【OS】Day12(2)

【Ch7】CPU Scheduling(2)

7.6 A New Metric: Response Time

We define response time as the time from when the job arrives in a system to the first time it is scheduled.

$$T_{response} = T_{firstrun} - T_{arrival}$$

STCF and related disciplines are not particularly good for response time.

If three jobs arrive at the same time, the third job has to wait for the pervious two jobs to run entirely before being scheduled just once.

Thus, we are left with another problem: *how can we build a scheduler that is sensitive to response time?*

7.7 Round Robin

To solve this problem, we will introduce a new scheduling algorithm, classically referred to as Round-Robin(RR) schedulng.

The basic idea is simple: instead of running jobs to completion, RR runs a job for a time slice(sometimes called a scheduling quantum) and then switches to the next job in the run queue. It repeatedly does so until the jobs are finished.

For this reason, RR is sometimes called time-slicing.

Note that the length of a time slice must be a multiple of the timer-interrupt period; thus if the timer interrupts every 10 millisecdons, the tiem slice could be 10, 20, or any other multiple of 10ms.

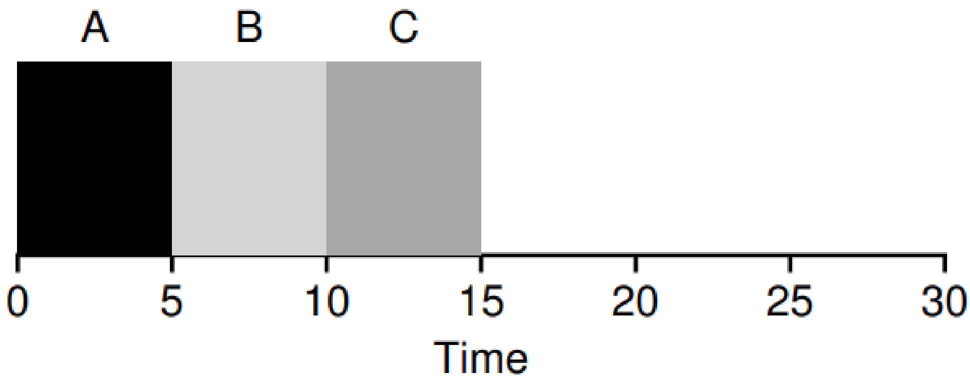

Figure 7.6: **SJF Again (Bad for Response Time)**

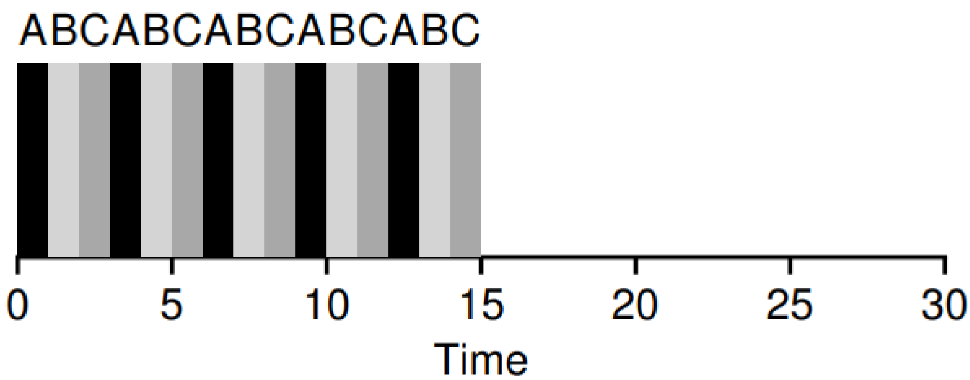

Figure 7.7: **Round Robin (Good For Response Time)**

The length of the time slice is critical for RR. The shorter it is, the better the performance of RR under the response-time metric.

However, making the time slice too short is problematic: suddently the cost of context switching will dominate overall performance.

