Lecture 3

Process scheduling

1. Taking control of the CPU
2. Common scheduling strategies
3. How to make scheduling fair?

I. Priority scheduling

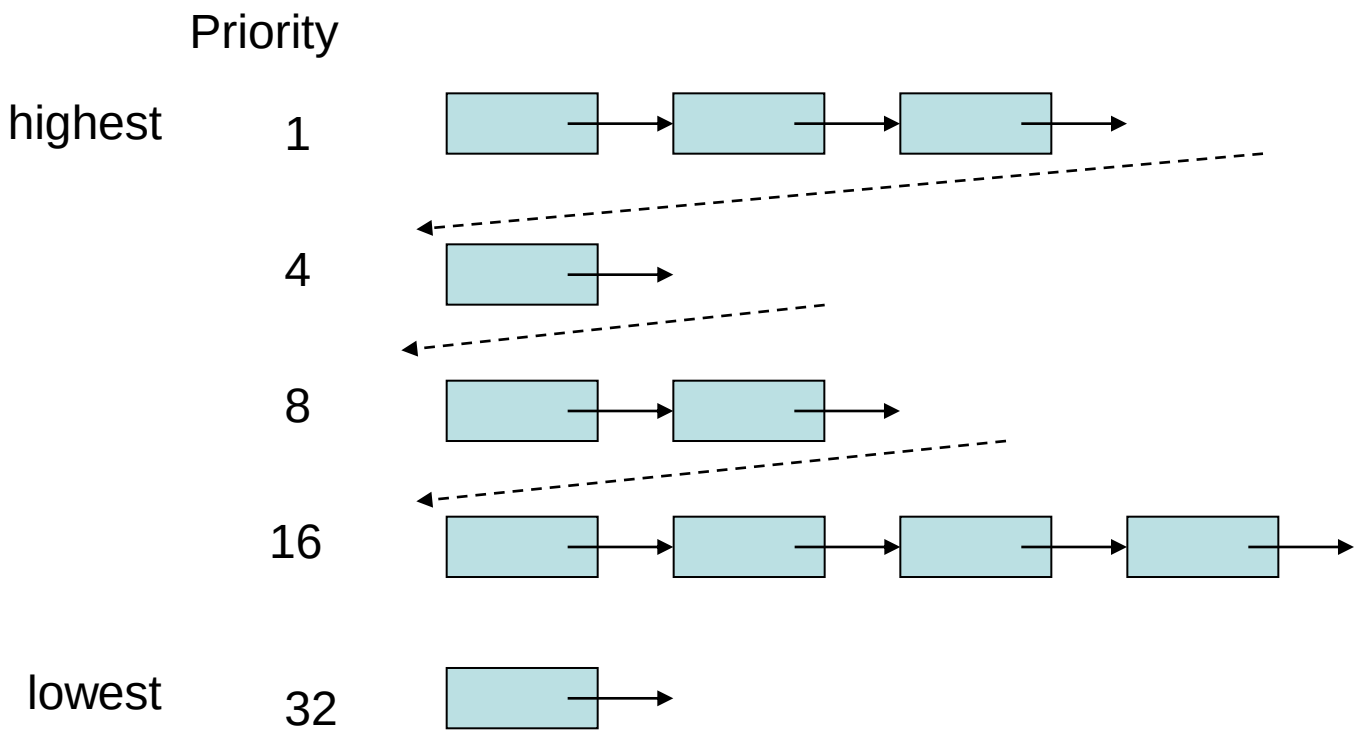
- Processes are different: some of them are interactive, others are computing demanding and therefore need to be dealt with differently in order to provide system responsiveness.
- One solution: assign priorities to processes. Fast, interactive processes will have higher priorities than computation strong ones.
- Note: Kernel processes have higher priorities than user processes.
- The priority of user processes is given considering the user or process attributes.
- If there are several processes with the same priority, they are scheduled in a round-robin manner.

- Priority is denoted by a small integer, generally a smaller value indicating a higher priority.
- Priority can be changed at runtime according to the process behaviour; for example, if a process takes too long to complete, its priority will be lowered.

Priority scheduling: multilevel feedback queues

- This is one implementation of dynamic priorities.
- There are several queues, each associated with a priority/level.
- Initially, a process gets a priority that puts it on a certain level.

