CHAPTER

5

처리기 스케줄링

스케줄링은 자원에 대한 경쟁이 있을 때 경쟁자들 중 하나를 선택하는 과정으로 컴퓨터 시스템 여러 곳에서 일어난다. 이중 처리기 스케줄링은 준비 상태에 있는 스레드들 중 하나를 선택하여 처리기를 할당하는 과정이다. 오늘날 다중프로그래밍 운영체제에서 수행 단위가 스레드이므로 처리기 스케줄링은 스레드를 대상으로 하며, 가장 중요한 운영체제의 기능이다.

학습 목표

- 처리기 스케줄링 알고리즘의 목표와 다양한 평가 기준을 설명할 수 있다.
- 처리기 스케줄링이 수행되는 상황과 커널에서 처리기 스케줄링이 일어나는 위치에 대해 이해한다.
- 선점 스케줄링과 비선점 스케줄링을 구분할 수 있다.
- 기아와 에이징의 개념을 이해한다.
- 다양한 처리기 스케줄링 알고리즘의 특성을 비교•설명할 수 있다.

학습 내용

- 단일 처리기 스케줄링 개요
- 단일 처리기 스케줄링 기본
- 단일 처리기 스케줄링 알고리즘들
- 멀티코어 환경에서의 스케줄링

1. 단일 처리기 스케줄링 개요

- 어떤 **목적**을 위해 **시스템 자원**을 프로세스(스레드)에게 할당하는 것
  - ➢목적(objectives)
    - ✓처리기 활용률 향상, 컴퓨터 시스템 처리율 향상
  - ➢시스템 자원(resources)
    - ✓컴퓨터 시스템, 메모리, 처리기, 입출력 장치, 파일, 데이터베이스 등

1. 단일 처리기 스케줄링 개요

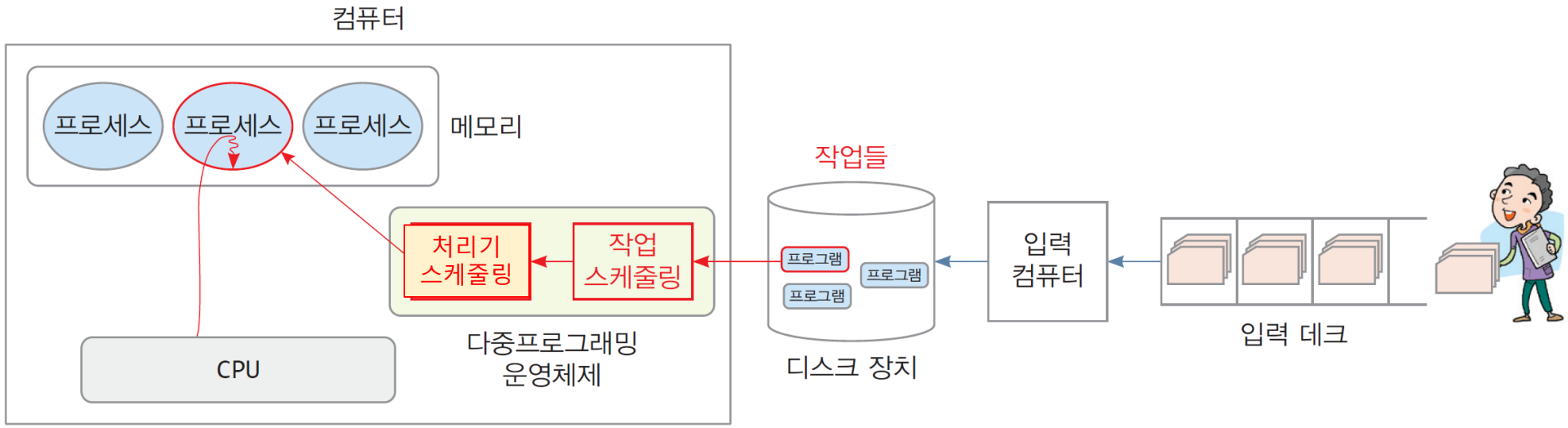
- 스케줄링은 왜 필요할까?
  - ➢자원에 대한 경쟁이 있는 곳에서 경쟁자 중 하나 선택
  - ➢자원: 컴퓨터 시스템, 메모리, 처리기, 디스크, 프린트, 파일, 데이터베이스 등
- 컴퓨터 시스템 내 다양한 스케줄링
  - ➢작업(job) 스케줄링
    - ✓대기중인 작업(Job) 중 메모리에 적재할 작업 결정
  - ➢처리기 스케줄링
    - ✓처리기를 프로세스/스레드 중에 하나를 선택하여 할당
  - ➢디스크 스케줄링
    - ✓디스크 장치 내에서 디스크 입출력 요청 중 하나 선택

