8강

스케줄링이란 무엇인가

- CPU 스케줄링

  프로세스들을 실행을 시켜야 하는데, 어떤 순서로 실행시켜야 하는지를 다루는 것이다.

  스케줄링은 매우 중요하다.

  스케줄링에 따라서 전체 시스템의 성능이 좌우된다.

  스케줄링이란 프로세스와 CPU를 맵핑시키는 일이다.

  어떤 프로세스를 CPU에 맵핑시킬 것인지를 결정하는 문제인 것이다.

  어떤 P_i가 CPU를 사용할 것인가에 대한 문제이다.

  multi core에서 어찌 할지는 뒷부분에서 다시 설명한다.

  - 어떻게 프로세스에게 CPU의 사용을 분배할 것인가
    - Multiprogramming 과 시분할에 기반함

      → 이것은 multiprogram과 관련이 있는 것이다.
    - 메모리 내 실행 준비된 프로세스들 가운데 하나를 선택하여 CPU를 할당함
      - 질문: 멀티코어는 어떻게 할까?

        → 그리고 멀티 코어가 되면 어떤 스케줄링이 필요한가?

- CPU 스케줄링의 목표
  - CPU 를 최대로 활용하는 것: idle 최소화
    - 질문: 커널이 사용하는 CPU 시간은?

      단, 이것은 dos과는 다르다.

      서버에 미친 듯이 메일을 보내는 것과는 다르다. 이것은 스케줄링 목표와는 다르다.

스케줄링의 종류

- CPU scheduling 의 결정은 다음 상태 변화에서 이루어짐.
  - “Running” 에서 “Waiting” 상태로 (I/O를 하는 경우)
  - “Running” 에서 “Ready” 상태로 (timer 인터럽트)

    → save를 하고 Next를 찾는 것

- 비선점형 스케줄링 (Non-preemptive Scheduling)
  - 프로세스가 I/O를 하는 상황에서만 수행되는 스케줄링
  - Multiprogramming의 기본 스케줄링 – OS가 강제로 CPU 사용을 중단시키지 않음

    →multiprogramming의 기본 스케줄링은 I/O를 할때만 cpu가 중단됨

- 선점형 스케줄링 (Preemptive Scheduling)
  - 타임퀀텀을 소진한 상황에서 스케줄링
  - OS가 현재 CPU를 사용하고 있는 프로세스의 수행을 강제적으로 정지할 수 있음

    → 강제로 Text를 찾게 하는 것이다.

    → Time sharing에서 쓰는 방식이다.

Scheduling Criteria(기준)

- CPU 활용률 (CPU utilization)
  - 전체 시스템 시간 중 CPU가 작업을 처리하는 시간의 비율

- 처리량 (Throughput)
  - CPU가 단위 시간 당 처리하는 프로세스의 개수

    → 오래걸리는 프로세스가 들어가면 처리량이 줄어든다.

    → 짧게 걸리는 프로세스를 넣으면 처리량이 늘어난다.

- 응답 시간 (Response time)

- 프로세스가 입출력을 시작해서 첫 결과가 나오는데 까지 걸리는 시간
    - 질문: 큐로 설명해 보기

→ 엔터를 치고 언제 결과가 화면에 나오는가? 그 시간.

→ 엔터를 치고 반응이 나오는데의 시간이 3초가 넘으면 사람이 다시 엔터를 치더라

→ waiting rhks spu처리시간을 합친 시간

Q. 그렇다면 응답시간이 대기시간을 포함하고 있는 개념인건가요??

A. yes

특정 프로세스 실행 중에 I/O가 여러 번이 수행될 수 있는데 응답 시간은 그때마다의 결과가 나오는데까지 걸리는 시간을 합한 개념인가요?

응답시간은 단발에 대한 request의 결과를 의미한다. 이것은 각각의 질문에대한 응답 시간을 말한 것이다.

- 대기 시간 (Waiting time)
    - 프로세스가 Ready Queue 내에서 대기하는 시간의 총합

      순수하게 큐에서 기다리는 시간
        - 프로세스가 만들어져서 cpu i//o 다 하고 사라지는 시간을 다 포함한 것이다.

- Turnaround time
    - 프로세스가 시작해서 끝날 때까지 걸리는 시간

      → 여기에는 대기사간, 응답시간이 전부 포함된다.

