Deepam Sarmah
2020050
deepam20050@iiitd.ac.in

---

Kernel: linux-5.14.3
To patch:
Kindly type in `patch -p1 -i deepam20050_patch.patch` and then type
`linux-5.14.3/arch/x86/entry/syscalls/syscall_64.tbl`
`linux-5.14.3/include/linux/sched.h`
`linux-5.14.3/kernel/sched/core.c`
`linux-5.14.3/kernel/sched/fair.c`
`linux-5.14.3/kernel/sys.c`
As and when prompted.

To test:
Compile command: `make`
To execute: `./test`

System calls:

1. change_vruntime(int curr_pid, long long add_delay) : For adding the delay in ms to the scheduler such that anytime the said process is chosen by the scheduler, it adjusts the vruntime of the said process so as to delay its selection.
2. print_nice_log(int nice_val): This function prints for each process with the same nice value as our test program the timestamps corresponding to when each process is selected and dispatched to run. This log is present in the kernel log and can be accessed by the `sudo dmesg` command.

Files modified and data structures used:

1. linux-5.14.3/include/linux/sched.h
    a. I added a new variable called add_vruntime in the struct sched_entity. This variable permanently stores the delay that needs to be added every time the vruntime is recalculated in the update_curr() function.
2. linux-5.14.3/kernel/sched/core.c
    a. Inside the __sched_fork() function I set the default value of add_vruntime as 0. If the change_vruntime() system call is called only then the default value changes, else it remains 0.
3. linux-5.14.3/kernel/sched/fair.c
    a. The update_curr() function is the function that updates the vruntime value. In code line 814 whenever the vruntime is updated I add the corresponding sched_entity's add_vruntime value.
    b. If a process which hadn't called change_vruntime() function underwent an update in it's vruntime then add_vruntime remains 0 so no change happens.

    c. In the other case if a process which called change_vruntime() function underwent an update in it's vruntime then add_vruntime which is set by the user is added every time vruntime is recalculated.
4. linux-5.14.3/arch/x86/entry/syscalls/syscall_64.tbl:
    a. I added the system call definition for the change_vruntime() function. It starts by going through all the processes using the for_each_process() function and checks for the process with the curr_pid using task_pid_nr() function. When found I then initialize the vruntime from the task_struct of that process to add_delay and initialize add_vruntime of task_struct to add_delay as well.
    b. I also added the system call definition for the print_nice_log() function. It starts by going through all the processes using the for_each_process() function and uses task_nice() function to check whether a process has the same nice value as the test program. If it does then it prints the PID of the process and the se.exec_start to the kernel log.
5. linux-5.14.3/kernel/sys.c
    a. I add the system call numbers 548 and 549 for change_vruntime() and print_nice_log() system calls in the system call table at the mentioned file.

General working:

- The code starts by taking input of the delay in ms from the user and then creates a child process using fork and then prints the PIDs of the child and parent process.
- The child process calls the change_vruntime() system call to add the required delay in ms to it's vruntime.
- The parent process doesn't call the change_vruntime() system call.
- In both the processes I run a loop that lasts for approximately 20 seconds.
- I then print the total execution time for both the processes.
- Finally I make a call to the print_nice_log() system call which prints for each process with the same nice value as our test program the timestamps corresponding to when each process is selected and dispatched to run in the kernel log.
- Proper error handling has been done in accordance with the linux man pages.

To view log:

```
sudo dmesg
```