1) 다중프로그래밍과 스케줄링

- 다중프로그래밍의 도입 목적 리뷰
  - ➢처리기 유휴 시간 줄여 처리기 활용률 향상
    - ✓수행 중인 프로세스가 I/O를 요청하면 다른 프로세스에게 처리기 할당
- 다중프로그래밍과 함께 2가지 스케줄링 도입
  - ➢작업 스케줄링(job scheduling)
    - ✓디스크 장치로부터 메모리에 올릴 작업 선택(초기 다중프로그래밍 시스템에서)
    - ✓처음에 혹은 프로세스가 종료할 때마다
  - ➢처리기 스케줄링(processor scheduling)
    - ✓메모리에 적재되어 있는 준비 상태인 프로세스 중 처리기에 수행시킬 프로세스 선택

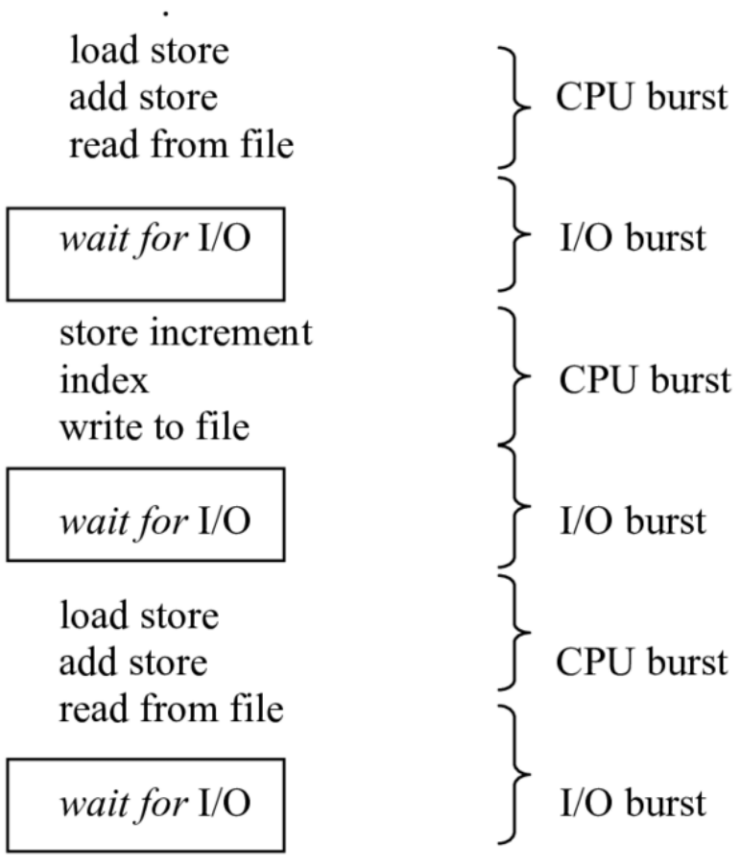
1) 다중프로그래밍과 스케줄링

- 다중프로그래밍 시스템에서 작업 스케줄링과 처리기 스케줄링



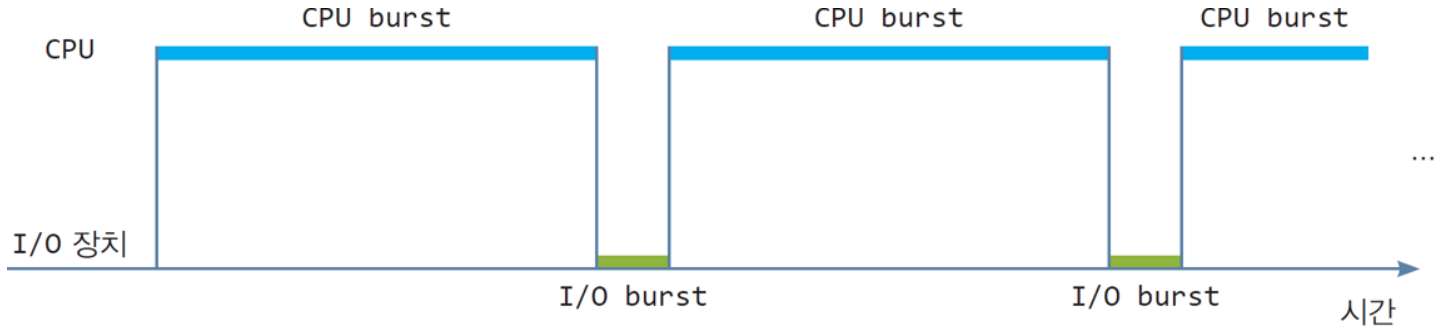
2) 프로세스 수행 특성

- 프로세스는 수행 중 CPU 연산 작업과 I/O 작업(화면 출력, 키보드, 입력, 파일 입출력 등)을 순차적으로 섞어함
- CPU burst
  - ➢프로그램 수행 중 CPU 연산(계산 작업)이 연속적으로 수행되는 상황
  - ➢프로세스가 CPU에서 일부 코드를 수행하는 데 소비하는 시간
- I/O burst
  - ➢프로그램 수행 중 I/O 장치의 입출력이 이루어지는 상황
  - ➢프로세스가 I/O 요청 완료를 기다리는 데 소비하는 시간

