OPERATING SYSTEMS

by
Marwa Yusuf

Lecture 9
Sunday 22-11-2020

Chapter 2 (2.4 to 2.4.2)

Processes and Threads

Scheduling

Scheduling

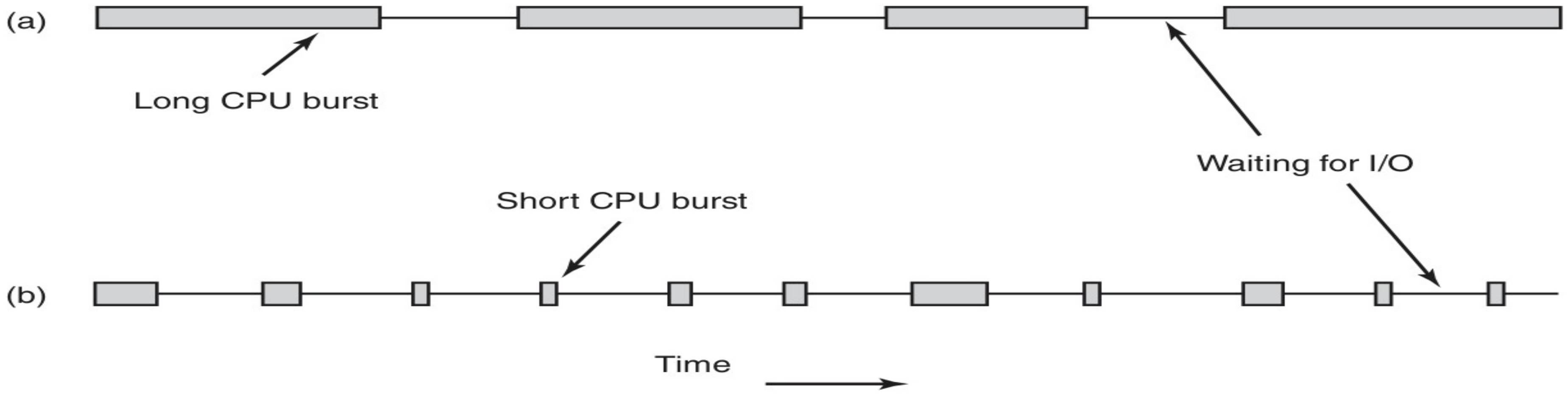
- **Scheduler** is the OS part that decides which process (or thread?) to run next from the set of ready processes (or threads?), according to a **scheduling algorithm**.
- Usually, the same questions apply to both processes and threads.
  - If threads are managed by OS, OS schedules per thread (whatever the process).
- Later, thread specific issues will be discussed.

Intro

- Old batch systems: run the next job on the tape.
- With multiprogramming, scheduler decide whether to run batch or interactive job.
  - CPU as a resource is expensive.
- With personal computers, scheduling may be not that much a deal:
  - One user, most of the time running one process.
  - Computers became faster. Usually, the user speed is the limitation.
  - Heavy CPU consumer processes are not the common case.
- With networked servers, scheduling is a big issue.
  - Choose a house keeping process or a user request?
- In smartphones and sensor network nodes, again scheduling is important:
  - Usually there is a need to optimize power consumption.
- Take care of the context switch overhead:
  - Switch to kernel mode, save process state (registers, memory map…), choose another process, reload the new process state, start the new process.
  - Cache memory flush (twice).

Process Behavior

- CPU-bound vs I/O-bound processes (w.r.t. CPU burst time).
- CPU advances faster than disks → more I/O-bound processes.
- Scheduling should favor I/O-bound processes, to keep disk busy.



