4️⃣ CPU 스케줄링

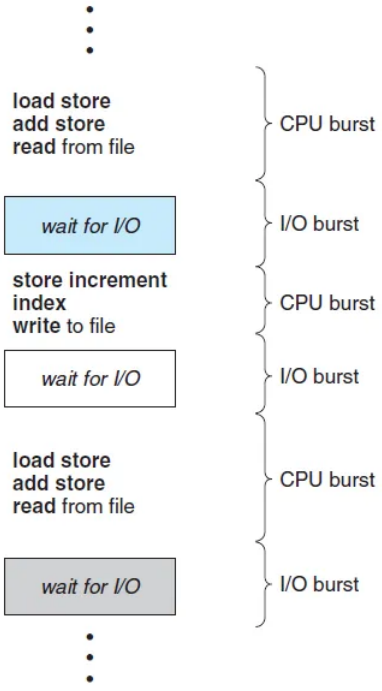
태그 2주차 CSS

💡 CPU 스케줄링과 디스패처가 필요한 이유와 개념

프로세스

프로세스 일생

- CPU burst, I/O burst의 연속



CPU or I/O bound job

- CPU bound job
  - 계산 위주
  - few very long cpu bursts
- I/O bound job
  - many short cpu bursts

> job들이 균일하지 않고 섞여 있기 때문에 CPU를 **누구**(프로세스)한테, **얼마나** 줄 것인가가 대단히 중요하다!

CPU Scheduler & Dispatcher

CPU Scheduler

- Ready 상태의 프로세스 중에서 CPU를 줄 프로세스를 고른다

Dispatcher

- 그 결정된 프로세스에게 CPU를 실제로 넘기는 역할을 하는 커널 코드
- Context Switch (save & load)

CPU 스케줄링이 필요한 경우

1. Running → Blocked (예: I/O 요청하는 시스템 콜)
2. Running → Ready (예: 할당 시간 만료로 timer interrupt)
3. Blocked → Ready (예: I/O 완료 후 인터럽트)
4. Terminate

→ 1, 4는 nonpreemptive (자진 반납)

→ 나머지는 preemptive (강제로 빼앗음)

스케줄링 성능 척도

시스템 관점

- CPU Utilization: 전체 시간에서 CPU가 놀지 않고 일한 시간, 높을수록 좋음
- Throughput(처리량): 단위 시간 당 완료한 프로세스, 높을수록 좋음

사용자 관점

- Time, 짧을수록 좋음
  - Response Time(응답 시간): CPU를 쓰러 들어와서 최초로 CPU를 얻을 때까지 걸린 시간
  - Turnaround Time(반환 시간): CPU Burst 전체 시간, CPU 기다린 시간 + 사용한 시간
  - Waiting Time(대기 시간): CPU를 쓰러 들어와서 대기열에서 기다린 시간의 총합

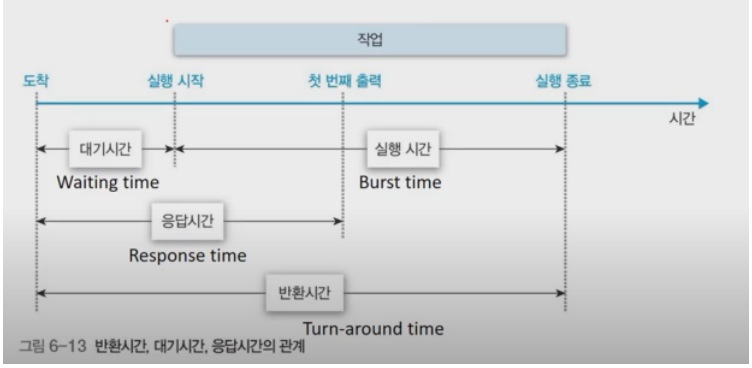


그림 6-13 반환시간, 대기시간, 응답시간의 관계

Scheduling Algorithms

FCFS (First-Come First-Served)

1. 아이디어

- 들어온 순서대로 CPU 할당
- 예: 은행, 화장실

2. 특징

- 비선점형
- 앞에 긴 친구가 들어오면 평균 대기 시간이 길어짐
  → Convoy Effect(호위 효과) 현상에 의해 이용률이 낮아질 수 있다

SJF (Shortest Job First)

1. 아이디어

- 처리 시간이 짧은 프로세스부터 CPU 할당

2. 특징

- FCFS에 비해 평균 대기 시간과 평균 응답 시간이 개선됨

2-1. Two schemes

- Nonpreemptive: 일단 cpu 잡으면 끝냄
- Preemptive: 중간에 더 짧은 애가 들어오면 빼앗음 → Shortest-Remaining-Time-First(SRTF)
  - Optimal: 대기 시간이 젤 짧다

2-2. 문제점

- Starvation: 처리 시간 긴 프로세스는 절대 할당 안 됨 (배고파 죽어...)
- 다음 Cpu Burst Time을 알 수가 없다 → 과거 Cpu Burst Time으로 추정만 가능, 수식은 생략!

Priority Scheduling

1. 아이디어

- 우선순위가 높은 프로세스를 먼저 cpu 할당
- 예: SJF

2. 특징

- 이것도 Preemptive, Nonpreemptive 둘 다 가능

2-1. 문제점

- Starvation

2-2. 해결책

- aging: 시간 지나면 우선순위를 높여주자

Round Robin (RR)

1. 아이디어

- 동일한 크기의 시간(q, quantum time)을 할당하는 전형적인 Preemptive 스케줄링

2. 특징

- q가 너무 크면 FCFS
- q가 너무 작으면 Context Switch 오버헤드가 커진다
- 일반적으로 SJF보다 average turnaround time이 길지만 response time이 더 짧다

Multilevel Queue

1. 아이디어

- 여러 줄을 선다 (CPU는 하나)
- 각 큐에 다른 전략을 적용한다
- foreground: interactive
- background: batch - no human interaction

Multilevel Feedback Queue

1. 아이디어

- 줄 간 이동 가능
- 0: 할당 시간 짧게, 1: 할당 시간 길게, 2: FCFS
- 0에서 시작해서 할당 시간 안에 안 끝나면 아래 큐로 이동

2. 특징

- Starvation - aging으로 해결

Multiple-Processor Scheduling

- Homogeneous processor: 큐에 한 줄로 세워 놓고 각 프로세서가 알아서 꺼내간다
- Load sharing
- Symmetric Multiprocessing
- Asymmetric Multiprocessing

Real-Time Scheduling

- Hard real-time systems: 주어진 시간에 꼭 끝내야 함
- Soft real-time computing: 예) 동영상 스트리밍, 데드라인이 있지만 어겼을 때 엄청 큰 일이 나지는 않는 일

Thread Scheduling

- Local Scheduling: User level thread의 경우 사용자 수준의 thread library에 의해 어떤 thread를 스케줄할 지 결정
- Global Scheduling: Kernel level thread의 경우 일반 프로세스와 마찬가지로 커널의 단기 스케줄러가 어떤 thread를 스케줄할 지 결정

스케줄링 알고리즘 평가

- Queueing models
  - 이론적인 방법, 구식
- Implementation & Measurement
- Simultation