Chris DeStefano

CIS - 3207 Sec 003

Dr. Tamer Aldwairi

9-17-2019

README.txt

**INTRODUCTION:**

The file included in this folder labeled P1.cpp is a discrete event simulator that simulates a basic scheduler for a computer’s CPU. I’m using the word basic because the simulator operates on the assumption that the CPU’s scheduler works on a FIFO (First in, First out) basis, which is most likely not how an efficient scheduler would function in reality but for the purposes of this simulation FIFO works fine.

**HOW IT WORKS:**

The system starts by generating a random number of Jobs. Maximum amount of jobs generated is equal to the constant on line 15 MAXJOBS and the system generates a random number of jobs between the interval 0 and MAXJOBS + 1. Once the system generates the jobs each job is given an arrival time and some other flags which will be used for future computing. The Arrival time of a job is generated randomly at time of the jobs generation and is meant to simulate how a scheduler would simulate priority of all the jobs it is managing. These jobs are loaded into a priority queue, with the priority decided by the arrival time.

Once the priority que is loaded then the simulation begins. The driver for the simulation works using a while loop that checks to ensure that the priority que still has a job to service, and that the maximum time of the simulation defined the included file Constants.txt as FIN\_TIME has not passed. After this while loop the simulator starts assigning jobs to components. The assignments are done on the condition that the next job will begin servicing at whatever component is the freest (queue size is the smallest). Except for the condition where a job exits the CPU to do disk I/O. In this case the job returns to the priority queue with the job’s need\_io flag set to true and an arrival time equal to the job’s original arrival time plus the job’s service time job\_time. When this case occurs, the job will begin service at the Disk with the shortest queue.

The time that each job spends servicing at the components is determined randomly at each job’s run time. Runtime is represented by the variable job\_time, defined on lines 94, 182, 259 (one for each component) and is uniformly distributed between the variable (COMPONENT\_MAX-COMPONENT\_MIN) + COMPONENT\_MIN. These constant values are in the included file Constants.txt and are named for their respective components, e.x. CPU\_MAX, CPU\_MIN).

Each time a new job is serviced it has a 50% chance to stay in queue for further computing instead of instantly leaving the component. This is meant to make my simulation even closer to a real-world simulation. By making jobs stay in the component for some time instead of leaving instantly. Thus allowing other components to acquire jobs while CPU is servicing its other jobs. When a new job is added to a component then that Job that was currently servicing has a 25% chance to finish this rotation. If it does, then the component immediately begins service on its next job. When the CPU services a job that job has a 20% chance that it will need further IO at a disk, this chance is determined by QUIT\_PROB in the Constants.txt file. If a job passes the stay check and if it’s In the CPU and passes the need disk io check then the job exits the system immediately after its job\_time has been finished.

**PROBLEMS I EXPERIENCED:**

My primary problem I experienced in coding this simulator was not in the design of the simulator but my language of choice. I originally started coding this in C, as it had been the language that I had worked in most recently and it was the most familiar in my mind. After some time of trying to code in C I decided to switch to C++. My choice in switching programming languages was made primarily because of how much more intuitive C++ is compared to C, and secondarily because of my inexperience compiling on a Linux terminal, my thinking being that I wouldn’t feel like I was fighting the language and compiler the whole time.

**TESTING:**

Testing was done at first using a small amount of jobs MAXJOBS = 50. In order to access that each part of the simulator was working. Once I was sure each part of the simulator was working, I ran tests on a varying number of MAXJOBS up to MAXJOBS = 2000 in order to test that the simulator still runs with a varying degree of settings.