CS444/544 Lab

Intro to Linux Scheduler
Process Scheduling

• Processes in Linux Kernel are just a group of threads sharing the same TGID (thread group ID)
• Kernel schedule threads directly
• Threads are called “task”
task_struct

• The data structure used in Linux that contains all the information about a specific task
• kernel/linux/sched.h: 1043
Task Classification

- CPU-bound vs. I/O bound
- Real Time vs. Normal
- Task Priority Values
  - Higher priorities have numerical smaller value
  - Real time: 1 - 99
  - Normal: 100 – 139
Scheduling Class

• Modular structure of scheduler
• kernel/sched/sched.h: around line 980

```c
struct sched_class {
    const struct sched_class *next;

    void (*enqueue_task) (struct rq *rq, struct task_struct *p, int);
    void (*dequeue_task) (struct rq *rq, struct task_struct *p, int);
    void (*yield_task) (struct rq *rq);
    bool (*yield_to_task) (struct rq *rq, struct task_struct *p, bool);

    void (*check_preempt_curr) (struct rq *rq, struct task_struct *p);

    struct task_struct * (*pick_next_task) (struct rq *rq);
    void (*put_prev_task) (struct rq *rq, struct task_struct *p);
}
```
Scheduler Classes

- Scheduler classes forms a linked list
  - stop_sched_class →
  - rt_sched_class →
  - fair_sched_class →
  - idle_sched_class →
  - NULL
Stop and Idle Tasks

- Generally system tasks
- Stop task cannot be preempted by any task
- Idle Task will be preempted by any task
- Real-time and normal tasks are between them
Main Runqueue

- Defined in kernel/sched/core.h (till 3.13.0)
- Keeps track of all runnable tasks assigned to a particular CPU
- Manages various scheduling statistics about CPU load or scheduling domains for load balancing
More about Main Runqueue

• a lock to synchronize scheduling operations for this CPU
  – raw_spinlock_t lock;

• pointers to the task_structs of the currently running, the idle and the stop task
  – struct task_struct *curr, *idle, *stop;

• runqueue data structures for fair and real time scheduling classes
  – struct cfs_rq cfs;
  – struct rt_rq rt;
The Scheduler Entry Point

• schedule(): kernel/sched/core.c: 2523
• Find the next task to be run and assign it to the local variable next.
• Then executes a context switch to that new task
Go through the schedule()

• Disable preemption (Nobody can preempt scheduling thread)
• Lock the Main Runqueue to make sure only one thread is modifying the struct
• Remove inactive previous task
• Ask each class to pick next task
• Execute context_switch()
Call the scheduler

• Regular runtime update of currently scheduled task
  – scheduler_tick() : kernel/sched/core.h:2303
Tasks go to sleep

• Currently running task goes to sleep

/* ‘q’ is the wait queue we wish to sleep on */

DEFINE_WAIT(wait);

add_wait_queue(q, &wait);
while (!condition) { /* condition is the event that we are waiting for */
  prepare_to_wait(&q, &wait, TASK_INTERRUPTIBLE);
  if (signal_pending(current))
    /* handle signal */
    schedule();
}
finish_wait(&q, &wait);
Tasks wake up

- Event calls `wake_up()` on wait queue which call scheduler function `try_to_wake_up()`: Line1582
  - It puts the task to be woken back into the runqueue.
  - It wakes the task up by setting its state to `TASK_RUNNING`.
  - If the awakened task has higher priority than the currently running task, the `need_resched` flag is set to invoke `schedule()`.
Complete Fair Scheduler (CFS)

• fair_sched_class in kernel/sched/fair.c
• Idea: all tasks get a portion of processing power based on the priority
• Example: Two tasks with priority 105 and 125
  – In 6 timeslices, Task 105 get 5 and task 125 get 1.
Virtual Runtime

• A virtual runtime (vruntime) was introduced for each task
• The vruntime is updated with the elapsed time since it was scheduled.
• If the vruntime of another task in the runqueue is smaller than the current tasks', a rescheduling is executed and the task with the smallest vruntime is selected to run next.
How priority is handled

• A high priority tasks' vruntime grows slower than the vruntime of a low priority one
Run Queue – Red Black Tree

- Red-Black tree is used to select next task
- Each node represents a task and they are ordered by the task's vruntime
- Insert and delete in $O(\log n)$
More to read

• Code Reading and Cross Reference

• Linux Kernel Scheduling