CMPSCI 577 Operating Systems Design and Implementation **Spring 2020**

Lecture 7: Feb 11, 2020

Lecturer: ***Prashant Shenoy*** *Scribe:* ***Vincent Pun (MS)***

7.1 Process Scheduling

Multiprogramming. The ability to run processes concurrently on a system.

Process execution states. A process starts in the *New* state. It cycles between the *Ready* state, the *Running* state and the *Waiting* state. When execution completes, it goes to the *Completed* state.

7.1.1 Three Levels of Scheduling

Long-term scheduling. Manages how many processes a system can accommodate at any time for concurrent execution. Polices entry of processes into the system. The *Admission Scheduler* is a kind of long-term scheduler.

Memory scheduling. Takes existing running applications and swaps the memory out into the disk if memory is constrained. The *memory scheduler* is a long-term schedulers.

Short-term scheduling. The *CPU scheduler* decides which active processes that are in the ready state get to execute. The short-term scheduler runs when any of these events occurs:

- A process switches from *Running* to *Waiting.*
- An interrupt occurs, because an interrupt handler runs, and the decision of which process to run next is made.
- A process is created or terminated.
- A quantum expires *(on a preemptive system).*

There are two types of short-term scheduling systems:

Non-preemptive system. The scheduler is not allowed to "preempt" a running process. It must wait for the process to complete execution.

Preemptive system. The scheduler can interrupt a running process. This is known as *quantum based scheduling.*

7.2 Process Behavior

A process can be classified as:

CPU-bound process : A process that spends most of its time computing. There are long bursts of computation and short bursts of I/O.

I/O-bound process : A process that spends most of its time executing I/O. There are short bursts of computation and long periods of I/O.

7.3 Scheduling Algorithm Metrics

These metrics for scheduling are defined, which can be used to optimize a scheduling algorithm:

CPU utilization : Percentage of time CPU is busy.

Throughput : Number of processes completing in a unit of time.

Turnaround time : Length of time to run a process from initialization to termination.

Waiting time : Total time a process is in the ready queue.

Response time : Time between when a process is ready to run and its next I/O request.

There are tradeoffs. Generally it is not possible to have a scheduler that optimizes for all of the criteria above.

7.4 Scheduling Policies

We make a few assumptions to make discussions easier: Each user runs only one process, processes are sequential, processes are independent and all processes run on a single-processor/single-core system.

First-Come-First-Served/First-In-First-Out (FCFS/FIFO). This is a nonpreemptive policy. Jobs are executed in the order they arrive. This is a simple policy, but there is high variance in average time. I/O-bound processes may be left waiting while a long CPU-bound process is running.

Round Robin. Jobs are executed in a round-robin order. When a quantum expires, the scheduler moves on to the next process. This is a preemptive policy. The only parameter in this policy is time slice length. A time slice too long causes throughput to suffer due to context switches. A time slice too short causes waiting times to suffer. This is a fair policy, but the average waiting time can be bad.

Shortest Job First (SJF). The job with the least amount of CPU time is first executed. It is the only policy that it optimal in terms of wait times. However, it requires knowledge of the length of the job, which can only be estimated, and is therefore impractical. It can also starve long jobs.

Shortest Remaining Time First (SRTF). The preemptive version of the SJF policy is known as SRTF.

Priority Scheduling. This makes use of round-robin scheduling and multiple queues. It starts with the highest priority queue, and when the queue becomes empty, it moves on to a lower-priority queue. Starvation is still possible—if a higher-priority queue is never emptied, jobs in a lower-priority queue will not execute at all.

High-priority processes include the kernel, system processes and device drivers.

Multilevel Feedback Queues (MLFQ). It is a variant of the priority scheduler with round-robin scheduling on each level. If a process remains in a lower-priority queue for a long time, it is bumped up a priority level.

New processes enter the highest-priority queue when created. A job that uses up an entire time quantum will have its priority **reduced**. A job that is blocked in the middle of execution, then its priority will be **increased**. In other words, CPU-bound processes will likely use up their time slices, whereas I/O-bound processes will not use up their time quanta as often. I/O-bound jobs therefore likely have a high priority while CPU-bound jobs likely have a low priority. Lower priority queues have longer time slices, reducing context switches.

MLFQ effectively approximates SJF without knowledge of the job lengths, as a result of favoring I/O-bound jobs.

Lottery. Every job gets some number of "lottery tickets." Each job has a $\frac{1}{n}$ chance of being selected if each job has one ticket.

Each process can have more than one ticket. This allows the CPU time to be controlled.

Lottery scheduling is a fair policy, where the number of lottery tickets is proportional to the CPU time a process gets.

7.5 Scheduling in MINIX

The scheduler uses the MLFQ policy and maintains 16 queues. Queues are zero-indexed. Queue 0 has the highest priority and queue 15 has the lowest priority. The `system` task, implementing all system calls, and the `clock` task are in queue 0. User processes have a lower priority and are in queue 7 through 14. The `idle` task is in queue 15, and does nothing except running an infinite loop that keeps the CPU busy.

The scheduler maintains the queues as a singly-linked list with two points: `HEAD` and `TAIL`.

Restart Task. The restart task is the way to invoke the CPU scheduler in MINIX.