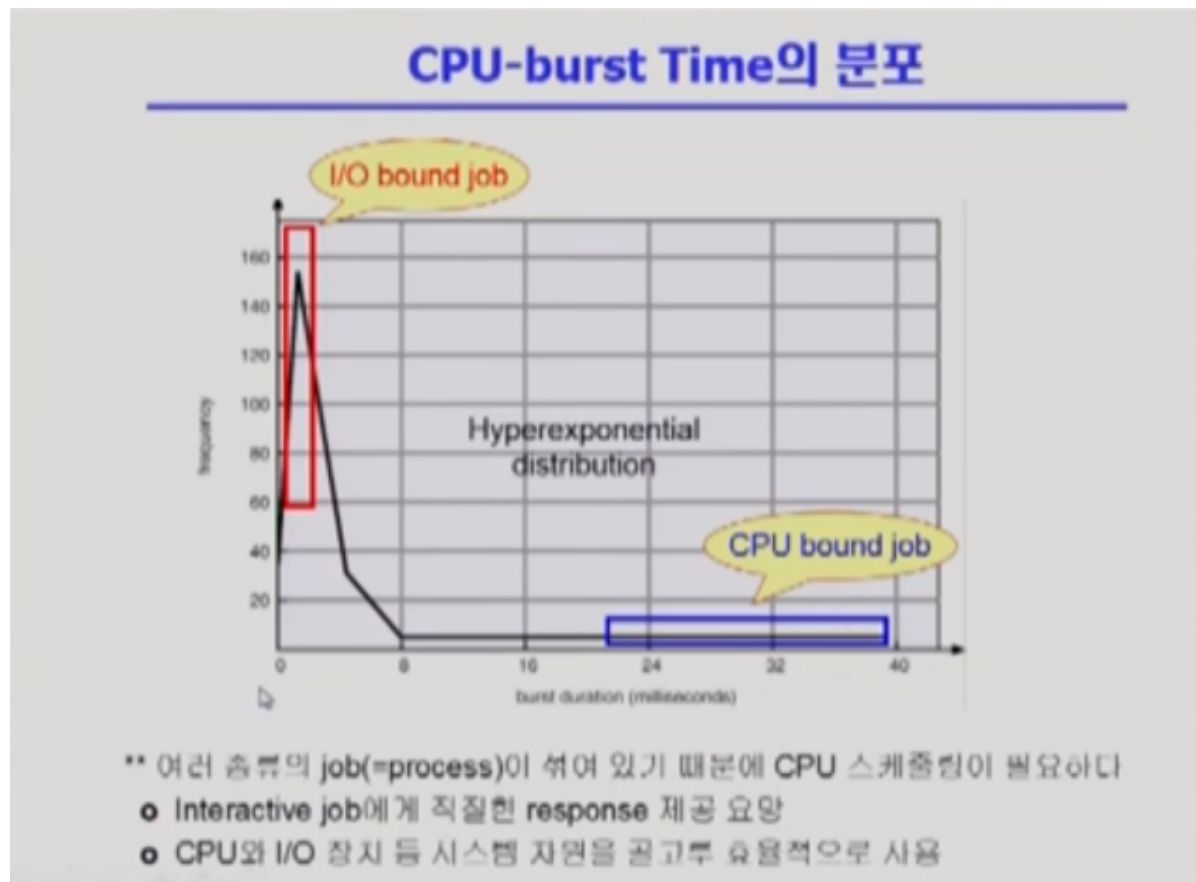


# CPU Scheduling



- cpu burst가 짧다? -> I/O가 중간에 많이 끼어들어서 짧게 나타난다.
- cpu burst가 길다? -> I/O가 거의 끼어들지 않아서 길게 나타남.
- I/O bound job : cpu burst가 짧은 프로그램들은 보통 사람과 하기 때문에 cpu를 너무 늦게 주면 응답시간이 길어져서 답답함
- cpu 스케줄링은 현재 ready queue에 들어와있는 즉 cpu를 얻고자 하는 프로세스들 중에서 어떤 프로세스한테 cpu를 줄건지 정하는 메커니즘

## 스케줄링 알고리즘 2분류

- non - preemptive(비선점형 스케줄링) : 강제로 cpu를 빼앗지 않음
- preemptive(선점형 스케줄링) : 강제로 빼앗음

## 스케줄링 알고리즘 평가 방법

## Scheduling Criteria

Performance Index (= Performance Measure, 성능 척도)

