CSE231 - Operating Systems

Assignment-3

Name: Ananya Lohani
Roll Number: 2019018
Branch: CSE

Description:

In this assignment, I have modified the **sched_entity** structure to contain a new field **rt_nice** which is the soft real-time requirement of a process(a value of x as **rt_nice** means that the process must receive at least x units of time-slice). To give priority to processes with soft real-time requirements I have modified the CFS scheduler to schedule tasks on the basis of **rt_nice** first, and vruntime second.

rt_nice System Call implementation:

I added a new field **rt_nice** to the **sched_entity** struct in **/kernel/sched/sched.h** and initialised it to 0 in the **__sched_fork()** function in **/kernel/sched/core.c**. I have added a system call **rt_nice** which takes two arguments: **PID of the process** and the **soft real-time requirement**. The system call gets the task corresponding to the given PID and sets the **rt_nice** field of its **sched_entity** to the given value.

Modified CFS implementation:

The CFS scheduler of Linux has been modified to give priority to the **rt_nice** values before giving priority to the **vruntime** of processes in the run-queue.

Functions modified:-

- **entity_before():** This function acts as a comparator between the vruntime of 2 **sched_entity** struct pointers. I added the code that compares the **rt_nice** values of the pointers first, and if both pointers' **rt_nice** is equal to 0, it compares their **vruntime**.
- **__pick_next_entity():** This function picks the next process to be executed. I modified it to check if any process in the runqueue has a non-zero **rt_nice** value and selected the process with the minimum

rt_nice to be executed. If no process has **rt_nice** greater than zero, it will execute the next process according to **vruntime**.

- **update_curr():** If the **rt_nice** value of a **sched_entity** is greater than 0, it updates **rt_nice** by subtracting from it the amount of time that the process ran.

Errors handled:-

Errors returned by the system call are handled by **perror()** in **test files**.

- **EINVAL**: If an invalid pid was given(pid not between 1 and 2147483647) or if invalid value for **rt_nice** was given(<0).
- **ESRCH**: If there's no process corresponding to the given pid.

Testing the Scheduler:

The working of the scheduler can be tested by the test files provided: **test1.c** and **test2.c**. **test1.c** takes a command line argument **rt_nice** from the user and calls the **rt_nice** system call to assign that value to itself. It runs a loop of 2000000000 and calculates internally the time it took to finish the process. **test2.c** forks a child process and assigns **rt_nice** as 200 to the parent and 0 to the child. Both the parent and the child processes execute a loop of 30000000 and calculate the time it took to finish the loop.

Expected output:

In **test1.c**, the process with a positive value of **rt_nice** took less time to be completed. The expected output is as follows:

```
→  modified-cfs-scheduler git:(main) ✗ ./test1
rt_nice: 0 sum = 2000000000 time: 6.015671 seconds
→  modified-cfs-scheduler git:(main) ✗ ./test1 100
rt_nice: 100 sum = 2000000000 time: 5.950315 seconds
```

In **test2.c**, the parent process with **rt_nice** = 200 finished the loop faster than the child process with **rt_nice** = 0. The expected output is as follows:

```
./test2
Parent Process PID: 4467, rt_nice = 200, time: 0.085729 seconds
Child Process PID: 4468, rt_nice = 0, time: 0.090100 seconds
```