      대기와 응답시간을 포함하는 것이다.

      이전에 대기시간이 응답시간에 포함된다고 하셨는데 Turnaround time이 대기시간 + 응답시간의 합이라면 계산이 중복되는 것이 아닌지 궁금합니다.

      이건 아니다.

      Turnaround time은 모 든 시간을 다 합친다는 의미이다.

      더 정확히 말하자면 모든 응답시간의 합을 의미한다.

Q. 교수님 응답시간과 대기시간의 차이가 뭔가요??

A. 응답시간은 하나의 CPU

그렇게 되면 처음 프로세스가 CPU를 할당받기 전 대기시간은 응답시간에 포함되게 되는데, 그러면 안되지않나요?

맞다.

그래서 엄밀하게 말하면 어떻게 가정하느냐에 따라 달라진다.

대기시간이 포함되는지 안되는지의 여부는 그 가정을 봐야한다.

프로세스가 만들어져서 끝날 때까지의 모든 시간으 의미하는 것이다.

Scheduler – design

- 이상적인 스케줄러
  - 최대의 CPU 사용률
  - 최대의 처리량
  - 최소의 응답시간
  - 최소의 대기시간
- 모든 조건을 만족 시키는 스케줄러를 만드는 것은 현실적으로 불가능

  → 사실 이것은 현실적으로 불가능함

- 시스템의 용도에 따른 요구사항이 달라짐.
  - 슈퍼 컴퓨터 – CPU 사용률
  - 개인용 컴퓨터 – 응답시간
  - 워크 스테이션 – 처리량

  → 자율주행 자동차는 뭐가 제일 중요한가?

  → 응답시간이 제일 중요할 것 같습니다.

  → 운전의 실시간성을 고려하여 안전하게 설계해야 할 듯 합니다.

  날씨 예측 슈퍼컴은 여기서는 TIme sharing을 쓰지 않는다.

Q. 1. 응답시간은 프로세스가 Waiting Queue에서 대기하는 시간을 의미하는 것인가요?

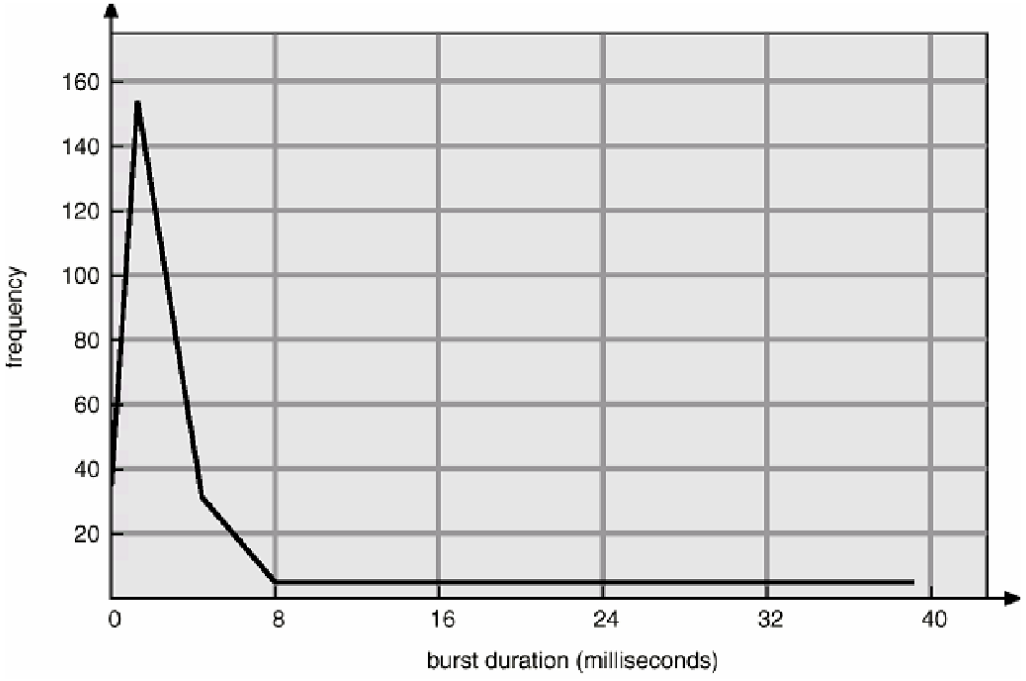
Waiting Queue에서 대기하는 시간 + 화면에 결과가 나오는 시간을 의미한다.