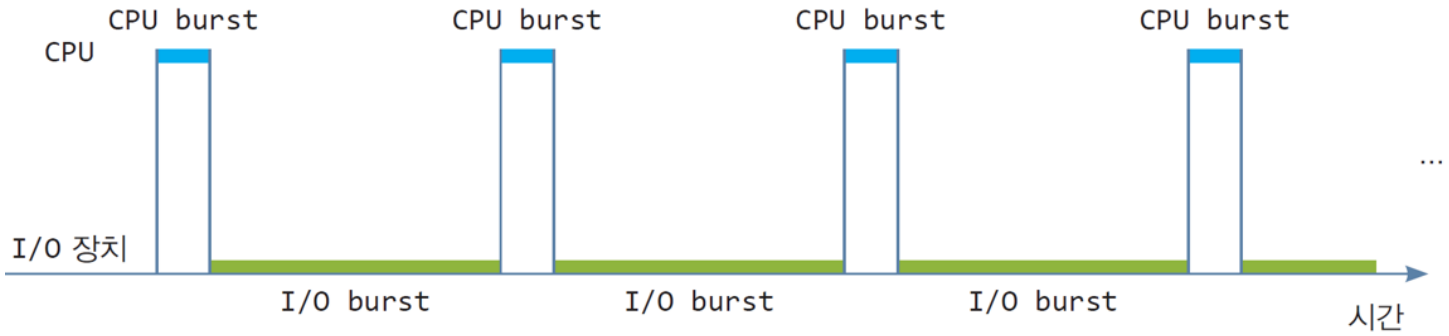


2) 프로세스 수행 특성

- **CPU 집중 프로세스**
  - ➢비디오나 이미지 처리 프로그램, 과학계산 프로그램 등



- **I/O 집중 프로세스**
  - ➢인터넷 검색, 데이터베이스 검색 프로그램 등



3) 단일 처리기 스케줄링 알고리즘의 평가 기준

- 처리기 활용률(processor utilization)
  - ✓전체 시간 중 처리기의 사용 시간 비율, 운영체제 입장
- 처리율(throughput)
  - ✓단위 시간당 처리하는 스레드 개수, 운영체제 입장
- 공평성(fairness)
  - ✓처리기를 스레드들에게 공평하게 배분, 사용자 입장
- 응답시간(response time)
  - ✓대화식 사용자의 경우, 사용자에 대한 응답 시간, 사용자 입장
- 소요시간(turnaround time) = 반환시간
  - ✓프로세스(스레드)가 컴퓨터 시스템에 도착한 후(혹은 생성된 후) 완료될 때까지 걸린 시간, 사용자 입장
- 대기시간(waiting time)
  - ✓스레드가 준비 큐에서 머무르는 시간, 운영체제와 사용자 입장
- 시스템 정책(policy enforcement) 우선
  - ✓컴퓨터 시스템의 특별한 목적을 달성하기 위한 스케줄링, 운영체제 입장