When to Schedule

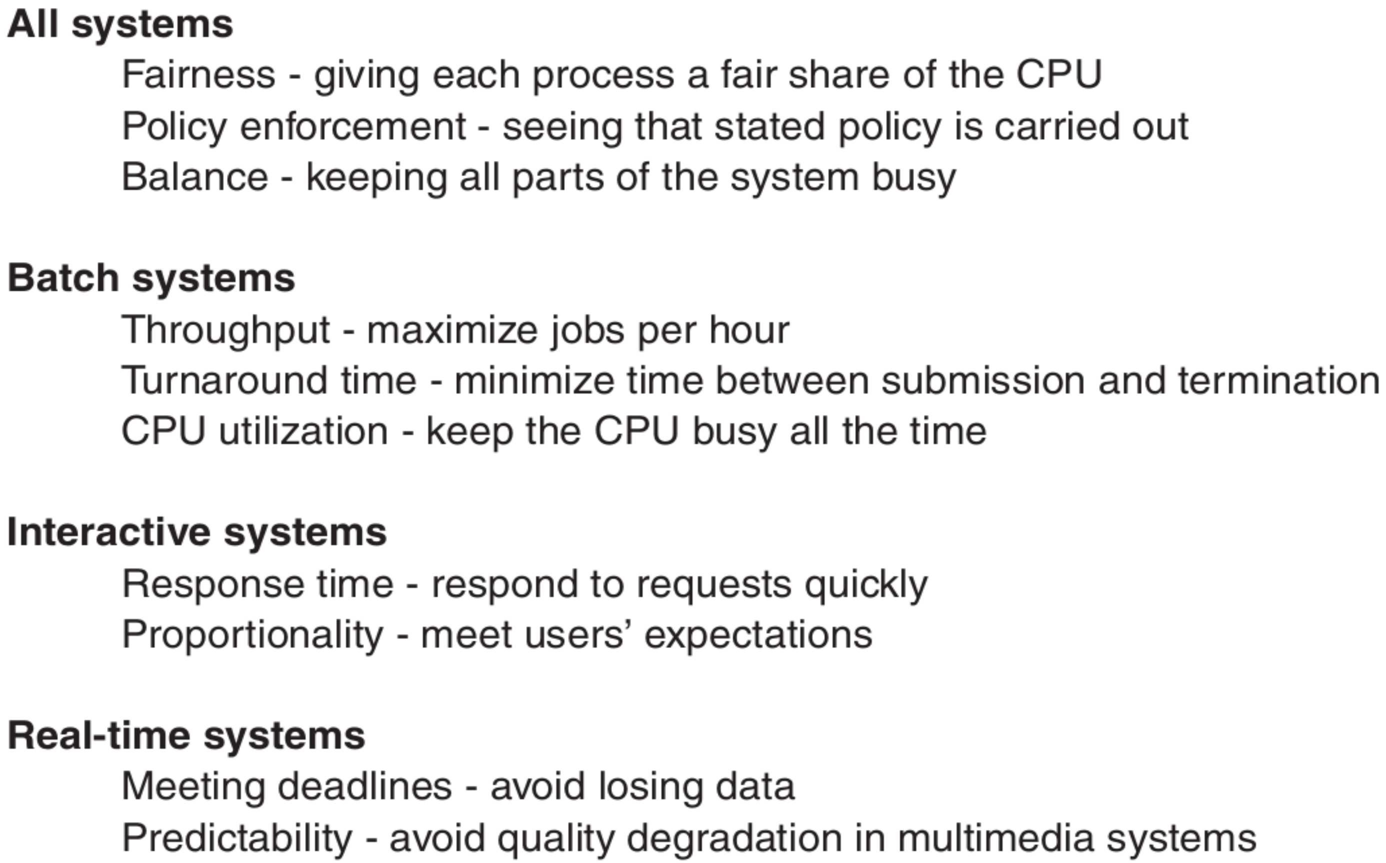
- New process creation: parent or child?
- Process exit. (if no-one is ready, system idle process)
- Process blocks (I/O, semaphore).
  - Reason of block may affect (run the other process that is blocking the important one). But, does the schedule know these info?
- At interrupts: the blocked on I/O process or the interrupted?
- H/W clock periodic interrupts (or a number of interrupts).
- **Nonpreemptive** vs **preemptive** scheduling.
  - **Preemptive** requires clock interrupts as a must.

Categories of Scheduling Algorithms

- Different environments, applications, Oss with different goals/criteria.

1) Batch:
   - Common in business world (banks, insurance com.), no users waiting.
   - Nonpreemptive or preemptive with long time periods.
     - Reduce switches → enhances performance.
   - General and applicable in other situations → good to know.
2) Interactive:
   - Preemption is necessary: prevent a process from hogging the CPU (intentionally or due to a bug).
   - Servers fall into this category.
3) Real Time:
   - Usually, preemption not needed, processes cooperate to further the task (not general purpose).