Q. 2. 최소의 대기시간을 가진다는 것이 어떤 의미를 가지나요?

큐에서의 대기시간을 줄인다는 의미이다.

Histogram of CPU-burst times

- 서로 다른 프로세스, 시스템에도 불구하고 대체적으로 아래와 같은 경향을 지님



burst는 i/o와 i/o사이의 시간을 의미한다.

burst의 많은 숫자가 8m/s이하이다.

한 프로세스내의 brust를 모아놓은 그래프이다.

교수님 질문 : 이 그래프가 가지는 의미가 무엇인가?

답변 : 흠... Time Sharing을 하면 오버해드가 발생하게 된다.

state를 저장하고 reload하는 과정이 필요한 것이다.

이 그래프는 타임쉐어링의 타임 퀀텀을 어떻게 정하는지의 여부를 정하는데 도움을 준다.

만일 8이하로 타임 퀀텀을 정하면 한 퀀텀안에 프로세스가 안끝나기 때문에

프로세스 수행 사이클의 구성

- CPU-I/O Burst Cycle

  CPU-I/O-CPU-I/O-CPU-I/O-...

  - CPU Burst : CPU로 연산을 수행하는 시간.
  - I/O Burst : I/O 처리를 위해 기다리는 시간.
  - 일반적인 프로세스는 두 burst를 번갈아 가며 수행됨
- 프로세스 분류에 따른 CPU Burst의 특징
  - CPU-bound 프로세스 : 긴 CPU burst

    → CPU 계산만 많이 한다.
  - I/O-bound 프로세스 : 짧은 CPU burst

    → I/O만 많이 한다.
  - 어떤 종류의 프로세스가 많은 지에 따라 스케줄링 기법의 효율성이 달라짐

process의 cpu burst time이 뭔지 다시 한번 설명해주실 수 있을까요?

ㅁ. CPU-I/O-CPU-I/O-CPU-I/O-...의 구성인데 그때 cpu가 사용되는 그 시간을 지칭한다..

3. Scheduling Algorithms

구체적으로 뭐가 있는지를 살펴보자

- First-Come, First-Served Scheduling
- Shortest-Job-First Scheduling
- Priority Scheduling
- Round-Robin Scheduling
- Multilevel Queue Scheduling

- Multilevel Feedback Queue Scheduling

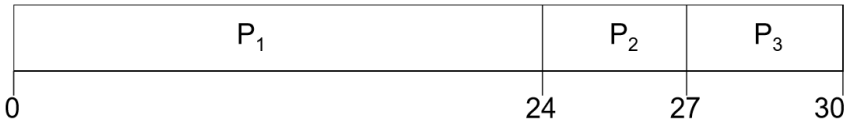
3.1 FCFS Scheduling

First Come First Out

- Ready 큐에 있는 순서대로 CPU를 할당한다.
  - FIFO 큐를 사용하여 간단하게 구현 가능.
- 예제

Process	Burst Time
P_1	24
P_2	3
P_3	3

- (1) P1, P2, P3 순서로 요청하였을 때



대기 시간 : P1 = 0, P2 = 24, P3= 27

평균 대기 시간 : (0 + 24 + 27) / 3 = 17

FCFS Scheduling – 순서바꿈

- (2) P2, P3, P1 순서로 요청하였을 때
  - 대기 시간 : P1 = 6, P2 = 0, P3= 3
  - 평균 대기 시간 : (6 + 0 + 3)/3 = 3
  - (1)의 경우 보다 짧은 대기 시간을 가짐.

- 작업의 수행 순서에 따라 대기 시간이 변할 수 있음.