  - ✓예) 실시간 시스템에서는 스레드가 완료시한(deadline) 내에 스케줄링하는 정책
- 자원 활용률(resource efficiency)

4) 처리기 스케줄링과 타임 슬라이스

- 대부분 운영체제에서
  - ➢하나의 스레드가 너무 오래 동안 처리기를 사용하도록 허용되지 않음
- 타임 슬라이스(time slice)
  - ➢스케줄된 스레드에게 한 번 할당하는 처리기 시간
    - ✓스레드가 처리기 사용을 보장받는 시간
  - ➢커널이 스케줄을 단행하는 주기 시간
    - ✓타이머 인터럽트의 도움을 받아 타임 슬라이스 단위로 처리기 스케줄링
    - ✓현재 수행중인 스레드 강제 중단(preemption), 준비 리스트에 삽입
  - ➢타임 퀀텀(time quantum), 타임 슬롯(time slot)이라고도 함

2. 스케줄링 기본

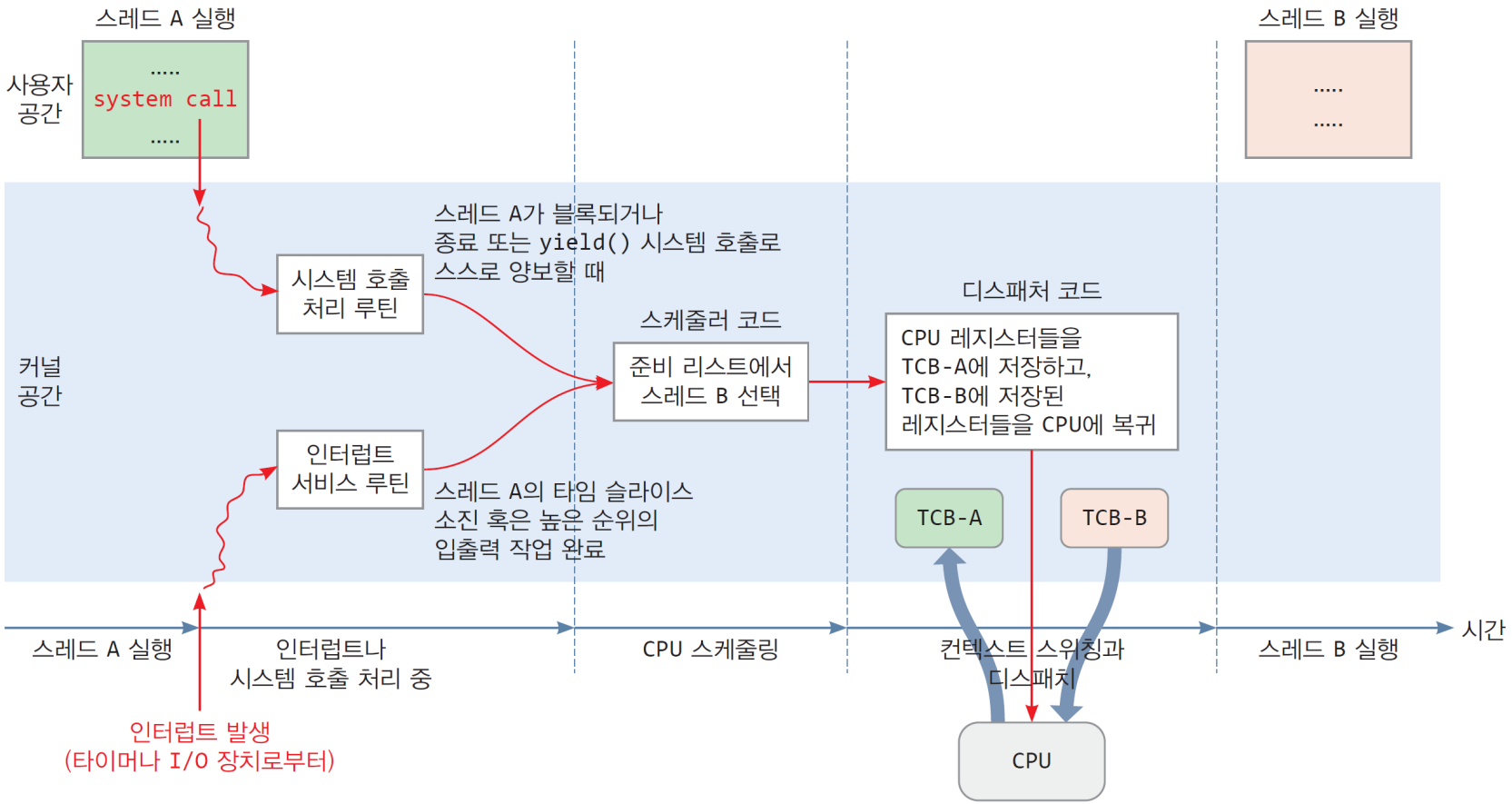
- **처리기 스케줄링이 수행되는 4가지 상황**
  - ➢스레드가 시스템 호출 끝에 I/O를 요청하여 블록될 때
  - ➢스레드가 자발적으로 처리기를 반환할 때
  - ➢스레드의 타임 슬라이스가 소진되어 타이머 인터럽트 발생
  - ➢더 높은 순위의 스레드가 요청한 입출력 작업 완료 인터럽트 발생