- **CPU utilization (이용률)**
  - ✓ keep the *CPU as busy as possible*
- **Throughput (처리량)**
  - ✓ # of *processes* that *complete* their execution per time unit
- **Turnaround time (소요시간, 반환시간)**
  - ✓ amount of time to *execute a particular process*
- **Waiting time (대기 시간)**
  - ✓ amount of time a process has been *waiting in the ready queue*
- **Response time (응답 시간)**
  - ✓ amount of time it takes *from when a request was submitted until the first response is produced*, **not** output  
(for time-sharing environment)

## 시스템 상에서 성능 척도

- CPU utilization
  - 전체 시간중에서 cpu가 놀지 않고 일한 시간
  - cpu를 가능한 일 많이 시켜라 -> 시스템 상에서 이득
- Throughput
  - 주어진 시간 동안 몇개의 일을 처리했는지

## 프로세스 입장에서 성능 척도

- Turnaround time
  - cpu를 쓰러 들어와서 다 쓰고 나갈때까지 걸린 시간
- Waiting time
  - 대기 시간
- Response time
  - ready queue에 들어와서 처음으로 cpu를 얻기까지 걸린 시간

## CPU 스케줄링 알고리즘 종류

# Scheduling Algorithms

- FCFS (First-Come First-Served)
- SJF (Shortest-Job-First)
- SRTF (Shortest-Remaining-Time-First)
- Priority Scheduling
- RR (Round Robin)
- Multilevel Queue
- Multilevel Feedback Queue

## FCFS

- 먼저 온 순서대로 처리
- 비선점형
- 별로 효율적이지 않음

### FCFS (First-Come First-Served)

→ Example:

Process	Burst Time
$P_1$	24
$P_2$	3
$P_3$	3

→ 프로세스의 도착 순서  $P_1, P_2, P_3$   
스케줄 순서를 Gantt Chart로 나타내면 다음과 같다



→ Waiting time for  $P_1 = 0$ ;  $P_2 = 24$ ;  $P_3 = 27$   
→ Average waiting time:  $(0 + 24 + 27)/3 = 17$

## SJF

- CPU를 사용하고자 하는 시간이 가장 짧은 프로그램에서 CPU를 줌
- 평균 대기 시간을 가장 줄여줌
- 두가지 방식을 나누어서 생각 가능
  - 비선점형
    - 일단 기다리는 프로세스 중에서 가장 짧은애한테 cpu주면 더 짧은 프로세스가 도착하더라도 앞에 애가 끝날때까지는 못 넘겨준다
  - 선점형
    - cpu를 줬다 하더라도 더 짧은 애가 도착하면 그 프로세스한테 cpu를 준다
    - SRTF라고도 부름
    - 평균 대기 시간 더 짧음

### SJF (Shortest-Job-First)

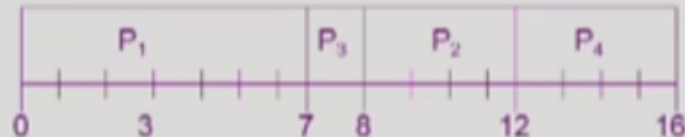
- ➔ 각 프로세스의 다음번 *CPU burst time*을 가지고 스케줄링에 활용
- ➔ *CPU burst time*이 가장 짧은 프로세스를 제일 먼저 스케줄
- ➔ Two schemes:
  - ✓ **Nonpreemptive**
    - 일단 CPU를 잡으면 이번 CPU burst가 완료될 때까지 CPU를 선점(preemption) 당하지 않음
  - ✓ **Preemptive**
    - 현재 수행중인 프로세스의 남은 burst time보다 더 짧은 CPU burst time을 가지는 새로운 프로세스가 도착하면 CPU를 빼앗겨
    - 이 방법을 Shortest-Remaining-Time-First (**SRTF**)이라고도 부른다
- ➔ SJF is optimal
  - ✓ 주어진 프로세스들에 대해 *minimum average waiting time*을 보장

- 예시

## Example of Non-Preemptive SJF

Process	Arrival Time	Burst Time
$P_1$	0.0	7
$P_2$	2.0	4
$P_3$	4.0	1
$P_4$	5.0	4

→ SJF (non-preemptive)

