CPU 스케줄링

프로세스 관련 시스템콜

fork()

exec()

wait()

exit()

CPU 스케줄링

- CPU 스케줄링은 준비 상태에 있는 프로스세스 혹은 스레드들 중 하나를 선택하여 CPU를 할당하는 과정
- 오늘날의 운영체제에서의 CPU 스케줄링은 프로세스가 아닌 스레드를 대상으로 함

CPU burst & I/O burst

프로그램에서 cpu의 연산 작업과 입출력 작업이 섞여있다.

CPU burst

- CPU가 코드를 집중적으로 실행하는 상황
- CPU burst 시간

I/O burst

- I/O 장치에 의해 입출력이 이루어지는 상황
- I/O burst 시간

⇒ I/O burst 시간동안 CPU의 노는 시간을 줄이기 위해 CPU 스케줄링 등장

CPU 스케줄링

CPU 스케줄러

디스패처

- 선택된 프로세스에게 CPU의 제어권을 넘김

스케줄링이 필요한 경우

1. 런닝 → 블럭(I/O 요청)
2. 런닝 → 레디 cpu 할당 시간 끝났을때 cpu빼앗음
3. 블럭 → 레디 (I/O 완료 후 인터럽트)
4. Terminate

스케줄링 기준

CPU utilization (이용률)

- 컴퓨터 전체 가동 시간에서 CPU 사용 시간의 비율
- CPU 이용률이 많게 유지 → CPU가 노는 시간이 적도록

Throughput(처리량)

- 단위시간당 프로세를 몇개 완료 시켰나?

Turnaround time(소요시간, 반환시간)

- cpu 기다린 시간 + cpu 사용한 시간

Waiting time(대기 시간)

- cpu를 쓰러와서 cpu를 기다리는 시간의 총 합

Response time(응답 시간)

- 프로세스가 cpu를 쓰기위해 기다리면서 최초로 cpu를 얻는데 걸리는 시간
- 대부분의 cpu가 preemptive한 상황이기에(빼앗기는 상황이기에)

CPU 스케줄링 알고리즘

FCFS(First Come First Served) - nonpreemptive

- 먼저 온 순서대로 처리함
- 은행에서 번호표를 뽑는 고객과 유사

- 문제점
  - convoy effect
  - 앞에 긴 프로세스가 선점하면 뒤에 짧은 프로세스가 오래 대기하여 시스템 전체가 느려지게 된다.
  - 즉 앞의 프로세스 실행시간에 따라 평균 대기시간의 변화가 생김

SJF(Shortest Job First) - nonpreemptive

- 실행시간이 짧은 것 부터 먼저 실행시킴
- 문제점
  - 짧은 프로세스에게 계속해서 우선권을 주기에 기아 발생 가능성

    → 화장실 10분이용고객이 1분 이용고객에게 계속 양보했을 경우 영원히 화장실 못 사용함

SRTF(Shortest Remaining Time First) - preemptive

- SJF의 preemptive 스케줄링
- 실행 중인 프로세스의 남은 실행 시간보다 더 짧은 실행 시간을 가진 스레드가 도착하면 현재 실행시키고 있는 프로세스를 중단하고 해당 프로세스를 실행시킨다.
- 매 시간마다 가장 빠른 프로세스를 실행 시키기에 대기시간이 가장 짧다.
- 문제점
  - 기아 발생 가능성

설명 : SJF, SRTF는 대기 시간이 짧다는 장점이 있다.

하지만 기아 발생 가능성 뿐만 아닌 프로세스의 예상 실행 시간을 알고있어야 함

그래서 이것을 어떻게 예측 할건데? 과거의 cpu burst time을 이용해서 예측

그렇다면 기아 발생 문제는 어떻게 해결할 것인가? ⇒ priority scheduling aging

Priority Scheduling

- 우선순위에 따라 프로세스를 실행시킴
- SJF는 일종의 Priority Scheduling
- 이 역시 기아 발생 가능성

- 해결책은 Aging
    - 큐의 대기 시간에 비례해서 우선순위를 높이는 방법을 통해 기아를 해결

RR(Round-Robin)

- 프로세스들에게 공평한 기회를 주기 위해 일정한 간격으로 번갈아 실행
- 타임 슬라이스를 기준으로 타임 슬라이스가 지나면 강제로 중단 시켜 그 다음 프로세스를 선택
- Response Time(응답시간)이 짧다. 즉 최초로 CPU를 얻기 위한 시간이 짧다. - 공평함
- 긴시간과 짧은 시간이 공존하는 상황에서 사용해야 장점이 있다.

Multi-level Queue

- 우선순위 레벨을 가진 큐로 분리하고 프로세스를 우선순위 별로 큐에 삽입함
- 신분별로 나누고 높은 신분에 속한 프로세스를 먼저 실행시키는 것을 의미함
- 다른 큐로 이동 불가능하고 고정된 우선순위를 갖는다.
- 기아 발생 가능성이 존재
    - 지속적으로 높은 신분의 프로세스가 큐에 도착하면
    - 낮은 신분의 프로세스는 언제 실행될지 모른다.

Multi-level Feedback Queue

- MLQ와 달리 다른 큐로 이동가능하며 지속적으로 우선순위가 변한다.