Homework 2

Homework 2 Writeup CS444 Spring2018 Group 37

Brandon Mei, Brian Huang

1 IMPLEMENT DESIGN

To implement the SSTF algorithm, we need to began with the current NOOP implementation but the SSTF have limitation of starvation. By implementation of LOOK it is important to prevent the issue of starvation. The LOOK algorithms is basically at the starting point, the next closest requested service. This service is moving through the disk in the elevator operation where it goes down one direction and request in that direction until there is no more request. After that the algorithms goes to the opposite directions and repeat the process.

2 CONTROL LOG

Detail	Author	Description
aaef9e7	Brian Huang	Added sstf-iosched.c
bf08926	Brandon Huang	working scheduler and working with the VM/Kernel
fa6bf31	Brandon Mei	Fixed bugs and missing semicolons

3 WORK LOG

Our group began working on the HW2 about three days ago. We did some changes to the kernel/WM to change to the default scheduler. When changing the files directory the issue was resolve and started working on the actual scheduler. We started to utilizing the existing Noop scheduler and test some of the core function. After testing different literation of the existing function and building new components process.

4 QUESTIONS

4.1 What do you think the main point of this assignment is?

We thought that the main point of this assignment was to build a good understanding of Linux in I/O schedulers and other algorithms we learned. Another thing of this assignment is to build a certain skills in modifying the lower levels of the Linux kernel.

4.2 How did you personally approach the problem Design decisions, algorithm, etc.

At the beginning, We thought this assignment made us nervous and confused on what exactly needed to do in order to have the requirements of the assignment. We began searching through the Internet about the information of elevator algorithm that is used and various variants and their differences. Later went thought the kernel's schedulers and use noop as a reference to build the LOOK based algorithm. Our implementation was mainly based on a existing noop algorithm with some modifications in dispatching and adding new request.

4.3 How did you ensure your solution was correct Testing details, for instance?

To make sure that the solution was correct, we went through couple attempts of rebuilding the VM/Kernel with the schedule implementation. Then able to develop an implement to build and print statements in the blocks to output what its happening. throughout the scheduler. When getting good enough information, we concluded that the scheduler had met the requirements and our solution was correct.

4.4 What did you learn?

The Group have learned enough of the low level kernel while creating and setting the default schedulers. This show us many information of the evelator algorithms and scheduling was also useful on this assignment. The kernel data structures and the Disk I/O was the important part of developing to understand it.

4.5 How should the TA evaluate your work? Provide detailed steps to prove correctness.

We have a couple of print statements in the add request function. To check for correctness the TA should look if the print statements are printing when they are supposed to.

REFERENCES

[1] J. Debian, "qemu-doc - qemu emulator user documentation," https://manpages.debian.org/jessie/qemu-system-x86/qemu-system-x86_64.1.en.htm May 2016, (Accessed on 4/15/2018).

[2] R. Love, "Linux kernel development," Pearson Education, 2010, Accessed on 5/6/2018.