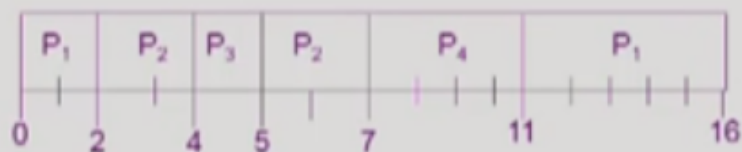


→ Average waiting time =  $(0 + 6 + 3 + 7)/4 = 4$

## Example of Preemptive SJF

Process	Arrival Time	Burst Time
$P_1$	0.0	7
$P_2$	2.0	4
$P_3$	4.0	1
$P_4$	5.0	4

→ SJF (preemptive)



→ Average waiting time =  $(9 + 1 + 0 + 2)/4 = 3$

- 두가지 문제점 존재
  - Starvation : 극단적으로 cpu 사용시간이 짧은거를 선호 함 그래서 cpu 사용시간이 긴 프로세스들은 사용 못 할 수 있음
  - cpu 사용시간을 미리 알 수 없다. 대신 추정은 가능

## 다음 CPU Burst Time의 예측

- 다음번 **CPU burst time**을 어떻게 알 수 있는가?  
(input data, branch, user ...)
- 추정(**estimate**)만이 가능하다
- 과거의 CPU burst time을 이용해서 추정  
(exponential averaging)

1.  $t_n$  = actual length of  $n^{th}$  CPU burst
2.  $\tau_{n+1}$  = predicted value for the next CPU burst
3.  $\alpha, 0 \leq \alpha \leq 1$
4. Define:  $\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$

- 과거의 CPU burst time을 이용해서 추정
- $t$ : 실제 cpu 사용 시간
- $\tau$ : cpu 사용을 예측한 시간

## Exponential Averaging

- $\alpha = 0$ 
  - ✓  $\tau_{n+1} = \tau_n$
  - ✓ Recent history does not count.
- $\alpha = 1$ 
  - ✓  $\tau_{n+1} = t_n$
  - ✓ Only the actual last CPU burst counts.
- 식을 풀면 다음과 같다
$$\tau_{n+1} = \alpha t_n + (1-\alpha) \alpha t_{n-1} + \dots$$
$$+ (1-\alpha)^j \alpha t_{n-j} + \dots$$
$$+ (1-\alpha)^{n+1} \tau_0$$
- $\alpha$ 와  $(1-\alpha)$ 가 둘다 1 이하이므로 후속 term은 선행 term보다 작은 기승지 값의 기승지

## Priority Scheduling

- 우선순위가 높은 프로세스에게 cpu를 줌
- 우선순위가 주어짐(작은 수 --> 더 우선 순위)
- 두가지 방법 가능
  - 비선점형
  - 선점형
- 문제점은 똑같이 starvation(기아현상)이 있다
  - 해결하기 위해 Aging 기법을 사용
  - 아무리 우선순위가 높아도 시간이 지남에 따라 우선순위를 줄여줌

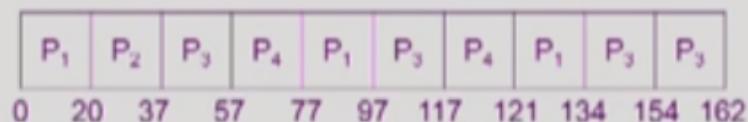
## Round Robin(RR)

- 현대적인 CPU스케줄링은 Round Robin에 기반함
- 선점형
- 동일한 할당 시간을 세팅해서 주고 시간이 지나면 cpu 뺏어감
- 장점
  - 응답시간이 빨라짐
- 할당시간을 아주 크게 잡으면 FCFS와 같아진다
- 할당시간을 아주 작게 잡으면 context switch 가 너무 자주 발생하여 오버헤드가 커진다.
- 고로 할당시간을 10 ~100 사이로 적당하게 잡는것이 중요하다
- 예시
- 

### Example: RR with Time Quantum = 20

Process	Burst Time
$P_1$	53
$P_2$	17
$P_3$	68
$P_4$	24

→ The Gantt chart is:



→ 일반적으로 SJF보다 average turnaround time이 길지만 response time은 더 짧다.