Computer Operating Systems, Practice Session 9

Linux Scheduler

Mustafa Ersen (ersenm@itu.edu.tr)

Istanbul Technical University
34469 Maslak, İstanbul

22 April 2015

Today

Computer Operating Systems, PS 9

Scheduling Principles

Linux Schedulers

Multitasking

Multitasking capable algorithms are classified into two types according to preemption type of the tasks:

- Cooperative multitasking (a.k.a time sharing): Process decides when to leave the processor.

  If one program does not cooperate, it can hog the CPU.

- Preemptive multitasking: Every task has an upper and a lower limit on the time interval on CPU retrieval time. Tasks can NOT decide their own CPU time.

Timeslice - Quantum

The longest period of time for a task to run without a preemption from the scheduler is called timeslice (or quantum).

(The scheduler is run once every time slice to choose the next process to run)

- Too Short: Context switching wastes time and cache does not stay fresh
- Too Long: Processes wait more to retrive CPU (Poor concurrency)

Priority

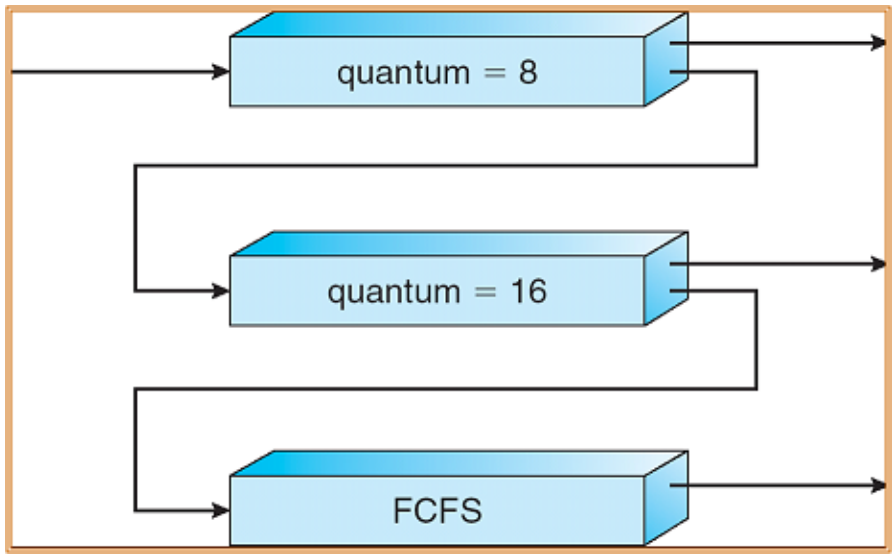
- A process's priority is determined by one of the two parameters: nice and RTPRIO
- nice: range from -20 (most favorable scheduling) to 19 (least favorable).
- RTPRIO (the realtime or idle priority): is in interval: [0, 31]. 0 is the highest priority.
- Real-time processes have higher priority than the others.

MLFQ - Multi-Level Feedback Queues (Linux 2.5 prior)

Processes are kept in the queues according to their priorities (short jobs and I/O bound processes are more favorable).

A process that can not finish its job within the given quantum is appended to the end of the queue in the lower level.

Third queue is on a FCFS basis (First-come, first-served).



O(1) Scheduler (Linux 2.5-2.6.23)

- O(1) Scheduler performs scheduling in constant time (`O(1)`), so it can scale well with increasing number of tasks.
- Two priority arrays are maintained: *active* and *expired*.
- Initially, all tasks are in the *active* priority array.
- A task that runs out of its timeslice is preempted and moved to the *expired* priority array.
- When there is no *active* process, two priority arrays are swapped.
- Processes having same priority is served with a "round robin" approach (circular order).

CFS - Completely Fair Scheduler (Linux 2.6.23 after)

- For each process, instead of a certain amount of time (timeslice), a proportion is assigned by considering its priority.
- Two processes having the same priority retrieve same proportion.
- There exist scheduling classes enabling to apply different principles for different classes (e.g., real time processes).
- O(log(n)) complexity for scheduling (based on red-black trees).

Scheduling Classes

- Scheduler classes has been defined as an extensible hierarchy of scheduler modules for providing more flexibility to the CFS scheduler.
- Scheduling classes keep specific queues and enable scheduler to operate with different principles on different scheduling classes.
- Scheduling classes are implemented via `sched_class` kernel data structure which provides event based functions to the programmers:
  - `enqueue_task()`: Called when a task enters a runnable state. It puts the task into the RB tree and increments the `nr_running` variable.
  - `dequeue_task()`: When a task is no longer runnable, this function is called to keep it out of the RB tree and decrement the `nr_running` variable.
  - `yield_task()`: This function is a dequeue followed by an enqueue.
  - `check_preempt_curr()`: Checks if a task that entered the runnable state should preempt the currently running task.
  - `pick_next_task()`: Chooses the most appropriate task eligible to run next.

Scheduling Principles

- A process can be classified into two types given below according to its general characteristics:
  - I/O Bounded
  - Processor Bounded
- Below scheduling policies are assigned to scheduling classes regarding to general characteristics of processes that will run on the system:
  - SCHED_NORMAL(POSIX:SCHED_OTHER)): Scheduling policy used for regular tasks.
  - SCHED_BATCH: For "batch" style execution of processes: a version of CFS that makes less process exchanges. Makes better use of the cache memory.
  - SCHED_FIFO/_RR: "real-time" policies for special time-critical applications that need precise control over the way in which runnable processes are selected for execution.

Scheduler Data Structure and `vruntime`

- The scheduling related information are kept in the data structure: `sched_entity` defined in `<linux/sched.h>`.
- `vruntime` variable in this data structure is an important variable representing the *virtual run time* of the process.
  (This virtual run time value is the normalized form of the real running time with respect to the number of the waiting prosesses)
- CFS selects the process with lowest `vruntime` value.

EDF - Earliest Deadline First Scheduling

- EDF(Earliest Deadline First Scheduling): Is based on giving the CPU time to the process which is closest to its deadline. (is included in Linux kernel 3.14 as SCHED_DEADLINE scheduling class).
- Optimality is proven for the systems having single processor and preemptive capability.
- Requirement of knowledge on all of the deadlines associated with each process is one of its restrictions.