DEFAULT SCHEDULER

by

CAN BERK DURAK

CSE 331 Operating Systems Design

Term Project Report

Yeditepe University

Faculty of Engineering

Department of Computer Engineering

Fall 2021

ABSTRACT

TABLE OF CONTENTS

1. INTRODUCTION

Process scheduling is an important part of multiprogramming operating systems. It is the process of removing the running task from the processor and selecting another task for processing. It schedules a process into different states like ready, waiting, and running.

The goal of the project is to compare and analyze the behavior of the standard scheduler and a Modified scheduler. The simulation will mimic the behavior of the schedulers in a multiprogramming system and will demonstrate the differences in CPU allocation and process management.

The standard scheduler will be implemented using one of the commonly used algorithms, such as Round Robin, Priority Scheduling, Completely Fair Scheduler (CFS), or Multilevel Queue Scheduling. The behavior of the standard scheduler will be observed and recorded.

The Modified scheduler will be implemented as an alternative to the standard scheduler. It will use a different approach to process management and CPU allocation, and its behavior will be compared to the standard scheduler.

2. DESIGN and IMPLEMENTATION

This part of report explains that how default scheduler works and how did I modify the default scheduler, differences between them and what are methods I used.

2.1. DEFAULT SCHEDULER

The priority-based scheduler in the Linux kernel's version 2.4 has a dynamic component to its computation of priorities. Processes are given a priority by the scheduler, which also adjusts them as necessary to maintain system performance. The objective of a scheduler, excluding the

The goal of real-time activities is to keep CPU-bound processes active as much as is practical while swiftly responding to I/O-bound processes. The Linux scheduler emphasizes I/O-bound operations while degrading the priority of CPU-bound processes in order to achieve this goal. The dynamic element of the priority-based scheduler is this activity.

Processes with time measured in clock ticks are offered by the Programmable Interval Timer. The processes are then permitted to continue running until their remaining CPU time has been used up. A fresh allocation is then made to each process as the cycle is restarted. A full cycle of CPU time allocation and process execution is known as an epoch. A process's static priority and how much CPU time it used in the previous epoch influence how much time is granted to it during the current epoch.

2.2. DESCRIPTION AND IMPLEMENTATION OF MODIFIED SCHEDULER

The default time sharing policy is SCHED_OTHER. The amount of time a task can use the CPU continuously if other tasks are waiting to use it is capped by the traditional time-slicing mechanism.

I should change the default Linux scheduler so that, in the presence of any other processes that are now executing, processes belonging to a specific user are viewed as low-importance and assigned the lowest feasible CPU use.

Linux default scheduler basically, sorting the processes according to them nice values, and gives priority to the best one. On the other hand our modified linux scheduler checks the processed uid and according to this uid values scheduler understand which is process is which user's then sets a different counter value.

For implementing this algorithm, we just need to make a small difference in 'repeat_scheduler' which is defined in 'kernel/sched.c', by adding new if statement in it we can handle it.

REFERENCES

The Linux Process Manager: The internals of scheduling, interrupts and signals

https://www.usenix.org/legacy/publications/library/proceedings/usenix01/freenix01/full_papers/alicherry/alicherry_html/node5.html

https://docs.kernel.org/scheduler/index.html