1) 처리기 스케줄링과 디스패치

- **처리기 스케줄링 코드의 위치와 수행 시점**
  - ➢스케줄링 코드는 어디에 위치?
    - ✓스케줄링 코드는 커널 코드의 일부로서 호출되어야 수행되는 함수 형태
  - ➢스케줄링 코드가 수행되는 시점은?
    - ✓시스템 호출이나 인터럽트 서비스 루틴이 끝나는 마지막 단계에서 수행
- **디스패칭(dispatching) 코드 수행**
  - ➢스레드 교환을 수행하는 커널 코드
  - ➢스케줄러에 의해 선택된 스레드를 처리기가 수행하도록 하는 작업
  - ➢커널 모드에서 사용자 모드로 전환
- 스케줄러와 디스패처 모두 수행 시간이 짧도록 작성

1) 처리기 스케줄링과 디스패치

- 처리기 스케줄링과 처리기 디스패치

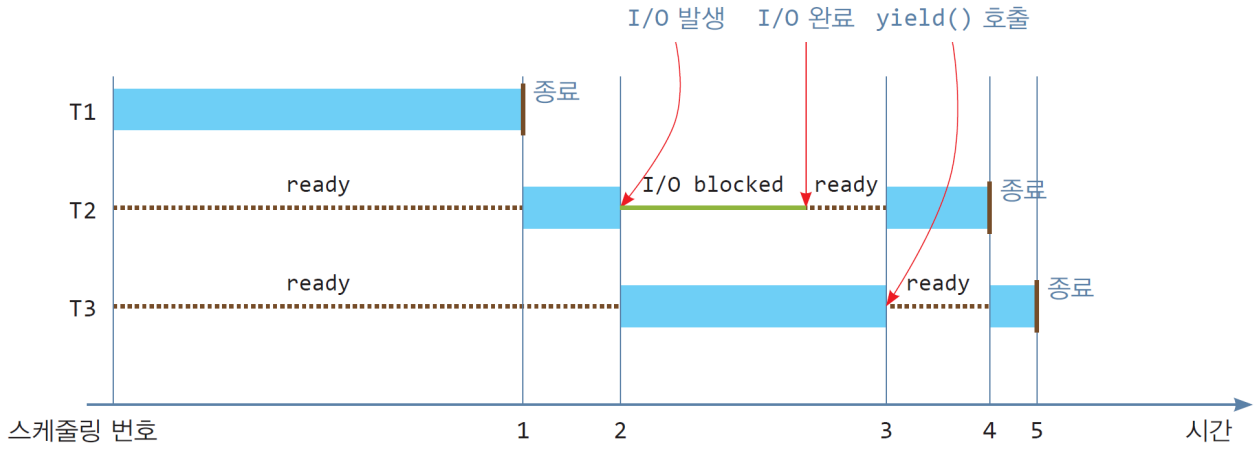


2) 스케줄링 타입

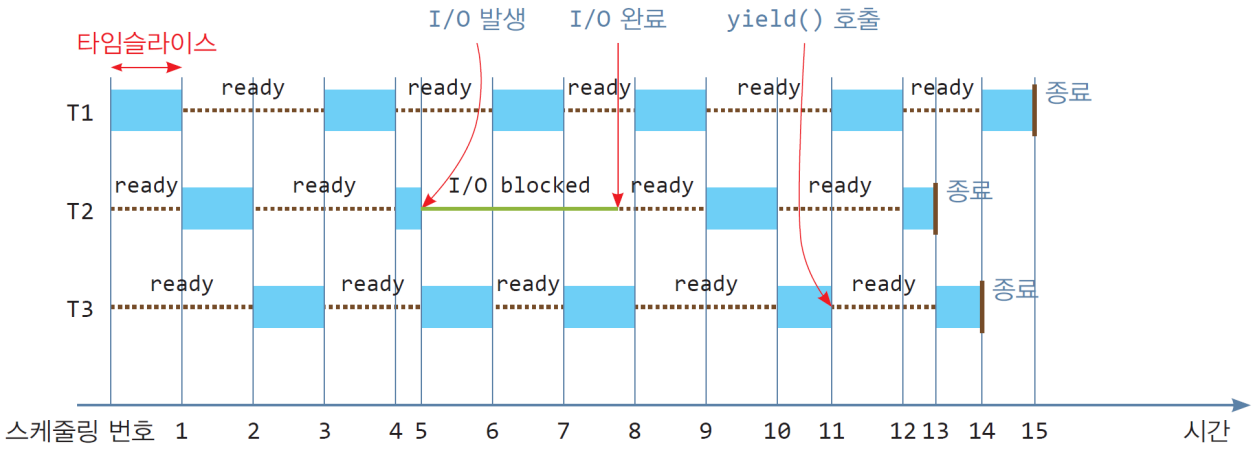
- 수행중인 스레드의 강제 중단 여부에 따른 처리기 스케줄링
  - ➢비선점 스케줄링(non-preemptive scheduling) 타입
    - ✓현재 수행중인 스레드를 강제로 중단시키지 않음
    - ✓스케줄링 시점
      - 처리기를 더 이상 사용할 수 없게 된 경우: I/O로 인한 블록 상태, sleep 등
