CS444 Homework 2

Lilliana Watts, Nicholas Skinner, Yong Ping Li

May 5, 2018

Design Plan

In order to implement the LOOK I/O scheduler, we had planned to use a C-Look algo. We had designed our scheduler to examine requests, and then order them such that they are placed in an order that operated similar to the C-LOOK algo. This means that we will have an intiger that represents the sector position of the last request dispatched, called head, as a member of the sstf data struct. This head value is compared against the block position of the newest request, such that requests that possess a sector position after the current head will be placed in an ordered fashion within the front segment of the queue, and requests that have a block position behind the head are placed at the back segment of the queue. this means that requests will be dispatched in one direction, then the head will choose to return to the next earliest sector position, and do another sweep of all the requests that are positioned after that.

Writeup Questions

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

• The main point of this assignment was to guide us into investigating, and learning about I/O schedulers, the Linux block layer, as well as how elevator algorithms operate. This assignment is also intended to help familiarize us with modifying some of the given QEMU command options.

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

• We had first examined the NO-OP I/O scheduler filer, and then researched on how look, and C-LOOK algorithms operated. We decided to implement a C-look algorithm, as it seemed to be the simpler of the two to understand and to implement. After learning how Linux list heads were implemented, as well as examining how the elevator.c functions operated, we had decided to implement the C-LOOK scheduler by examining the sector position of each request currently in the queue, and the ncomparing it against the sector position of the request to be added to the queue. We had implemented some logic to place trquests ahead of the current head's sector position in a sorted order in front of our current head, and requests

that are behind the current head's sector position after those requests are in sorted order. This had caused requests to be dispatched while they were travelling in one direction, and then jumped back to the earliest sector poisition to travel in the same direction again. This Logic and implementation modelled a C-LOOK approach to the I/O scheduling.

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

• We used printk statements inside of the dispatch and add request functions to report what sector positions they were reading or writing to. We had examined the output to prove that it was following a C-LOOK model.

D. What did you learn?

• We learned about how the block layer, elevator algorithms, and the I/O schedulers operate to organize and carry out write requests on a drive. We also learned how we can disable the virtual I/O used on out previous QEMU command line.

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

• First, run makemenuconfig, and then select the sstf scheduler

The TA should use our patchfile to patch the kernel, and patch the block. if a hunk failure occurs, please delete the kconfig that is within the block structure. Then, they should boot with the following command line (assuming the files folder contains the core-image...ext4 and the current directory contains the linux-yocto-3.19 file)

• qemu-system-i386 -gdb tcp::5507 -nographic -S -kernel linus-yocto-3.19/arch/i386/boot/bzImage -drive files/core-image-lsb-sdk-qemux86.ext4 -enable-kvm -net none -usb -localtime –no-reboot –append "root=/dev/hda rwconsole=ttyS0 debug"

Now, navigate to /var/log/ in the kernel, and cat the 'messages' file. This should show printf statements that were printed by the scheduler, indicating adds and dispatches, and the sectors they occured in. You will also notice that when queued, dispatches alwasy follow the elevator-algorithm of the C-Look.

GITHUB LOG

Detail	**Author**	**Description**
d33bba9	Nicholas	Start Writeup
2f128bf	Nicholas	Submit sstf
1b0d943	Nicholas	Finish Writeup
4cf94f3	Nicholas	Submit Patch

WORK LOG

Date	Name	Hours	Description
4/30	Nicholas	1	Setup Document
5/1	Nicholas	1	Work on writeup
5/2	Nicholas	2	Research C-look
5/3	Nicholas	3	Begin coding
5/4	Nicholas	3	Finish coding
5/4	Nicholas	2	Testing
5/2	Nicholas	2	Finish Writeup