6 jobs per second
   - The service time for I/O bound jobs is exponential with mean = 10 msec
   - The arrival process for CPU-bound jobs is Poisson with rate = 3 jobs per second
   - The service time for CPU bound jobs is exponential with mean = 300 msec
   - Total simulation time = 100 seconds
   - You can assume that the exact service time for a job is known a priori (i.e., the scheduler does not have to estimate it)

   For each of the schedulers, compute the average slowdown for I/O-bound jobs to that of CPU-bound jobs. How do these schedulers rank from the perspective of I/O-bound jobs?

   What to hand in: Source code that we can compile and your results/answers.

3. A web server has a caching subsystem that holds a set of files out of all the files in the system. Each file in the file system has a unique ID (e.g., filename). Requests served from the cache take much less time than those served from the disk. Thus, in an attempt to speed up its

operation, the web server uses the following scheduler for HTTP requests:

i. When a request for a file *fi* is received, if *fi* is in the cache, then the request is put in Q1, otherwise it is put in Q2.
ii. When the server is available to process a request: If Q1 is not empty, then the server picks the next request from Q1, otherwise the server picks a request *fj* from Q2, where *fj* is the smallest ID of all requests larger than the maximum ID of any file in the cache.
iii. When a file *fj* is retrieved from disk (in response to a request de-queued from Q2), then *fj* is cached by replacing the file *fk* with the minimum ID amongst all files in the cache.

Answer the following questions:

a. Show that starvation is possible in the above system. Specifically, describe a particular scenario under which starvation is possible.

   To minimize the impact of starvation, a friend of yours suggested that the following initial step be incorporated at the beginning of the above solution:

   > *Requests to the web server are batched up in groups of up to N requests. A batch is processed by going through steps (i), (ii) and (iii) for all requests in the batch. Once all requests in a batch are processed, a new batch is considered, etc.*

b. What kind of disk scheduling algorithm was your friend thinking of when she suggested the above fix?
c. Discuss the effect of N on the effectiveness of the cache (i.e., is a large N better for improving the cache performance or is a small N better?)
d. Discuss the effect of N on the fairness of the above system (i.e., is a large N better for improving fairness or is a small N better for fairness?)
e. The value of N can be set to 1, 10, or 100, depending on the load on the server. Which value would you pick under each of the conditions below (briefly explain your choice):
   - The load on the system is light and a small number of files are highly popular compared to the rest (i.e., there is strong locality of reference).
   - The load on the system is moderate and a small number of files are highly popular compared to the rest (i.e., there is strong locality of reference).
   - The load on the system is high and all files in the system are equally likely to be requested (i.e., there is no locality of reference to speak of).
   - The load on the system is extremely high and a small number of files are highly popular compared to the rest (i.e., there is strong locality of reference).

4. There are 3 classes of jobs in a computer system.

   Jobs of class A arrive to the system very infrequently, but are of the highest possible priority. Jobs of class C arrive to the system very infrequently and have the lowest possible priority. Jobs of class B arrive to the system quite frequently and have medium priority. Assume that all three

classes of processes need the CPU. To manage the CPU, priority scheduling is used, whereby the highest priority Job that needs the CPU is scheduled (preempting any other job of lower priority that may be using the CPU, if necessary). Ties are broken arbitrarily. Answer the following questions:

a. Is starvation possible? Which of the three classes of jobs is most susceptible to starvation?

Assume that jobs of class A arrive periodically every minute (i.e. exactly every minute a job of class A arrives) and that each such job needs the CPU for exactly 5 seconds. Also, assume that jobs of class B arrive periodically every second and that each such job needs the CPU for exactly 0.5 seconds. Finally, assume that jobs of class C arrive periodically every 5 minutes.

b. Assuming that jobs of class C need the CPU for exactly 20 seconds. Answer the following questions:
   - What is the worst-case response time for jobs of class A?
   - What is the worst-case response time for jobs of class B?
   - What is the worst-case response time for jobs of class C?

c. What is the maximum amount of CPU time per period for jobs of class C beyond which the system will never reach steady state? In other words, what is the maximum amount of CPU time per period of jobs of class C that would make the worst-case response time for some jobs in the system grow infinitely large over time...

As it turns out, jobs from class A and class C need another resource R. R is a resource which cannot be shared. In other words, if a job starts using resource R, then R cannot be released to another job until that process is done---think of a printer, for example. Thus, if a job needs resource R and that resource is ``busy'' then that job must block waiting for the resource. When the resource R is released, it is given to the highest-priority job waiting for it with ties broken arbitrarily.

Assume that a job from classes A or C needs the resource R concurrently with the CPU (i.e. R will be ``busy'' as long as the job is not done with the CPU). Notice that if a job cannot get a hold of R, then it cannot compete for the CPU (it has no use for the CPU without having R).

d. Explain why the priority scheme devised for managing the CPU is not adequate when sharing resources such as R. In particular, show that under the above conditions, it is possible for a job from class A (which is presumably of the highest-priority) to be waiting for a job from class C (which is presumably of the lowest-priority) to release the shared resource R.
e. What is the worst-case response time for jobs of class A?

Read the article on *"What Really Happened on Mars?"* and answer the following:

f. Define what is meant by priority inversion.
g. Define what is meant by priority inheritance.
h. Assuming that priority inheritance is used to schedule the CPU for jobs of classes A, B, and C. What is the worst-case response time for jobs of class A?
