Name	NetID
Juanli Shen	jshen35
Funing Xu	fxu12
Wei Yang	weiyang4
Yunsheng Wei	wei29

1. Design
    - 1.1. Proc filesystem is used to communicate between kernel and user spaces. User jobs can write messages (REGISTRATION, YIELD, DE-REGISTRATION) to the Proc file. Kernel module uses write callback function to parse messages. Status of registered jobs can also be viewed via read callback function of the Proc file.
    - 1.2. The Linux Process Control Block is augmented with scheduling parameters, timer and list members to a new struct 'mp2_task_struct' for job scheduling.
    - 1.3. Timer interrupt is used to wake up sleeping processes, or more precisely, issue a new period for it. Each process is attached with a timer. The timer interrupt mechanism is based on two halves, where the top half is wake-up timer handler which changes the process state to READY, and the bottom half is the dispatching thread which do the context switch.
    - 1.4. Admission Control is used to ensure all registered jobs and the new job can be scheduled without missing deadline.
    - 1.5. Slab allocator is used to allocate object for 'mp2_task_struct'. Specifically, slab allocator allocates the memory from a lookaside cache for objects with the same size. Usually slab allocator helps to improve allocation performance.
    - 1.6. Dispatching thread is used to do proper context switch after: 1) receiving Yield message; 2) wake-up timer interrupts. Dispatching thread is a kernel thread, and is asleep unless context switch is needed.
    - 1.7. Semaphore mutex is used to ensure mutual exclusion when different threads want to get into the critical section. One advantage of using semaphore is that the thread will sleep while waiting for the resource instead of busy waiting.
2. Implementation
    - 2.1. During context switch, the dispatching thread must be set to sleep state before doing its work so that it can guarantee the highest priority task is chosen before dispatching. It can solve the problem that during the context switch a new task with higher priority is ready. (Please refer to the comment in the code to see more details.)
    - 2.2. When handling yield, the lock must be released before setting the task to sleep state. Otherwise, there may be deadlock. (Please refer to the comment in the code to see more details.)
    - 2.3. seq_file interface is used to read Proc files and print out large output without worrying about page size limit in Linux kernel.
    - 2.4. A dispatching thread is initialized to have FIFO priority=99 so that when RMS scheduling is needed, an awake dispatching thread can be chosen by Linux scheduler. The dispatching thread will sleep until being waken up. When it's waken up, the dispatching thread iterate the task list and picks a task with max priority and has state = READY or

RUNNING. The preempted job goes to sleep with NORMAL priority=0. The picked job is scheduled with FIFO priority=90, and its state is set to RUNNING.

2.5. When a user job writes to Proc files, a callback function parses the message string and switch among REGISTRATION, YIELD or DE-REGISTRATION to do the according work.

2.6. Upon REGISTRATION of a new job, Admission Control is used to ensure the sum of processing time by period among all registered jobs and the new job is less than or equals to 0.693. Here the ratio of processing time by period times to 1000 to avoid floating number computation which is too expensive. After a new job successfully registers, its state is set to SLEEPING. And a wake-up timer is initiated without starting.

2.7. Upon the initial YIELD message, set the timer to wake up at the beginning of the next period. The yielding job goes to sleep; the dispatching thread is called to do context switch.

2.8. Upon wake-up timer expiration, its associated job's state will set to READY, and dispatching thread is called.

2.9. When linked list of 'mp2_task_struct' is modified by a thread, a semaphore is used to ensure mutual exclusion.

3. How to test

3.1. make

3.2. sudo insmod rate_monotonic_scheduler.ko

3.3. ./app 100000000 2 1000 200 & ./app 10 5 300 10 &
And you can see the wakeup time of the processes.

3.4. dmesg
You can grep on it to check, e.g., preempt.
Also, you can see a lot of logs as following.

- Registration
- Yield
- Wake up process when ready to run
- Preempt current job if there is another job ready to run and has higher priority
- De-registration

3.5. sudo rmmod rate_monotonic_scheduler