# CPU governance and scheduling using machine learning implemented on a Linux kernel

Fall 2020 semester report

**Jonathan Gao**

Cornell University
December 13, 2020

# Introduction

As an exploratory dive into the Linux kernel, I plan to modify the Linux scheduler to use learned decisions from a machine learning model. At this project's inception, there were many potential options for where a machine learning model may be practical. As an introduction, I will discuss several of those options, the research done in regards to those paths, and the planned route onwards.

## The Scheduler

The scheduler in an operating system manages the job of "scheduling" processes. A operating system deals with numerous processes, all wanting to perform some task using CPU and memory. However, a modern computer may often only have a few cores or processing units, each only capable of running one process at once. Therefore, the operating system must decide which process gets to run on a processor for how long.

Scheduling decisions can be based on a number of parameters. Some simple scheduler implementations include the First In First Out (FIFO) scheduler, where processes get CPU runtime at a first-come-first-served basis. However, this is scheduler is clearly suboptimal, as we care about parameters such as process latency, process deadlines, and overall process throughput. Thus, we have the notion of a CPU quantum, where a process given a CPU quantum will be given a short CPU runtime, say 20ms, before it is taken off and put back onto the process queue.

CPU quanta introduced to the FIFO scheduler becomes the Round Robin scheduler, where the processes get run in FIFO order, and then get enqueued back onto the queue if it does not complete execution within its quantum. The terminology for the notion of a period of execution for a process has been referred to by several names, but has since become the primarily point of focus for modern CPU schedulers.

Scheduling a process is no trivial task, and extensive work has been done to implement an optimal scheduler. Most modern operating systems implement one of the two following schedulers, the Multi-level Feedback Queue scheduler (MLFQ) or the Completely Fair Scheduler (CFS). For example, Windows and macOS use versions of the MLFQ scheduler, whereas Linux has primarily used the CFS scheduler. I considered both of these schedulers for possible machine learning modifications, both of which will be discussed in the following sections.

### MLFQ schedulers

Multi-level feedback queue schedulers use several queues rather than the single process queue discussed earlier. The "multi-level" queues correspond to different levels of time quanta on the CPU. The highest queue has the shortest quanta on the CPU, with each subsequent queue having longer quanta. With several queues, processes placed onto these queues will run for the time quantum specified by its queue. The number of queues and the range of time quanta will vary, but this structure allows for processes to be classified into groups based on their needs [1]. Using this model, we can prioritize interative and I/O bound processes by introducting priorities, preemption, and feedback across the queues.

# References

[1] "Multilevel queue," *Wikipedia*, Nov. 2018.