      - 자발적으로 처리기 양보할 때
      - 종료할 때
  - ➢선점 스케줄링(preemptive scheduling) 타입
    - ✓현재 수행중인 스레드를 강제 중단시키고 다른 스레드 선택
    - ✓스케줄링 시점
      - 타임 슬라이스가 소진되어 타이머 인터럽트가 발생될 때
      - 인터럽트나 시스템 호출 종료 시점에서, 더 높은 우선순위의 스레드가 준비 상태일 때
      - 스케줄링 기준을 충족할 때

2) 스케줄링 타입

- 비선점 스케줄링과 선점 스케줄링의 비교



(a) 비선점 스케줄링 사례



(b) 선점 스케줄링 사례

3) 기아와 에지징

- 기아(starvation)
  - ➢스레드가 스케줄링되지 못한 채 오랜 동안 준비 상태에 있는 상황
  - ➢사례
    - ✓우선순위 기반 시스템에서, 더 높은 우선순위의 스레드가 계속 시스템에 들어오는 경우
    - ✓수행 시간이 짧은 스레드를 우선 수행시키는 시스템에서, 자신보다 수행 시간이 짧은 스레드가 계속 도착하는 경우
  - ➢스케줄링 알고리즘 설계 시 기아가 발생하지 않도록 설계하는 것이 바람직함
- 에이징(aging)
  - ➢스레드가 준비 상태에 머무르는 시간에 비례하여 스케줄링 순위를 높이는 기법
  - ➢기아의 해결책

