CSIE 5374 Assignment 3 (Due on 04/16 23:59)

In assignment 3, you are asked to create a new scheduler class for the Linux kernel based on Linux 5.15. The scheduler should utilize the Multilevel Queue (MLQ) algorithm. Additionally, your scheduler must support various scheduler-related system calls in Linux.

Assignment 3 is a group task. You should collaborate with your group members on this assignment.

Development Tips

In the following sections, we assume you already know how to compile the Linux kernel and create a filesystem image and run it with `qemu-system-aarch64`. Therefore, we will skip those steps already mentioned in assignment 1.

Requirements

Implementation (60%)

Your are required to create a new scheduler class named `sched_mlq_class` for a new scheduling policy called `SCHED_MLQ`. The value assigned to `SCHED_MLQ` should be 7. The priority of `sched_mlq_class` is expected to be higher than `sched_fair_class` but lower than `sched_rt_class`.

The priority of a task using the `SCHED_MLQ` policy should be 1 (highest priority), 2, or 3 (lowest priority). Each priority level is linked to a task queue from which the scheduler selects tasks to schedule. Tasks with higher priority must always be scheduled first. For instance, no priority 2 task can be scheduled if there exists at least one priority 1 task available for scheduling. Additionally, a task with a higher priority should preempt a task with a lower priority.

Tasks with priority 1 or 2 are to be scheduled using the round-robin algorithm, using time slices of 50ms and 100ms, respectively. Tasks with priority 3 should be scheduled using the First-Come First-Served (FCFS) algorithm.

Setting `SCHED_MLQ` as the scheduling policy for a task is only possible by invoking the `sched_setscheduler()` system call. You should not change the default scheduling policy for `init` and kernel threads. The only way to alter a task's priority is through invoking the `sched_setparam()` function. Otherwise, a task's priority must remain unchanged.

Your implementation will undergo testing via scheduler-related system calls in Linux. The expected behavior of your scheduler for each invoked system call is outlined below:

1. `sched_setscheduler()`
   This system call should be able to set the task's scheduling policy to `SCHED_MLQ` and assigning the task a priority level (1, 2, or 3) via the `sched_priority` field in the `param` parameter. The system call must return an error if an invalid priority is provided.

2. `sched_getscheduler()`
   The system call should return `SCHED_MLQ` if the task's policy is set to it.
3. `sched_setparam()`
   The system call should be able to set the task's priority, similar to the `sched_setscheduler()` system call.
4. `sched_getparam()`
   The system call should retrive the `sched_param` of the task using the `SCHED_MLQ` policy.
5. `sched_get_priority_max()`
   The system call should return 3 if the task is using the `SCHED_MLQ` policy.
6. `sched_get_priority_min()`
   The system call should return 1 if the task is using the `SCHED_MLQ` policy.
7. `sched_setaffinity()`
   The system call should be able to set the CPU affinity mask for a task using the `SCHED_MLQ` policy. The scheduler should assign a task to one of the specified CPUs on which it is eligible to run.
8. `sched_getaffinity()`
   The system call should retrive the `cpu_set_t` of the task using the `SCHED_MLQ` policy.

Note that you may not need to modify all of these system calls to support the newly added scheduler class. As long as you implement your scheduler class correctly, some of these system calls will function properly without any modifications.

For guidance, you can examine `kernel/sched/rt.c` and `kernel/sched/fair.c` to understand how a scheduler class is implemented. Additionally, refer to `kernel/sched/core.c` to understand how the Linux scheduler operates.

Write-up (30%)

In addition to the implementation, you should conduct experiments on your new scheduler class to show its features and demonstrate how priority influences task execution when using your new scheduler.

Each group is required to provide a write-up about the assignment. The write-up should include explanations of your source code (e.g., how your code satisfies the specified requirements), the results obtained from your scheduler experiments, and contributions from each group member.

Bonus (10%)

You are encouraged to optimize the scheduler beyond the stated requirements to enhance its efficiency. For instance, you could implement features such as load balancing, dynamic adjustment of task priorities, etc. Since these optimizations may conflict with the above requirements, they should be disabled by default. In your write-up, specify the optimizations made, how to activate them, and the experiments conducted to observe their effects.

Homework submission

You should submit the assignment via NTU Cool.

Submission Format

You are required to submit the kernel patch for Linux v5.15 and a write-up file. Compress both files into `hw3.zip`, and upload the zip file to NTU Cool.

Grading criteria

Please run checkpatch against the changes you made to the Linux kernel.

```
./scripts/checkpatch.pl patch
```

You will lose **5 points** if checkpatch reports either warnings or errors.

In your implementation, you are required to properly manage resources (free allocated memory), handle errors, and return error codes to userspace.

Your submitted kernel patch will be applied to the Linux v5.15 kernel, and the patched kernel will be tested in the QEMU Linux environment. If we fail to apply your patch to the kernel source, or the patched Linux fails to compile, you will get zero points.

Plagiarism policy

You are allowed to reference sources from the internet. If you do, please attach all of your references in the write-up. If we find that your code is similar to other's from the internet, we will count it as plagiarism if we could not find the corresponding references. You will automatically get zero points for the assignment.

Late Policy

We do not accept late submissions for this assignment. Please start the assignment early.