Scheduling

Andrew Quinn

Learning Objectives:

1. How do we evaluate scheduling disciplines?
2. How do current scheduling policies behave on these systems?

SCHEDULING

We saw how an operating system uses context switches to regain control of the hardware from a running process. But, how does an operating system decide the process that it should execute? Operating systems follow scheduling algorithms, called *Scheduling Disciplines* in your book, to decide what process to execute.

Before we discuss common scheduling disciplines, let's outline some features that we might want from a scheduling discipline:

1. **Turnaround time**: A job's turn around time is defined as the difference between its completion time and its arrival time.
2. **Response Time**: Many computing tasks are interactive—users actually type on a keyboard or use a mouse to interact with the jobs. Response time attempts to capture the "interactivity" provided by a job. There are many possible definitions, here's the one that we'll use: the maximum duration in which a job is schedulable, but remains unscheduled.
3. **Fairness**: intuitively, a scheduling discipline should try to apportion system time to jobs fairly, so that each job receives an equal share of resources. There are various ways to measure of fairness, such as Jain's Fairness Index, the variance of CPU time that is provided, etc.
4. **Makespan**: Makespan is defined as the total time that it takes for a system to finish executing a group of jobs. Theoreticians often analyze scheduling disciplines using makespan; we won't spend much time on it in this class.

SCHEDULING DISCIPLINES

With that in mind, lets go through a few standard scheduling disciplines. Each of the disciplines will assume different amounts of knowledge about the system's tasks; we will outline the required assumptions as we go.

FIRST-IN, FIRST OUT (FIFO). FIFO, sometimes called First-Come First-Serve (FCFS), processes jobs in their arrival order. FIFO is *non-preemptive*, meaning that it does will not context switch from a task that is not finished. This makes FIFO a poor approach when we have tasks that ever enter a blocked state.

Moreover, FIFO creates an issue called the *Convoy Effect*. This effect happens when a large number of relatively short jobs queue behind a long-running job. The convoy effect can drastically hamper the average turnaround time of the tasks in the system.

Despite these shortcomings, FIFO is, surprisingly, frequently used in some scheduling systems. Usually these use-cases are outside of an operating system, as a "meta-level" scheduler for large clusters of computers. These systems are concerned with minimizing makespan, so FIFO's advantage is that its simple design imposes little overhead.

SHORTEST JOB FIRST (SJF). One partial solution to the Convoy effect is called shortest job first (SJF). SJF requires that the operations system have an estimate for the work to be completed by a task; it then schedules the tasks so that the shortest possible jobs execute first. If all jobs arrive at the same time, then SJF is provably optimal in terms of average turnaround time. However, SJF is non-preemptive, so the convoy effect can still happen if jobs arrive at different times.

SHORTEST TIME-TO COMPLETION FIRST (STCF). STCF takes a similar approach to SJF, except it uses preemption to resolve the convoy effect. STCF assumes an estimate for the remaining execution time of each task. It then schedules the task with the smallest amount of time remaining. This is also called Preemptive Shortest Job First (PSJF). The issue with STCF is that it remains very poor in terms of fairness and response time.

ROUND ROBIN (RR). RR scheduling executes each job for a small period of time (called an epoch or time slice) before switching to the next job. RR can improve response time by using smaller epochs. However, using smaller epochs imposes more overhead on the system, as more time is spend on scheduling. Even when ignoring this overhead, RR will result in average turnaround times that are far larger than optimal.