Ezequiel Valencia, Hunter Krasnicki

Professor Wei Zhang

Operating Systems 4300

November 10, 2022

CPU Scheduling Algorithm Project Proposal

Within any operating system there is a CPU scheduler which queues the next instructions to be executed within the device. An algorithm is utilized to determine the next instructions that will be executed within the operating system, and puts the instructions in queue. There are plenty of simplistic algorithms such as FIFO, which takes instructions and executes them based upon the time of arrival. More complex algorithms take into account turnaround time, throughput, response time, and fairness. The goal for all these algorithms is to allow for the most efficient and seamless execution of instructions, leaving the user unaware that multiple processes are running concurrently, each having their own instructions that must be executed. By analyzing the CPU scheduling process it allows for a greater understanding of algorithmic efficiency, instruction sequencing, and proper management of tasks within a complex system.

Natively, OS161 employs a round-robin scheduling algorithm to handle switching between threads, prioritizing fairness in sharing processors between threads evenly, though at the expense of higher overall turnaround times for tasks, which in many cases can be undesirable. Thus, our proposal is to implement a form of a priority based scheduling algorithm, perhaps either a form of multi-level feedback queue or similar algorithm that seeks to simultaneously promote fairness, whilst minimizing the drawbacks of round robin scheduling in comparatively

high turnaround times. If however this proves to be too ambitious, we had contemplated the idea of creating our own version of a priority based scheduling algorithm in which processes enter in with a default priority that shifts according to certain process behaviors or user preferences. For example, a user may be able to set their own process priority levels, or the OS may dynamically adjust the priority based on a collection of factors such as the rate at which a process blocks, through ‘aging’ where we raise a process’s priority based on how long it’s been waiting in queue, or by its average CPU and I/O burst times. Though the baseline for which we plan to construct a priority based scheduling algorithm is one in which a user can define priority levels, as this seems amongst other aspects to be most readily accomplishable.