- If not completed, after consuming each time slice, the process priority is lowered to the next level, until it reaches the lowest acceptable priority. At that level, the strategy is round-robin.
- However, after being blocked, the process gets a higher priority (priority boost). Consequently, during its existence, one process can have a priority that varies within a defined range.



Multilevel feedback queue

A process for power management

- Many systems have an idle process, which has the lowest priority. When there is no other process to execute, the CPU is given to the idle process that makes some cleaning and switches the system into sleep state(-s).

- The idle process implements the kernel power policy manager. It owns the decision-making and the set of rules used to determine the appropriate frequency/voltage operating state. It may make decisions based on several inputs, such as end-user power policy, processor utilization, battery level, or thermal conditions and events.
- The processor driver is used to make actual state transitions on the kernel power policy manager's behalf.

Fairness: adjusting scheduling parameters

- Priority
  - Dynamic priorities allow to avoid process starvation when, for example, a medium-level priority process is computation strong and never blocks. Lower priorities processes will starve waiting for their time slice. In this case, their priorities can be raised at the medium or even higher level.

- Time slice size
  - The time slice (quantum) can be different for each priority level. For example, the highest priority level will have the shortest time slice, and then this can be increased exponentially for lower level priorities; if the base quantum is q, level i will have the time slice $2^i q$.

Priority inversion

- Priority inversion occurs when two or more threads with different priorities are in contention to be scheduled. Consider a simple case with three threads. Thread 1 is high priority and becomes ready to be scheduled. Thread 2, a low-priority thread, is executing code in a critical section. Thread 1, the high-priority thread, begins waiting for a shared resource from thread 2. Thread 3 has medium priority. Thread 3 receives all the processor time, because the high-priority thread (thread 1) is waiting for shared resources from the low-priority thread (thread 2). Thread 2 will not leave the critical section, because it does not have the highest priority and will not be scheduled.

- The scheduler solves this problem by randomly boosting the priority of the ready threads (in this case, the low priority lock-holders). The low priority threads run long enough to exit the critical section, and the high-priority thread can enter the critical section. If the low-priority thread does not get enough CPU time to exit the critical section the first time, it will get another chance during the next round of scheduling.

II. Two-level scheduling

- Sometimes, there are too many processes that can't fit in the main memory in the same time. Therefore some will have to be stored on the disk. However the process of restoring the process in the main memory while other(-s) are saved on the disk is time consuming (can lead to the thrashing phenomenon).

- One solution is to use *two-level scheduling*:

  – a higher-level, long-term scheduler that runs more slowly will select the subset of processes resident in the main memory;

  – these processes are then managed by a different scheduler, lower-level and short-term.

III. Real-time scheduler

- In real-time applications, computing systems react to events signalled by interrupts. The interrupt triggers the scheduler to give control to the respective event handler.
- If more than one occurs in the same time, priority and deadlines are considered. For example, if a new event has a higher priority or a tighter time requirement than the current running process, then the scheduler will preempt the existing one.
- One popular technique is earliest deadline first (EDF). For each process, there is an attribute value that indicates the time by which it should be completed.
- The scheduler always selects the process with the earliest deadline. After the process used its time slice, the deadline is updated and the process is added to the ready list.
- Generally, the new deadline is computed by adding a constant period to the time at which the time slice ended.
- Example: time slice = 100 ms and three processes, *a* with const period of 300 ms, *b* with 500 ms and *c* with 1000ms.

  All are ready at t = 0 and have deadlines equal to their periods.

  a is selected first, its next deadline is 400ms which makes it the next candidate. Then, the deadline is 200 + 300 = 500 ms, after b. b is scheduled and its new deadline is 300 + 500 = 800 ms,…

Conclusions

- Scheduling is a key service of the kernel.
- Its algorithm is dictated by the nature of the applications run by that computer.
- Scheduler's parameters can sometime be modified dynamically.
- All processes need to be treated fairly.

References

- Brian L. Stuart, Principles of Operating Systems, 2009, Thomson Learning.
- http://www.intel.com/technology/itj/2007/v11i4/9-process/2-intro.htm
- http://lwn.net/Articles/80911/