Homework

William Maxwell, Jing Huang, Deval Prashant

May 6, 2018

Abstract

For this project we studied I/O scheduling in the Linux kernel. We implemented our own I/O scheduler, compiled and build a kernel image, and tested our scheduling algorithm.

1 Design Plan

First, we create a new file `block/sstf-iosched.c`. We copy the contents of `block/noop-iosched.c` into our new file to use as a template. We do a search and replace to change all instances of *noop* to *sstf* using the command `%s/noop/sstf/g` in `vim`.

Next, we modify the contents of `block/Makefile` so our new scheduler will be compiled. We add the line `obj-$(CONFIG_IOSCHED_SSTF) += sstf-iosched.o`.

We add a configuration option for the new scheduler in the `block/Kconfig.iosched` file. The line `config IOSCHED_SSTF` to create the option, and the line `default "sstf" if DEFAULT_SSTF` is added to make our scheduler a default option.

Now, running `make menuconfig` under the block layer options you will see the new SSTF scheduler. Select it as the default scheduling algorithm and run `make -j4 all`. We also change the `LOCALVERSION` variable to append the string *Group12* to the kernel name. Then we can use `uname -a` to be sure that we've booted into the proper kernel. We can now run QEMU, but we need to make sure it is using the new kernel. We simply change the path for the `-kernel` flag to `linux-yocto-3.19/arch/i386/boot/bzImage`.

```
qemu-system-i386 -gdb tcp::5604 -S -nographic -kernel linux-yocto-3.19/arch/i386/boot/bzImage
-drive file=core-image-lsb-sdk-qemux86.ext4,if=virtio -enable-kvm -net none
-usb -localtime --no-reboot --append "root=/dev/vda rw console=ttyS0 debug"
```

To verify that the scheduler is being used enter the command `cat /sys/block/hdc/queue/scheduler` to see the list of available schedulers. The scheduler inside square brackets is the one being used. Finally, we create a patch file for the `block` directory with the command `git diff block/ > group12.patch`.

2 Work Log

Detail	Author	Description
752ca01	Jing Huang	initial commit
3e6b10a	Jing Huang	add gitlog2latex.sh
1b590a1	Jing Huang	Add gitlog2latex folder
0be0a6a	Jing Huang	Delete gitlog2latex.sh in main folder
5a2cdd2	JING HUANG	Update README.md
797ede0	William Maxwell	patch file, sstf-iosched.c, latex template
eccae7b	will	adding latex makefile and script to generate latex table from the git history
09d248d	will	test script, finished latex writeup, and modified makefile

3 Questions

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

There are two purposes to this assignment. The first is to understand the main ideas behind scheduling algorithms by implementing one ourselves. The second is to become familiar with Linux kernel development. This involves reading the documentation for scheduling algorithms and understanding the build process behind compiling your own scheduler.

3.2 How did you personally approach the problem?

We started by reading the Linux elevator source code. Much of this was in `elevator.h`. We identified the functions used to add requests to the request queue and to dispatch requests. These are the

functions we needed to modify in order to implement our scheduler.

Next, we studied different types of scheduling algorithms. We read pseudocode describing the methods for scheduling, and translated it into code using the Linux kernel data structures and functions.

3.3 How did you ensure your solution was correct?

We used the `printk` function to write to the kernel log. We prepended our kernel messages with `[SSTF]`. Then we ran a script which writes to two files 1000 times each. Issuing the command `dmesg | grep [SSTF]` shows that our scheduler is being used.

3.4 What did you learn?

We learned about the uses of I/O scheduling in the kernel. We learned that scheduling is very important, since retrieving data from the hard disk is a huge bottleneck in the overall performance of the system.

We learned how to compile a scheduling algorithm, and how to tell the kernel which scheduling algorithm to use.

We learned how to read and understand kernel documentation. Specifically we learned the contents of `elevator.h` and how the data structures for interfacing with schedulers work.

Finally, we learned how the LOOK and CLOOK algorithms are implemented and learned the differences between the two algorithms.

3.5 How should the TA evaluate your work?

Apply the `.patch` file to the `yocto-linux-3.16/block` directory. We only patched the block directory, since it's the only directory we made changes to. Compile the kernel with `make -j4 all` with the SSTF scheduler selected in `make menuconfig`. Boot into the VM and run the test script provided. You should see output from the command `dmesg | grep [SSTF]`.