SCHEDULING - AN INTRODUCTION

In a multiprogramming system, frequently multiple process competes for the CPU at the same time. When two or more process are simultaneously in the ready state a choice has to be made which process is to run next. This part of the OS is called Scheduler and the algorithm is called scheduling algorithm. Process execution consists of cycles of CPU execution and I/O wait. Processes alternate between these two states. Process execution begins with a CPU burst that is followed by I/O burst, which is followed by another CPU burst then another i/o burst, and so on. Eventually, the final CPU burst ends with a system request to terminate execution.

Long term and short term scheduling:

If we consider batch systems, there will often be more processes submitted than the number of processes that can be executed immediately. So incoming processes are spooled (to a disk).

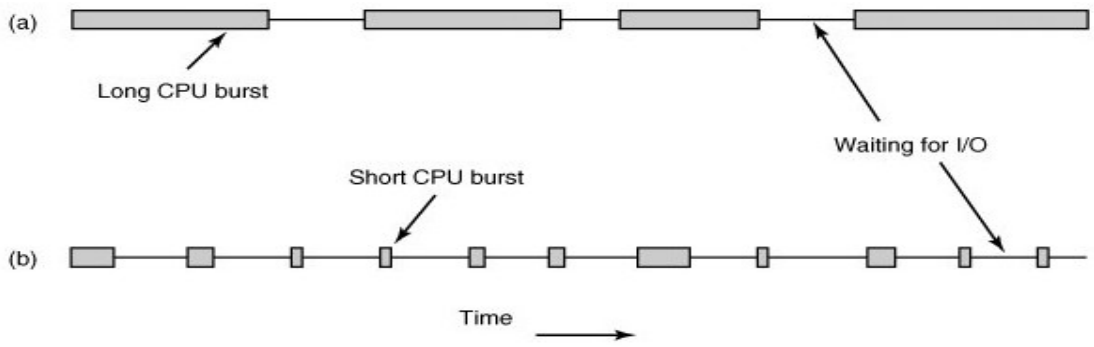
The long-term scheduler selects processes from this process pool and loads selected processes into memory for execution.

The short-term scheduler selects the process to get the processor from among the processes which are already in memory.

The short-time scheduler will be executing frequently (mostly at least once every 10 milliseconds). So it has to be very fast in order to achieve a better processor utilization. Time-sharing systems (mostly) have no long-term scheduler. The stability of these systems either depends upon a physical limitation (number of available terminals) or the self-adjusting nature of users (if you can't get response, you quit). It can sometimes be good to reduce the degree of multiprogramming by removing processes from memory and storing them on disk. These processes can then be reintroduced into memory by the medium-term scheduler. This operation is also known as swapping. Swapping may be necessary to improve the process mix or to free memory.

Process Behavior:

Fig: . Bursts of CPU usage alternate with periods of waiting for I/O. (a) A CPU-bound process. (b) An I/O bound process



CPU Bound(or compute bound): Some process such as the one shown fig a. above spend most of their time computing. These processes tend to have long CPU burst and thus infrequent I/O waits. example: Matrix multiplication
I/O Bound: Some process such as the one shown in fig. b above spend most of their time waiting for I/O. They have short CPU bursts and thus frequent I/O waits. example: Firefox

Scheduling Criteria:

Many criteria have been suggested for comparison of CPU scheduling algorithms.

CPU utilization: we have to keep the CPU as busy as possible. It may range from 0 to 100%. In a real system it should range from 40 – 90 % for lightly and heavily loaded system.

Throughput: It is the measure of work in terms of number of process completed per unit time. For long process this rate may be 1 process per hour, for short transaction, throughput may be 10 process per second.

Turnaround Time: It is the sum of time periods spent in waiting to get into memory, waiting in ready queue, execution on the CPU and doing I/O. The interval form the time of submission of a process to the time of completion is the turnaround time. Waiting time plus the service time.
Turnaround time= Time of completion of job - Time of submission of job. (waiting time + service time or burst time)

Waiting time: its the sum of periods waiting in the ready queue.

Response time: in interactive system the turnaround time is not the best criteria. Response time is the amount of time it takes to start responding, not the time taken to output that response.

Types of Scheduling:

Scheduling algorithms can be divided into two categories with respect to how they deal with clock interrupts.
1.Preemptive Scheduling
2.Non preemptive Scheduling
Nonpreemptive scheduling algorithm picks a process to run and then just lets it run until it blocks (either on I/O or waiting for another process) or until it voluntarily releases the CPU. Even it runs for hours, it will not be forceably suspended. In effect no scheduling decisions are made during clock interrupts.
In contrasts, a **preemptive** scheduling algorithm picks a process and lets it run for a maximum of some fixed time. If it is still running at the end of the time interval, it is suspended and the scheduler picks another process to run (if one is available). Doing peemptive scheduling requires having a clock interrupt occur at the end of time interval to give control of the CPU back to the scheduler. If no clock is available, nonpreemptive scheduling is only the option.

CPU scheduling decisions may take place under the following four circumstances:

1. When a process switches from the running state to the waiting state (for example , I/O requests, or invocation of wait for the termination of one of the child process).
2. When a process switches from the running state to the ready state (for example, when an interrupt occurs).
3. When a process switches from the waiting state to the ready state(for example completion of I/O).
4. When a process terminates.

Source : http://dayaramb.files.wordpress.com/2012/02/operating-system-pu.pdf