Lecture 5

Linux process scheduling

The Linux scheduler

- It gives preference to interactive processes and processes with lower nice values.
- Key features:.
  - There are two multilevel feedback queues/CPU, for active and expired designated processes.
  - It uses a bitmap to quickly determine which queue is the highest priority nonempty one.
  - Interactive processes that completely use their quantum are placed back into the active queue and are scheduled round-robin – these are processes that were recently blocked for a significant amount of time.
  - Compute-bound processes that completely use their quantum are moved to the expired queue.
  - When the active queue becomes empty, it is swapped with the expired queue.
  - Priority and time slice length are computed each time a process is placed into a queue.
  - Processes scheduled according to the SCHED_FIFO policy are real-time processes with a higher priority than any time-sharing process.
  - The same as above for SCHED_RR but with finite time slices.

Priorities

- Linux has two classes of processes, real-time (RT) and time-sharing (TS).
- RT are scheduled according to the SCHED_FIFO or SCHED_RR policies. They have a static priority value in the range [0, 99].
- TS processes are scheduled according to the SCHED_NORMAL policy. Their priority is in the range [-20, 19]. Internally, they are scaled to the range [100, 139].
- Smaller values correspond to higher priorities.
- Each TS has a *nice* value and *sleep_avg* value which is the amount of time spent recently blocked.
- On a transition to Ready, sleep_avg is incremented by the amount spent on blocked up to a max of MAX_SLEEP_AVG (1 sec). For each clock tick spent running, sleep_avg is decremented by some period of time down to 0. From this measure, the effective priority is computed as:

$$p = n + \frac{s}{M_s} M_b - \frac{M_b}{2}$$

- Where n is the nice value in the internal representation, s is the average sleep time, M_s is the max average sleep time and M_b is 10.
- The effect of this is to linearly map the range of average sleep times, 0 – MAX_SLEEP_AVG onto the range [-5, 5] and add to nice.
- If the value resulted for p is outside the range [100, 139], it is set to the corresponding end of the range.
- As a result, processes that spent a lot of time blocked are given a priority boost.
- Additionally, Linux also adjusts the length of a time slice based on nice: negative values of nice determine a longer time slice.

Handling RT/TS processes

- For each clock tick, *scheduler_tick()* is called:
- If the current process is the idle one and the system has multiple cores, rebalance_tick() rebalance the load; otherwise it does nothing.

- If the current process is in the expired queue, then the process is preempted and reschedule is called - this situation is considered an error.

- The priorities of RT processes are never adjusted and they never move to the expired queue.
- These processes (SCHED_FIFO policy) are allowed to run until they yield the CPU. For a RR RT process, its time slice is finished, therefore it'll be put at the back of the queue and the reschedule flag will be set.

- For a TS process, when its time slice is finished, the process is pulled out of its queue and its priority and quantum are recomputed. The question is in what queue to put this process now, the active or the expired one ?
- If the process is compute-bound or if processes in the expired queue are starving, then the process will be moved into the expired queue.
  - the first condition is based on the current value of nice and the current priority boost;
  - does enough time passed such that all ready processes have had a chance to get as much time as the max sleep average ? If the answer is yes, the processes are put into the expired queue.

- If there is a process with a very long time slice and there are processes with the same priority, the time slice is split such that other processes get some time. The preempted process is put back in the active queue.
- If scheduler_tick() determines that a new process needs to be scheduled, it sets a flag in the current process that indicates that reschedule is needed.
- Before returning from the clock interrupt, the kernel checks that flag. If the flag is clear, the kernel returns from interrupt. If it is set, the kernel calls *schedule(),* prior to returning from interrupt.

The Linux scheduler

- It runs in constant time !
- The first thing it does is to check for an error condition: if the current process is flagged as nonpreemptible, the scheduler should not be run !
- The same applies if the current process is the idle one and it's not running.
- If the prev process has just blocked, it is taken out of the active queue by deactivate_task().
- The active ready queue has some ready processes. The process at the head of the highest priority nonempty level should be selected.
- The function *sched_find_first_bit()* is architecture-dependent to take advantage of any special features found in the CPU instruction set; i.e., may processors have an instruction that will indicate the first bit in a word set to 1.
- After that, *next* is set to point to the first process table entry in the list.
- The following operation is the context switch, executed by calling *context_switch():*
  - The first stage is to switch all the memory management details to the new process.
  - The second is to switch registers and the stack so that the return from interrupt goes back to the new process.

- If there are no processes in the ready state, there will be an attempt to balance the load. If this is not possible, a switch to the idle process will take place.
- If the active queue has no processes it is time to swap between the active and expired queues.

Summary

- There are different sorts of OS that meet the requirements of targeted application areas. One key difference is how they deal with concurrency and the implementation of the control flow in programs (process, thread, activity).
- Generally, the set of system calls and process states is very similar.
- The scheduler plays a key role within the kernel. It implements algorithms for resource allocation to processes.
- Linux is a very popular OS that implements innovative solutions, especially regarding multicores.
- Priority and time slice are recomputed in order to favour interactive processes.

Questions

- What does the scheduler_tick() system call ?
- How does the scheduler() work ?
- What periods of time are computed during the scheduling process and how are they used ?
- What are the differences between RT and TS processes ?
- How and when is the priority updated ?