3. 단일 처리기 스케줄링 알고리즘

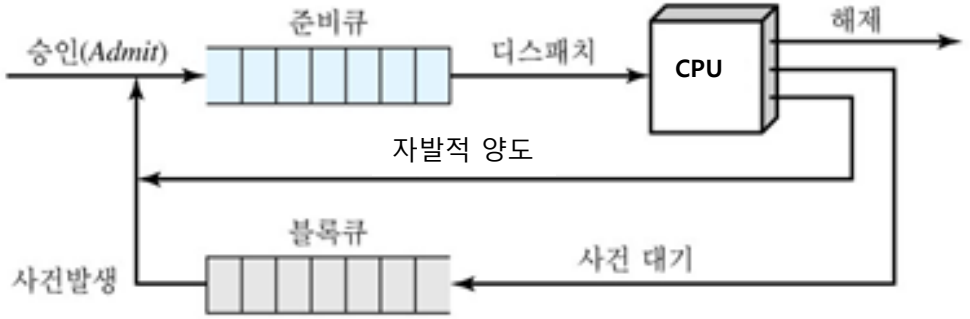
- FCFS(First Come First Served): 비선점 스케줄링
- RR(Round-Robin): 선점 스케줄링
- SJF(Shortest Job First): 비선점 스케줄링
- SRTF(Shortest Remaining Time First): 선점 스케줄링
- MLFQ(Multi-level Feedback Queue): 선점 스케줄링
- Priority Scheduling: 선점 스케줄링
- MLQ(Multi-level Queue): 비선점 스케줄링

3. 단일 처리기 스케줄링 알고리즘

- 선택 함수(selection function)
  - ➢다음 번 수행을 위해 준비 큐에서 대기 중인 프로세스 중 하나를 선택할 때 사용하는 알고리즘을 함수 형태로 표현
  - ➢함수 인자
    - ✓w: 지금까지 수행 또는 대기하면서 시스템 내에 머문 시간
    - ✓e: 지금까지 수행하는 데에 소요한 시간
    - ✓s: e를 포함하여 프로세스가 요구한 총 수행 시간
- 스케줄링 타입
  - ➢선택 함수가 호출되는 시점이 언제인가 하는 것을 나타냄
  - ➢비선점(Non-preemptive)형
    - ✓프로세스 일단 수행 상태에 진입하면 종료되거나 자발적으로 CPU를 놓을 때까지 CPU를 빼앗기지 않는 타입
  - ➢선점(Preemptive)형
    - ✓운영체제가 현재 수행 중인 프로세스를 인터럽트하여 준비 큐로 이동시키고 CPU를 강제적으로 빼앗는 타입

1) FCFS(First Come First Served) 스케줄링

- 알고리즘
  - ➢준비 큐에서 대기 중이던 스레드 중 가장 오랫동안 기다렸던 스레드 선택
- 선택 함수: max(w)
- 스케줄링 타입: 비선점형
- 기아: 발생하지 않음
- 성능 이슈
  - ➢수행시간이 짧은 스레드보다 긴 스레드에게 유리함
  - ➢입출력 중심 스레드보다 CPU 중심 스레드를 우대하는 경향이 있음 → CPU와 입출력 장치의 이용률이 저하됨

