

Project #3 Simulation of an Interactive Computer System with Memory Constraints

Due: Wednesday, Apr. 19, 2006

This project investigates a variant on the model of the interactive, multiprogrammed computer system that was the subject of Projects 1 and 2. The new model incorporates explicit constraints imposed by memory demands.

In a simplified model of an interactive, multiprogrammed computer, jobs can be in one of three states:

- thinking - at a terminal
- ready - no longer thinking but without a memory allocation
- active - not thinking and having a memory allocation

An active job can be blocked waiting on CPU service, receiving CPU service, blocked waiting on disk access, or accessing a disk. It holds its memory allocation as long as it is active, surrendering the allocation only when it returns to the thinking state.

If a job enters the ready state and the system has enough free memory to give the job its allocation, the job immediately enters the active state. However, a ready job may require more memory than is currently available, and if so, it must wait until enough memory is released by jobs returning to the thinking state to allow it to receive its full allocation. A job is given its full allocation or none; it cannot receive a partial allocation and continue to wait until additional memory becomes available.

Three scenarios:

- (a) Each job requires a memory allocation of the same size. If one or more jobs are ready, the job that has been waiting the longest receives its allocation first (strict FCFS allocation).
- (b) Jobs require different size allocations. Ready jobs still receive allocations in FCFS order, even if a job that has been ready a shorter time could have had its demand for memory satisfied earlier.
- (c) If a job releases its memory allocation so that the total free memory is X , and only one ready job needs X or less, that job is given its allocation, regardless of how long it has been ready. If two or more ready jobs are each waiting for an allocation of no greater than X , the job that has been waiting longest for its allocation is given memory first.

Compare the performance of the system under all three scenarios, with 20 terminals and the amount of memory available ranging from 16 Mbytes to 160 Mbytes, in increments of 16 Mbytes. In (a), each job requires 8 Mbytes. In (b) and (c), half of the jobs require 4 Mbytes and half require 12 Mbytes (a job requires the same allocation for the entire simulation). Performance should be

compared in four ways: throughput, response time, average number of ready jobs, and standard deviation of response time.

Turn in a project report with a cover page that states, on separate lines,

ELEC 428
Spring Semester 2006
Project 3

(date submitted)

(your name)

You decide what goes into the report. You've done two project reports already, and you should use what you've learned from these in writing a report that is complete, well organized, concise, and informative. Use graphs and/or tables where they help the presentation. You *do not* have to show how you determined the end of the transient phase in your simulations or how long it is. Compute appropriate confidence intervals on the response time, but not on the other performance metrics.

Describe how you implemented the memory allocation policy in your simulation, in enough detail that someone reading your description could implement it in the same way. Do *not* simply print the relevant part(s) of your simulation program, although you may want to provide code in an appendix if you think it helps.

The project will be graded as follows:

Correctness:	50%
Report:	
Organization and completeness:	20%
Evaluation/explanation	10%
Quality*:	20%

Correctness is judged based on the accuracy (or lack thereof) of the results you obtain for throughputs, response times, average number of ready jobs, standard deviations of response time, and confidence interval widths for response times, plus any other performance metrics you choose to present.

Organization and completeness of the report: Did you follow the directions regarding the project? Is the report well-organized, clearly delineated, and complete? Have you presented the relevant data? Did you use tables and/or graphs when appropriate? If so, are they done well? Did you include a description of the memory allocation implementation? Is it clear? *Is it efficient?*

Evaluation/explanation: Does the text of the report provide all of the requested information? Is the explanation of the results reasonable? Did you adequately explain seemingly anomalous results? Did you fail to note suspicious or totally outrageous results? Basically, does the report demonstrate that you reflected on the results and came to justifiable conclusions about how to interpret them?

Quality relates to a number of objective and subjective aspects of the report: grammar, spelling, presentation (formatting), clarity, conciseness, and appropriateness for the intended audience. The audience you should target for this report is someone who knows what an interactive, multiprogrammed computer system is, understands the memory constraints, and is reasonably accomplished with YACSIM. That is, your audience is someone like you. Think of how you would explain the model and your results to one of your classmates if she or he were not doing the same project.

All reports should be printed and spell-checked. Any tables and plots should be computer-generated, although you may annotate or elaborate plots by hand.

You may compare your results for this project with those obtained by other students in the class, and you may help one another with any programming or YACSIM-related problems. You may not copy or use in any way another student's program(s), results (except for comparison purposes as explained earlier), or any part of his/her report. With this understanding,

SIGN THE PLEDGE