Introducing Empirical Investigation in Undergraduate Operating Systems

Steven Robbins
Division of Computer Science
University of Texas at San Antonio
San Antonio, TX 78249-0667
210-458-5544
srobbins@utsa.edu

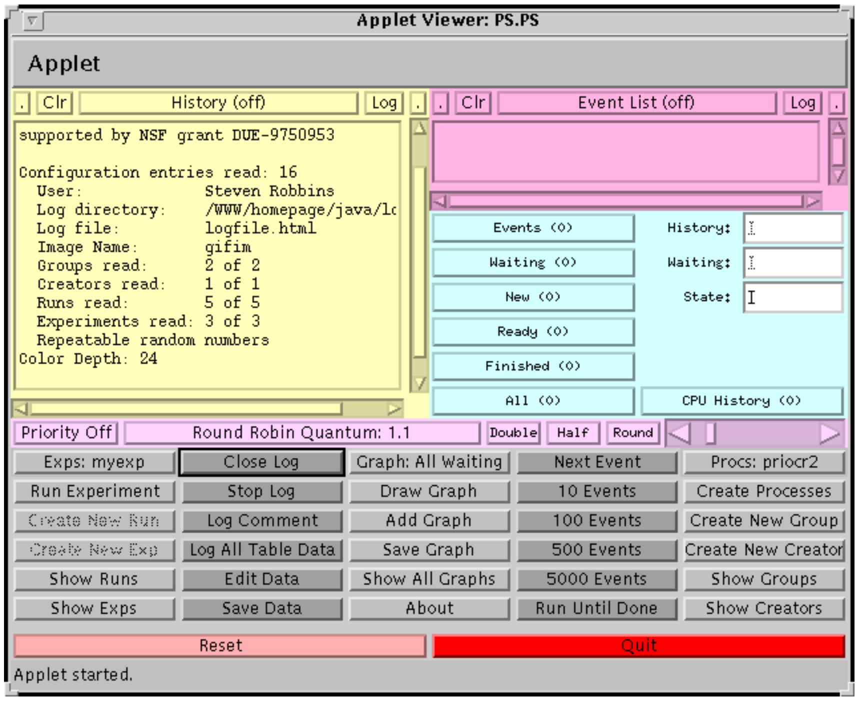
Abstract: The undergraduate operating systems course can provide students with a valuable introduction to empirical testing and experimentation. This paper announces the availability of a process scheduling simulator designed to develop student empirical skills while they are learning part of the standard operating systems curriculum. The simulator is written in Java and available for direct experimentation via the World Wide Web. By accessing the remote URL through an appletviewer, students can permanently save input test data and simulator results generated in HTML format. Students are asked to test specified hypotheses about process scheduling and to develop their own hypotheses about the influence of different parameters on behavior. In order to devise experiments that make sense, students must understand how factors such as burst time and variability influence behavior. Interested faculty are invited to try the simulator and support materials in their operating systems classes.

Keywords: operating systems, process scheduling, education, undergraduate curriculum

1 Introduction

Experimentation is the centerpiece of the traditional scientific method. Experimental exploration can provide new insights, eliminate unproductive approaches and validate theories and methods [1]. Computer science, as a discipline, has a notoriously poor record in the area of empirical validation. Several studies of computer science publications [2, 3] have shown that the percentage of papers providing no substantiation for claims that needed experimental verification was much higher than in other disciplines in either the hard or soft sciences.

This paper announces the availability of a web-based simulator and supporting curriculum mate-



```
supported by NSF grant DUE-9750953

Configuration entries read: 16
  User:              Steven Robbins
  Log directory:     /WWW/homepage/java/lc
  Log file:          logfile.html
  Image Name:        gifim
  Groups read:       2 of 2
  Creators read:     1 of 1
  Runs read:         5 of 5
  Experiments read:  3 of 3
  Repeatable random numbers
Color Depth: 24
```

```
Applet started.
```

Figure 1: A view of the main simulator window.

rials that have been developed to introduce students to empirical exploration in the undergraduate operating systems course. A primary goal is to help the students develop a better understanding of process scheduling including the working of the algorithms and the system parameters that influence their performance. A second goal is expose students to empirical methods in a realistic computer science setting.

