Assignment 3 Writeup

Operating Systems (CSE231)

Vaibhav Saxena **2019342**
vaibhav19342@iiitd.ac.in

In this assignment, we had to modify the CFS Scheduler to consider the soft real time requirements for certain processes.

Implementation :

1. I created a **field rtnicevalue** in the **sched_entity** structure in the **sched.h** file in include/linux/ folder and initialized it to 0 in the **core.c** file in kernel/sched/ folder, so that for every process the value of rtnicevalue is 0 by default.
2. To implement the modified scheduler, I changed two functions in the **fair.c** file in kernel/sched/ folder.
    - **update_curr()** - I modified this function to modify the rtnice value of a particular process every time it runs. The rtnicevalue field is decremented by delta_exec, which denotes the time taken in process execution. I also checked here that only one out of the vruntime and rtnicevalue is updated, not both. If rtnicevalue is not updated, only then is the vruntime value to be updated.
    - **wakeup_preempt_entity()** - I modified this function to find the difference between the current rtnicevalue of the process and the stored rtnicevalue, which helps, along with the vruntime, in picking the next process that is to be run.

3. Then I made a **system call rt_nice** which takes in PID and rtnice value as arguments and sets the rtnicevalue field as the input rtnice, which is the soft real time value, for the process that corresponds to the input PID. I added my system call at **440** index in the **syscall_64.tbl** file in arch/x86/entry/syscalls/ folder to add it to the list of system calls defined in the kernel. Then I added a prototype in the **syscalls.h** header file in include/linux/ folder to tell the kernel that the syscall definition exists. Finally I added the definition in the **sys.c** file in kernel/ folder.
4. In the definition of the syscall, I used **printk** to print the output in the kernel log. The kernel log can be seen from the terminal using the **dmesg** or **sudo dmesg** command in linux. We traverse all the existing processes and find the one with the pid same as the pid given to the syscall. If found we update that process' rtnicevalue with the one given to the syscall.

All these changes that I made are visible in the **diff.txt** file that I generated using the original kernel files that I downloaded and the modified kernel files.

Working :

1. **More priority** is given to the soft real time requirements of a process than its vruntime. Everytime a new process is picked to be run, rtnicevalue is **checked first**.
2. If it's rtnicevalue is **greater than 0**, that means it is a soft real time process, and we will give it priority over the non real time processes. Among all the soft real time processes the process with **min rtnicevalue** that is the process which requires the least amount of time, is run first.
3. Each time a soft real time process is run, the corresponding rtnicevalue for that process is updated. The rtnicevalue is decremented by the amount of time the process executed for.

4. If there are no soft real time processes, that is if all of them have rtnice = 0, the next process is picked according to the vruntime just like the normal CFS scheduler.

Testing :

In the **test.c** file we fork a process so that we now have two processes. In the parent process we give some rtnicevalue hence giving priority whereas in the child process we leave the rtnicevalue at default that is = 0. Then we print the time taken by each process, and we see that the parent process took less time to execute as it was given priority.

We can test other cases, by changing the rtnicevalue of both the processes through the system call.

Output :

The test.c file first outputs the **PIDs** of the two processes that were created and then prints the **time taken** by each of the processes so that we can compare them and see whether the system call worked or not.

The **kernel log** which can be seen using dmesg or sudo dmesg shows if any rtnicevalue was updated or if any error occurred or not.

Errors :

I handled two errors:

1. If the rtnicevalue that is given to the system call is **negative**, it exits the system call and prints the error in the kernel log.
2. If **no existing process** corresponding to the input PID is found, it exits the system call and prints the error in the kernel log.