It turnaround time is our metric, then RR is one of the worse policies. For the example above, A finishes at 13, B at 14, and C at 15, thus have an average turnaround time of 14s.

More generally, any policy(such as RR0 that is fair(i.e. evenly divides the CPU among active processes on a small time scale) will perform poorly on metrics such as turnaround time.

This is an inherent trade-off:

- If you are willing to be unfair, you can run shorter jobs to completion, but at the cost of response time
- If you instead value fairness, response time is lowered, but at the cost of turnaround time.

7.8 Incorporating I/O

A scheduler clearly has a decision to make when a job initiates an I/O request, because the currently-running job won't be using the CPU during the I/O; it is blocked waiting for I/O completion.

If the I/O is sent to a hard disk drive, the process might be blocked for a few milliseconds or longer, depending on the current I/O load of the drive. Thus, the scheduler should probably schedule another job on the CPU at that time.

Assume we are trying to build a STCF scheduler. H*ow should such a scheduler account for the fact that A is broken up into 5 10-ms sub-jobs, whereas B is just a single 50-ms CPU demand?*

A common appraoch is to treat each 10-ms sub-job of A as an indepenet job. Thus, every time a sub-job of A is complete, only B is left, and it begins running.

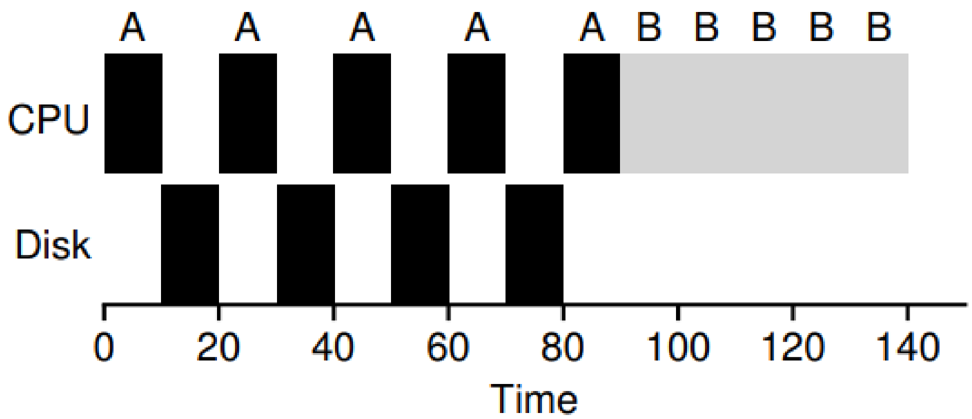

Figure 7.8: **Poor Use Of Resources**

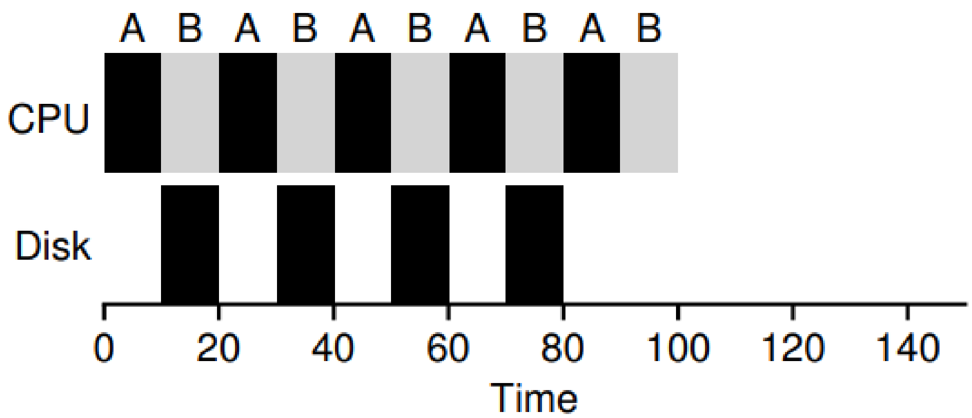

Figure 7.9: **Overlap Allows Better Use Of Resources**