3.2 Shortest Job First Scheduling

- 다음 CPU burst 시간이 가장 짧은 프로세스에 CPU를 먼저 할당한다
  - 최소의 평균 대기 시간을 제공.
- 비선점형 방식 자발적 양보 시점에 다음 프로세스를 선택하여 수행함
- 선점형 방식
  - 타임퀀텀마다 CPU burst 시간이 가장 짧은 프로세스를 선택함
    - CPU burst 시간을 미리 알 수 없기 때문에 approximation: Shortest Remaining Time First Scheduling (SRTF)

      → 이것을 알 수 가 없기 때문에 근사치를 구해야 함

Q. Shortest job first scheduling에서 비선점형, 선점형 방식 모두 CPU burst 시간을 모르기때문에 추정해서 수행하나요?

A. 이것은 가장 짧은 burst time을 안다는 전제 하에서 가능한 것이다.

그럼에도 불구하고 이론적인 upper bound이기에 설명을 하는 것이다.

Q. SJF 설명하실 때 일반적으로 선점형의 오버헤드가 작다고 하셨는데, 왜 그런것인가요?

A. 선점형이 오버해드가 더 작다.

어떤 의미에서의 오버해드가 작다는 것인가?

큐에 올라온 순서에 대해서 평균대기 시간이

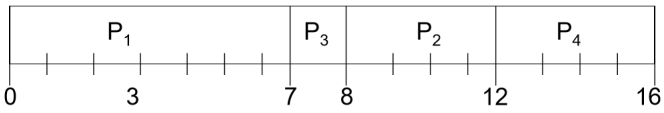
비 선점형은 아이오를 할때만 스케줄링을 하는데, 선점형은 타임 퀀텀이 지나면 그때마다 스케줄링을 하기 때문이다.

Shortest Job First Scheduling 예

- SJF의 시나리오

Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- 비선점형 SJF스케줄링 결과



평균 대기 시간 = (0 + 6 + 3 + 7)/4 = 4

Q. 13쪽도 왜 0 + 6 + 3 + 7 인지 이해가 가질 않습니다

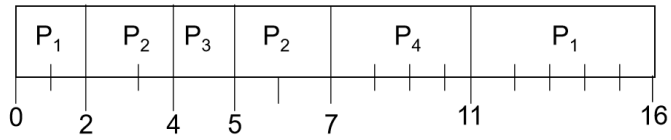
A.

Shortest Job First Scheduling -선점형

많은 경우에 선점형으로 사용하게 된다.

Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- 선점형 SJF 스케줄링 결과



- 평균 대기 시간 = (9 + 1 + 0 +2)/4 = 3

Q. 14쪽 9 + 1 + 0 + 2가 어떻게 나오는지 궁금합니다

A.

프로세스 Priority에 따라 CPU를 할당한다

- 비선점형 방식
  - 현재 수행중인 프로세스가 자발적 양보를 할 때, 우선순위에 따라 다음에 수행될 프로세스를 선택한다

- 선점형 방식
  - 타임퀀텀마다 현재 실행되는 프로세스보다 높은 priority를 가지고 있는 프로세스를 찾아 수행한다

  유저레벨 프로세스보다는 커널 레벨 프로세스에 더 높은 priority를 제공한다.

  → 이것은 수행전에 미리 정해진 것이다.

- 문제점 – 기아 상태 (Starvation)
  - 낮은 priority를 가진 프로세스는 전혀 수행되지 않을 수 있다

- 해결 방법 – Aging 기법
  - 기다리는 시간에 따라 프로세스 priority를 증가시켜주는 방법

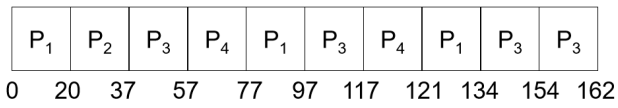
    → 이것은 동적으로 이렇게 정해지는 것이다.

3.4 Round Robin Scheduling

- CPU를 시간 단위(time quantum)로 할당한다.
  - 선점형 스케줄링 방식.
  - 보통 time quantum은 1 – 100 ms
  - Time quantum을 수행을 한 프로세스는 Ready 큐의 끝으로 들어가 다시 할당을 기다림.

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

Time quantum = 20



Q. Round Robin 스케쥴링이 "FCFS에 선점형을 적용한것"과 차이가 있나요?

A. 순서에 영향을 크게 받지 않도록 만들어주는 것이 Round Robin Scheduling이다.

Round Robin Scheduling은 순서를 다지지 않는다.

Round Robin Scheduling은 처음에 수행하는 것이 순서에 영향이 이을지는 몰라도 그 이후 부터는 그냥 실행한다.

