OS Assignment-3: Modifying CFS Scheduler

Karanjot Singh
2019050

Description of the code:

The Linux scheduler CFS aims to maintain balance in providing processor time to tasks. The CFS maintains the amount of time provided to a given task in what's called the virtual runtime and maintains a red-black tree which is ordered on the basis of Vruntime values. Here, each node is a sched_entity object representing each task.

All the changes made as per the diff are in order to modify the scheduler in such a way that when every time a process is selected through the RB-tree, all the process' soft-realtime requirements are compared to see which one of those require the CPU more urgently than the one selected through the regular RB- tree.

The modifications made are summarised as follows:

1. In /include/linux/sched.h rtnice is declared with type u64
2. In /kernel/sched/core.c the rtnice attribute is given initial value =0
3. In /kernel/sched/fair.c the entity_before function is modified with our new comparator for rtnice values
4. update_curr() - this updates the sched_entity->vruntime and rtnice for the currently running group.
5. A system call rtnice with arguments as PID and soft_requirements is defined to update the rtnice values.

User Input:

The user enters the soft realtime requirements for a process. "Greater" requirement implies that the process has more time to execute, i.e. it should receive higher priority.

Expected Output and Interpretation:

The output shows the execution time of a process with and without the soft real-time guarantees.
As observed the processes with a higher soft realtime guarantee are scheduled before the other processes with soft realtime guarantee =0.
This demonstrates that the modified scheduler is functioning.

Sample test cases:

Enter the Soft real-time requirements (s): 13
Modified scheduler time: 22.41233
CFS scheduler time: 45.72348

Enter the Soft real-time requirements (s): 15
Modified scheduler time: 21.66234
CFS scheduler time: 40.89479

Error Values and Interpretation:

When a system call fails, it usually returns -1 and sets the variable errno to a value describing what went wrong. (These values can be found in <errno.h>.)The following errors possible:

EPERM 1 Operation not permitted - If rtnice value supplied < 0
ESRCH 3 No such process - When the PID of the process couldn't be found
Error in Fork Process - returned If the fork process fails.

References:

https://josefbacik.github.io/kernel/scheduler/2017/07/14/scheduler-basics.html
https://docs.huihoo.com/doxygen/linux/kernel/3.7/structsched__entity.html

Implemented as Assignment 3 in CSE231 - Operating Systems at IIIT Delhi