2 Simulator Overview

The simulator interface shown in Figure 1 makes it easy to run experiments on collections of processes with different scheduling parameters and to compare such statistics as throughput and waiting time. Students are introduced to the simulator and then given two types of assignments. In one type of assignment the students are presented with a specific hypothesis about process scheduling and are asked to devise and perform experiments to support or disprove the hypothesis. In the second type of assignment, students are asked to develop and test their own hypotheses about process scheduling. An automatic logging facility outputs tables, graphs and comments in HTML format, so that the students can easily keep track of their experiments and produce web-based reports of results.

3 Use of the Simulator

The simulator can be used for exploration in various ways. For example, the experiment below presents a specific hypothesis that students are asked to support or disprove. Students must understand the concepts of CPU burst and I/O burst as well as the meaning of various scheduling parameters in order to devise experiments that make sense.

Experiment:

Hypothesis: *Round robin (RR) scheduling with n processes makes each user think the machine is running at 1/n-th the speed as long as the I/O times are small.*

1. Devise tests to support or disprove the hypothesis.
2. Conduct a series of experiments to determine the effect of perceived performance as a function of I/O burst time.

The process scheduling simulator was introduced in an undergraduate operating systems course in the Spring of 1998 after two 50-minute lectures on process scheduling. The simulator was demonstrated during the third class period using a laptop and video projection unit. Most of the students had previously taken two semesters of calculus-based probability and statistics. There were a few students in the class with graduate training in a technical field other than computer science, and these students showed considerably more sophistication in their designs than the best computer science undergraduates.

The reports were graded, and a class critique was posted on the web. The results were also discussed in class. Students indicated that the discussion and critique were useful to them. They also felt that it would be helpful to get feedback on one design before they had to submit the full report. The consensus of the class was that the experience was useful. In addition to gaining experimental experience, students had a much clearer understanding of the mechanics of time sharing and the implications of the CPU burst and I/O burst times on performance.

4 Simulator Availability

I am currently seeking feedback and participation of other faculty who are interested in incorporating empirical methods into the computer science curriculum at the undergraduate level. If you have any comments or are willing to test the instructional materials, please contact me at `srobbins@utsa.edu`.

One caveat about the current simulator should be mentioned. The current Java security model poses a problem for remote operation of the simulator. Although the simulator can be run through a web browser by accessing the Java source and configuration from a remote site, remote operation in that way does not allow access to the local file system for reading configuration files or storing the log file. One solution is to download the entire simulator package and run it locally through an appletviewer. While this method will work, it requires some expertise and motivation to install the simulator locally. Local installation also does not allow for easy distribution of updates and bug fixes.

A flexible security mechanism has been introduced in Java 1.2 that should eventually solve this problem when it is incorporated into common browsers. The Java 1.2 security model allows the user to specify directories that the remotely loaded applet is allowed to access. Instructions for running the simulator remotely through a Java 1.2 appletviewer and the status on running remotely through the standard browsers are available on the simulator web page [4].

The process scheduling applet described here is part of a larger project supported by NSF to incorporate an experimental approach into undergraduate operating systems and networks courses. The web site for the project is: `http://vip.cs.utsa.edu/nsf/`. The simulator applet and others are available at this site for general use. A more detailed paper and supporting instructional materials are also available at this site.

5 Acknowledgements

This work has been supported by NSF grants: *An Electronic Laboratory for Operating Systems and Computer Networks*, DUE-9750953 and *A Web-Based Electronic Laboratory for Operating Systems and Computer Networks*, DUE-9752165.

References

[1] Tichy, W. F., "Should computer scientists experiment more?" *IEEE Computer*, May 1998, pp. 32–40.

[2] Tichy, W. F., et al., "Experimental evaluation in computer science: A quantitative study," *J. Systems and Software*, Jan 1995, pp. 1–18.

[3] Zelkowitz, M. V. and Wallace, D. R., "Experimental models for validating technology," *IEEE Computer*, May 1998, pp. 23–31.

[4] http://vip.cs.utsa.edu/nsf/process_scheduling.html