Round Robin은 순서가 별로 의미가 없다.

3.4 Round Robin Scheduling분석

- 장점: response time이 짧다
  - Interactive 프로세스에 필요함
- 분석
  - Ready 큐 내의 프로세스 n개, Time quantum q
  - 각각의 프로세스가 할당 받는 시간
    - 1/n 만큼의 CPU 시간을 q로 쪼개어 할당 받음.
  - 각 프로세스의 다음 time quantum이 돌아오기까지의 대기시간
    - 최대 (n-1) x q

- 성능
  - q가 클 경우 : FCFS
  - q가 작을 경우 : 문맥전환(Context Switching)에 필요한 시간보다 짧다면 효율이 매우 떨어짐

  Time quantum q가 커지면 대기시간이 줄어든다. 그래서 적절한 q를 정하는 것이 매우 중요하다.

  Q. 1/n 만큼의 CPU 시간을 q로 쪼개어 할당 받는다는 것이 어떤 의미인가요?

  A. n개의 프로세스가 있다면 시간을 타임 퀀텀으로 나누어서 n개의 프로세스가 나누어 가진다는 의미이다.

  Q. 조금전에 FCFS 방식에서도 선점형방식이 overhead가 적다고 하셨는데, FCFS방식에서는 선점형과 비선점형 방식 둘다 선착순으로 하나의 process가 끝날때까지 계속해서 CPU를 사용한다는 점에서 overhead의 차이가 없지않나요?

  A, tunaround time은 같다.

대기시간과 response 타임이 달라지는 것이다.

3.5 Multilevel Queue Scheduling

- Ready 큐를 여러 개 만들어 각각에 대해 다른 우선순위와 스케줄링 알고리즘을 사용하는 기법
- 예시:
  - Foreground 큐
    - Interactive한 동작이 필요한 프로세스를 위한 큐

      사람들이 느끼기에 바로바로 반응이 나와야 하는 프로세스들, 카톡같은 것들

    - 큐 동작: Round Robin 기법 사용.
  - Background 큐
    - CPU 연산 작업을 주로 수행하는 프로세스를 위한 큐

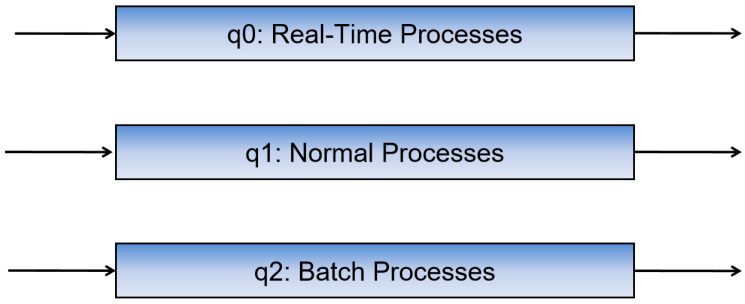
      딥러닝 프로세스와 같은 것들

    - 큐 동작: FCFS 기법 사용
  - 문제점
    - 상위 큐에 프로세스가 계속 있으면 하위 큐에 기아 현상 발생

Q. Foreground 큐와 Background 큐 사이의 우선순위는 Foreground가 무조건 우선되는 것인가요? 아니면 프로세스마다 경우가 다른 것인가요?

A. yes. 포크라운드에서 카톡 같은 것들이 수행된다.

3.5 Multilevel Queue Scheduling 예시



반도체 제조 공장이라고 생각하라. 반도체 공장도 굉장히 실시간적이다

Foreground 큐와 Background 큐 사이의 우선순위는 Foreground가 무조건 우선되는 것인가요? 아니면 프로세스마다 경우가 다른 것인가요?

A. q_1, q_0가 아무것도 없으면 그냥 q_2를 실행시킴

19 페이지에서 q2 큐가 FCFS 기법을 이용한다고 가정했을때, q2에 들어있던 프로세스 p가 실행되는 동안은 q0 q1이 비록 우선순위가 높더라도 p가 끝날때까진 기다리고 있는건가요??

yes

micro kernel은 사용자 프로세스와 동등한 취급을 한다고 하는데, 그렇다면 cpu scheduling시 다른 사용자 프로세스와 같이 우선순위 경쟁을 하는지 궁금합니다

no 커널이 더 높다.