CS 444 Project 2: I/O Elevators

Will Sims, Aidan O'Malley

CS 444

Fall 2017

Abstract

This document contains our write-up for Project 2 of Operating Systems II. Included is the design for our CLOOK algorithm, a version control log, work log and responses to the main write-up questions.

I. Design for CLOOK Algorithm

Our plan of action for this assignment was mainly to change the add_request function of the FIFO scheduler. Most of the other functionality could be kept the same as in the FIFO scheduler implementation if the CLOOK request queue is set up correctly. We decided first that wed need to set a diskhead variable when we first dispatch a request to know where the diskhead is currently, the diskhead location is a main difference between CLOOK and FIFO since CLOOK is an elevator algorithm. We then use that diskhead value to figure out where to insert new requests into the CLOOK request queue. In the add_request function, by using the diskhead variable and by comparing the sector location of the new request with the sector locations of the requests already in the CLOOK queue, we can make a queue that is circular. The scheduler will dispatch correctly with regards to the CLOOK algorithm even though we use FIFO functions for the rest of the implementation (for the most part).

II. Version Control Log

Table from repo logs:

Commit Link	Author	Commit Message
3f942907cd1aef64215014b8296081002eeedc8e	William Sims	Added to latex and created IO python script
52703a0cbda4da1a56f5d784e7dc5acad19a40b0	omalleya	reinitialize
fa180fbb15299647724e7adb8d56295e86466c5c	omalleya	adds changes to sstf-iosched add request function to start implementing c look algorithm
476bce90488ee059258115d312b2e7c2887a81eb	William Sims	implements random number generation Updated Kconfig and Makefile to use CLOOK implementation
e14dd5e9392a891ac909ae5f4cf21bee8ca1a6c9	omalleya	finishes implementing clook, all changes were made in dispatch and add functions
2ed03bbcc05d3b87d95d1c10ea9d60a1e0113794	William Sims	Added description to CLOOK scheduler

III. Work Log

What work was done and when:

10/23/17 Met at Allan Bros for 2 hours and created skeleton for the dining philosophers problem with paired programming.
10/25/17 Aidan worked remotely and finished the dining philosophers concurrecy.
10/26/17 Will wrote a test suite for the dining philosophers problem.
10/27/17 Aidan fixed bugs in test suite.
10/27/17 Will added readme and submitted concurrency 2.

10/29/17 Planned out kernel assignment and researched scheduling algorithms for 3 hours.

10/29/17 Used paired programming and began implementing the CLOOK algorithm.

10/29/17 Created Latex file for writeup and began answering early questions.

10/30/17 Met at Johnson Hall to finish CLOOK implementation, updated Makefile and Kconfig.ioshed file.

IV. Questions

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

The main point of this assignment is learn about different scheduling algorithms and how to implement them in the Linux kernel. The current FIFO (also known as no-op) implementation takes all incoming requests and places them at the end of the queue. The requests are served in the order that they were recieved. However, there are more effective scheduling algorithms such as elevator algorithms which services requests in one direction until it reaches the end of the disk. After the end is reached, the direction is reversed and the same process is repeated. For this assignment, we implemented the C-LOOK algorithm which sweeps from either inside or outside and when the edge of the disk is reached, the head jumps to the other end and services requests in the same direction.

How did you personally approach the problem?

We approched this problem by first researching different scheduling algorithms such as LOOK and CLOOK implementations[1]. We then thought about how these algorithms differed from the FIFO scheduler. After considering the similarities and differences between the two schedulers, we made our plan of action for CLOOK (see the design section earlier). We then implemented this plan.

How did you ensure your solution was correct?

We ensured our solution was correct by using print statements in our scheduler code and running ”dd if=/dev/urandom of=./clook_test bs=1024 count=8192” which writes random data to the disk. We also did other simple IO operations such as making directories and test files. After observing the printed queue, the requests were in sorted order and matched expected CLOOK output which made us confident that our solution is correct.

What did you learn?

We learned how FIFO, CLOOK, and LOOK schedulers are implemented. We also learned how to tell qemu what scheduler to use, how to create and select a scheduler, and how to create a linux patch.

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

```
[Step: 1] START FROM CLEAN linux-yocto v3.19.2
[Step: 2] git apply assign2.patch
[Step: 3] source /scratch/opt/poky/1.8/environment-setup-i586-poky-linux
[Step: 4] cp /scratch/files/core-image-lsb-sdk-qemux86.ext4 *top of linux tree*
   //copy driver file if not inside already
```

```
[Step: 5] cp /scratch/files/config-3.19.2-yocto-standard *top of linux tree*
  //copy config if not inside already
[Step: 6] make menuconfig
[Step: 7] General Setup
[Step: 8] Change local version name to -group-19
[Step: 9] Save and Exit
[Step: 10] make -j4 all
  //build kernel
  //select clook scheduler here or after starting VM run below command in VM
[Step: 11] echo clook > /sys/block/hda/queue/scheduler
[Step: 12] command to start VM:
  qemu-system-i386 -gdb tcp::5519 -S -nographic -kernel arch/x86/boot/bzImage -drive
  file=core-image-lsb-sdk-qemux86.ext4 -enable-kvm -net none -usb -localtime --no-reboot
  --append "root=/dev/hda rw console=ttyS0 debug"
[Step: 13] $GDB
  //From another terminal
[Step: 14] target remote :5519
[Step: 15] continue
[Step: 16] Login to VM as root
[Step: 17] dd if=/dev/urandom of=./clook_test bs=1024 count=8192
  //This will write random data to the disk and you should see the correct queue from
  the CLOOK output.
[Step: 18] run command dmesg if logging to console is not sufficient to view the CLOOK
  operations
```

References

[1] I. I. of Technology, “Debian manpages for qemu-system-i386,” http://www.cs.iit.edu/~cs561/cs450/disksched/disksched.html.