Divyansh Rastogi
2019464

Writeup

Task

Modifying the current Linux (5.9.1) CFS scheduler to suit soft real-time requirements.

Implementation:

All of the below mentioned changes could be cross-checked in diff.txt.

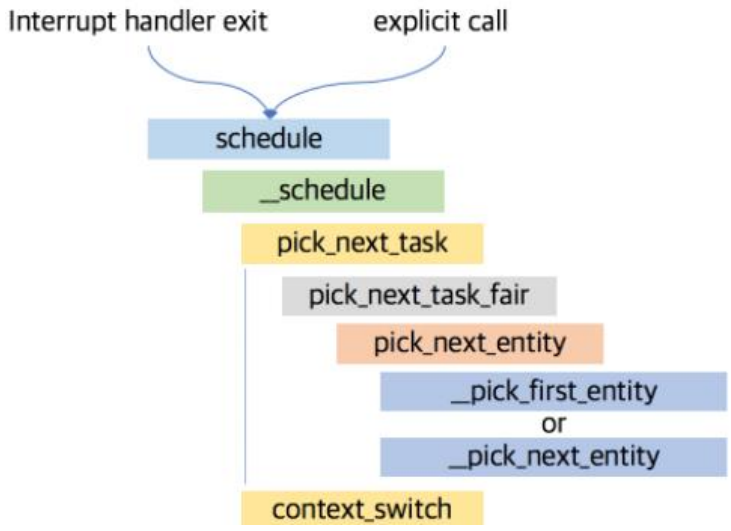
We introduce a parameter *soft_runtime* in the struct *sched_entity* which can be found under *include/linux/sched.h.* More *soft_runtime* indicates the urgency of the task to be put in execution state.

We initialize *soft_runtime* for all the *sched_entity* as 0 in *kernel/sched/core.c.*

As all the processes are stored in the *CFS(kernel/sched/fair.c) red black tree,* in order to prioritise *soft_runtime* as a parameter, first we make change in *entity_before* which is the comparison function which is found under *__enqueue_entity* which is used to enqueue an node into the red black tree.

As a task's runtime stats are updated in *update_curr* we decrement the priority of our soft real-time process whenever it is executed.

Furthermore, referencing to the given call stack during process selection ->



We make changes in *pick_next_entity* as the function starts with picking up the leftmost node from the red black tree and then checks for pre-empted processes, but we override it with the same giving priority to processes with *soft_realtime* requirements and returning the same.

We also make changes in *pick_next_task_fair*, as during the best task selection, we make a change as whenever we encounter a soft real-time process, we return its task struct *using task_of* at the very moment prioritizing it further.

Divyansh Rastogi
2019464

Checking & Error Handling:

To implement checks, we implement a system call *rt_nice* under *kernel/sys.c* and added its suitable entry under *arch/x86/entry/syscalls/syscall_64.tbl*. The purpose of this system call is to take a processes PID and change its *soft_runtime* to a given soft runtime. *Suitable error handling in case of invalid arguments and processes is implemented.*

In user space, we have *test.c* to check our kernel's accuracy, we first fork a child process and then using *getpid()* and a given *soft_runtime,* we give the child process soft real-time requirements and observe its execution for a loop of 1e9 using *timeval structs and gettimeofday()* in *ms.* During the aforementioned, the parent process *waits using wait()*, as the reason for the same being due to our systems being multi-core, with both processes simultaneously running we can't expect any fruitful output. After the same, we run a loop of 1e9 in parent process and observe its execution time. *Suitable error handling in case of failure of our system call and fork is done.*

Results:

Our new kernel maintains an accuracy of over 85% - 95% in prioritizing processes with soft real-time requirements. We observe differences ranging from 50ms to 300ms given with respect to our soft real-time proceses.