Scheduling Algorithm Goals



All systems

- Fairness - giving each process a fair share of the CPU
- Policy enforcement - seeing that stated policy is carried out
- Balance - keeping all parts of the system busy

Batch systems

- Throughput - maximize jobs per hour
- Turnaround time - minimize time between submission and termination
- CPU utilization - keep the CPU busy all the time

Interactive systems

- Response time - respond to requests quickly
- Proportionality - meet users' expectations

Real-time systems

- Meeting deadlines - avoid losing data
- Predictability - avoid quality degradation in multimedia systems

Scheduling Algorithm Goals (all systems)

- Fairness:
  - All processes get a chance (no starving)
  - Priorities: safety control should run first in all cases.
- Policy enforcement:
  - What is intended is what really happens (safety waits?!!)
- Balance:
  - A good mix of CPU-bound and I/O bound to keep all busy and avoid long waits.

Scheduling Algorithm Goals (batch systems)

- Throughput:
  - More processes per time unit is better.
- Turnaround:
  - Process finishes faster is better.
- May contradict. (ex: run shorter jobs first to enhance throughput → may lead to starvation and infinity turnaround time)
- CPU utilization:
  - Good to know to decide when new resources are needed.

Scheduling Algorithm Goals (interactive systems)

- Response time:
  - A user request should be serviced before a background process.
- Proportionality:
  - Relates to the user expectation (that me be wrong).
  - Ex: sending a large file vs breaking a connection.

Scheduling Algorithm Goals (real-time systems)

- Meeting deadlines:
  - Must (should) meet all (most) of deadlines.
- Predictability:
  - Specially multimedia (not fatal, but quality suffers)
  - Specially sound.
  - Scheduling should be predictable and regular.

Scheduling in Batch Systems

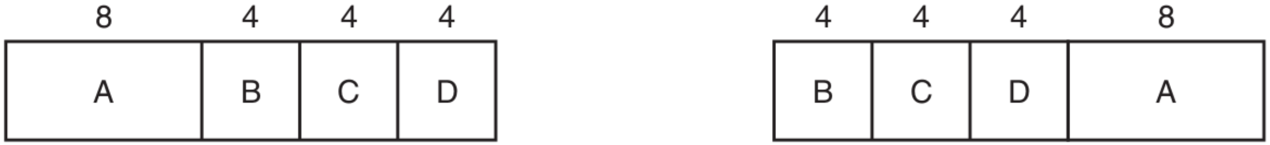
- Remember:
  - General Requirements:
    - Fairness.
    - Policy enforcement.
    - Balance.
  - Specific Requirements:
    - Throughput.
    - Turnaround.
    - CPU utilization.

First-Come, First-Served

- The simplest.
- Non-preemptive.
- A queue of ready processes.
- Easy to understand, easy to implement.
- Fair (in theory).
- What about the balance between compute-bound and I/O-bound processes?
  - Ex: 1 compute-bound process with 1 sec compute burst with a number of I/O-bound processes that need 1000 disk accesses.
  - FCFS vs preempt at 10 msec.

Shortest Job First

- Non-preemptive
- Assumes priory knowledge of running times (may be possible with recurring jobs and profiling)
- Optimize turnaround time
- Ex:
  - Turnaround times are:
    - FCFS: 8, 12, 16, 20 ---- avg: 14
    - SJF: 4, 8, 12, 20 ---- avg: 11



(a) (b)

- SJF is provably optimal:
  - Average turnaround time = (4a + 3b + 2c + d)/4 → a contributes most, it should be the shortest, and so on.
  - Only when they are available simultaneously:
  - Ex:
    - Order A, B, C, D, E → 4.6
    - Order B, C, D, E, A → 4.4

	A	B	C	D	E
Arrival	0	0	3	3	3
execution	2	4	1	1	1

Shortest Remaining Time Next

- Preemptive version of Sjf.
- When a newer shorter process arrives, its time is compared to the currently running process which is preempted if needed.
- Provides better service to new short processes.
- Ex:
  - A(2), B(1), C(1), D(1), E(1), B(3)
  - Avg turnaround = (2 + 9 + 1 + 2 + 3)/5

    = 3.4

	A	B	C	D	E
Arrival	0	0	3	3	3
execution	2	4	1	1	1