Lab 2: Scheduler

Handed out Monday July 22, 2024
Due Sunday August 11, 2024

Please note that the default setup is to run on two CPUs, which will make debugging and understanding the scheduler output harder. In the Makefile, there is a line that says:

CPUS := 2

Change that to 1.

In this assignment, you will change the scheduler from a simple round-robin to a priority scheduler. Add a priority value to each process (lets say taking a range between 1 to 10). The range does not matter, it is just a proof of concept. When scheduling from the ready list you will always schedule the highest priority thread/process first.

Add a system call to change the priority of a process. A process can change its priority at any time. If the priority becomes lower than any process on the ready list, you must switch to that process.

To get started, look at the file proc.c Implement three of the next five items. If you implement more, each is an 8% bonus. Note that lab bonuses only count towards lab scores.

1. To avoid starvation, implement aging of priority. If a process waits increase its priority. When it runs, decrease it. Note that you have to remember the baseline priority. Otherwise, the design is up to you. (Possible Bonus 1)
2. Implement priority donation/priority inheritence. (Possible Bonus 2). You can demonstrate this by changing priority of a child with lower priority with a parent with higher priority waiting for it to exit.
3. Add fields to track the scheduling performance of each process. These values should allow you to compute the turnaround time and wait time for each process. Add a system call to extract these values or alternatively print them out when the process exits. (Possible Bonus 3)
4. Implement lottery scheduling: Based on priority, you give every process a proportional number of tickets (e.g., priority 1 gets 1 ticket, priority 10, 16 tickets). To schedule, you hold a lottery and pick a random ticket number, scheduling the process that holds that ticket (you have to design a way to associate numbers with tickets so that you can map from a random number you generate to identify the process that holds that ticket). (Possible Bonus 4.)
5. Implement processor affinity. Reenable multiple CPUs, but schedule so that each process keeps getting scheduled to the same CPU to take advantage of its data being in the caches of that CPU. (Possible Bonus 5)

Goals: Understand how the scheduler works. Understand how to implement a scheduling policy and characterize its impact on performance. Understand priority inversion and a possible solution for it.