Introduction

Operating systems are complex software programs that allow users to safely interface with a computer and perform various tasks. While most operating system are composed of millions of lines of code with thousands of hours of development and thought behind their designs, it is possible to construct a simple software model called a discrete event simulator that abstracts the behavior of an operating system's process management methodology and provides a sandbox for investigating operating systems related concepts (such as CPU scheduling) without worrying about the complexity of the software itself or the specifics of different types of processes.

In the discrete event simulator model, processes with randomly distributed CPU and I/O burst times move between ready, running, and I/O blocked states in the form of discrete events. Discrete events encapsulate "important" (with respect to the OS behavior) moments in a processes' lifetime. Events are chracterized in 6 different flavors (process submitted, process terminated, I/O request, I/O complete, and time slice expired) occurring at various time steps during the course of the simulation. For a given event, its associated process' parameters will be updated accordingly and cause the process to move between ready, running, and I/O blocked states.

This document serves to profile the performance of the classical round robin CPU scheduling algorithm and compare it against an artificial intelligence guided process management scheme that seeks to learn adaptive scheme for time quantum assignment.

System Description

Here we describe the way the software model is designed and provide more detail to the algorithms behind round robin and adaptive preemption schedulign schema.

Software Hierarchy

The discrete event simulator software is an object oriented implementation consisting of several interacting portions:

- **Process** is the "currency" of the system. Processes are instantiated with specific parameters (CPU and I/O burst times, CPU demand, arrival time, etc) and traded throughout the previously mentioned ready, running, and blocked states until either the simulation completes or the process' total demand reaches zero.
- **Event** are points in a Process' lifetime that are important to the OS. Events point to a specific process at a certain timestep and inform the system how to react.
- **CPU** is an object that accumulates its activity (context switch, idle time, active time) throughout the simulation. CPUs also enable the functionality of allocating a processs and executing it for a certain duration, either until the process terminates, reaches an I/O fault, or its time quatum expires.
- **DiscreteEventSimulator** guides the entire system. The largest portion of the DiscreteEventSimulator codebase consists of 6 conditions that, given an event, allow that event to be handled and trigger the creation of a new subsequent event.
- **ProcessFactory** is not part of the model whatsoever, but rather provides an elegant solution to on-the-fly process creation.

Round robin scheduling

Adaptive process control

Experimental Design

Experiments were running by testing round robin

Results

Discussion

Future Work