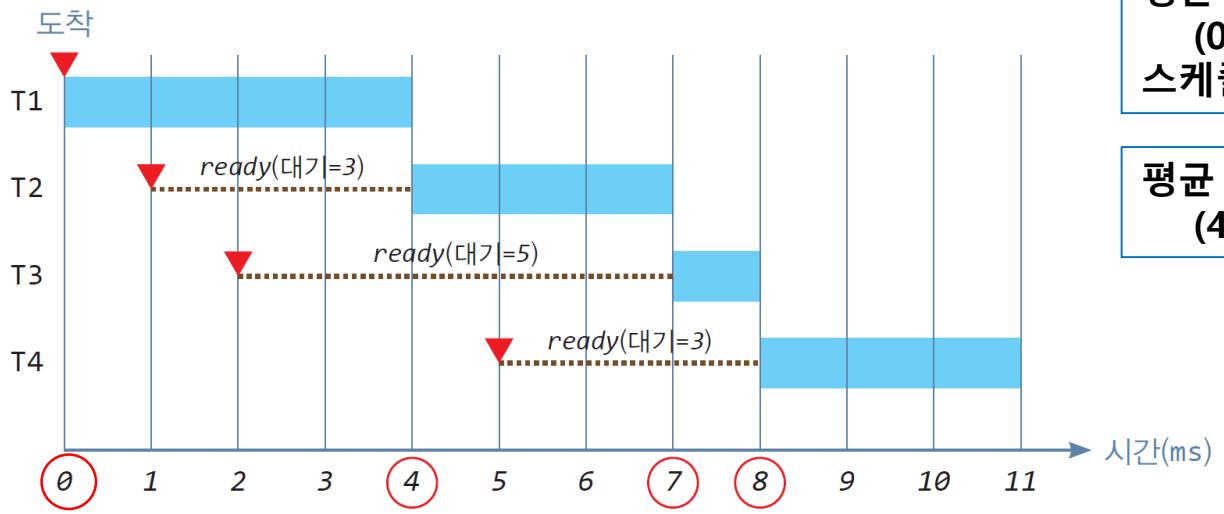


1) FCFS(First Come First Served) 스케줄링

- FCFS 스케줄링 사례

스레드	도착 시간	실행 시간(ms)
T1	0	4
T2	1	3
T3	2	1
T4	5	3

실행 시간 동안 입출력은 발생하지 않는다고 가정한다.



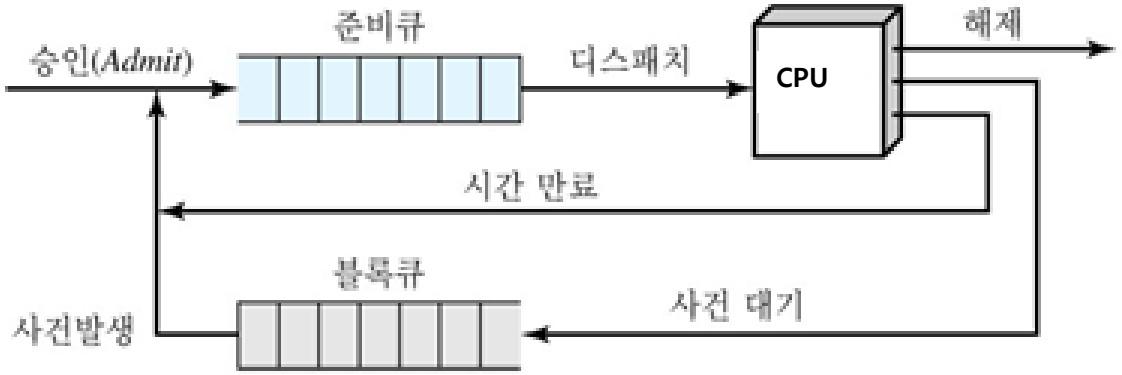
평균 대기 시간 :
(0 + 3 + 5 + 3)/4 = 11/4 = 2.75ms
스케줄링 횟수: 4번

평균 반환 시간 :
(4 + 6 + 6 + 6)/4 = 22/4 = 5.5ms

※ 빨간 원은 스케줄링이 일어나는 시점을 나타낸다.

2) RR(Round-Robin) 스케줄링

- 알고리즘
  - ➢정해진 일정한 시간 주기로 준비 큐에 대기중인 스레드를 FCFS로 선택
  - ➢스레드가 수행 중 정해진 일정한 시간이 만료되면 준비 큐 끝으로 이동
  - ➢RR은 FCFS에서 짧은 스레드가 피해보는 현상을 완화하는 가장 간단한 방법임
- 선택 함수: 시간 할당량(time slicing, time quantum)
- 스케줄링 타입: 선점형
- 기아: 없음

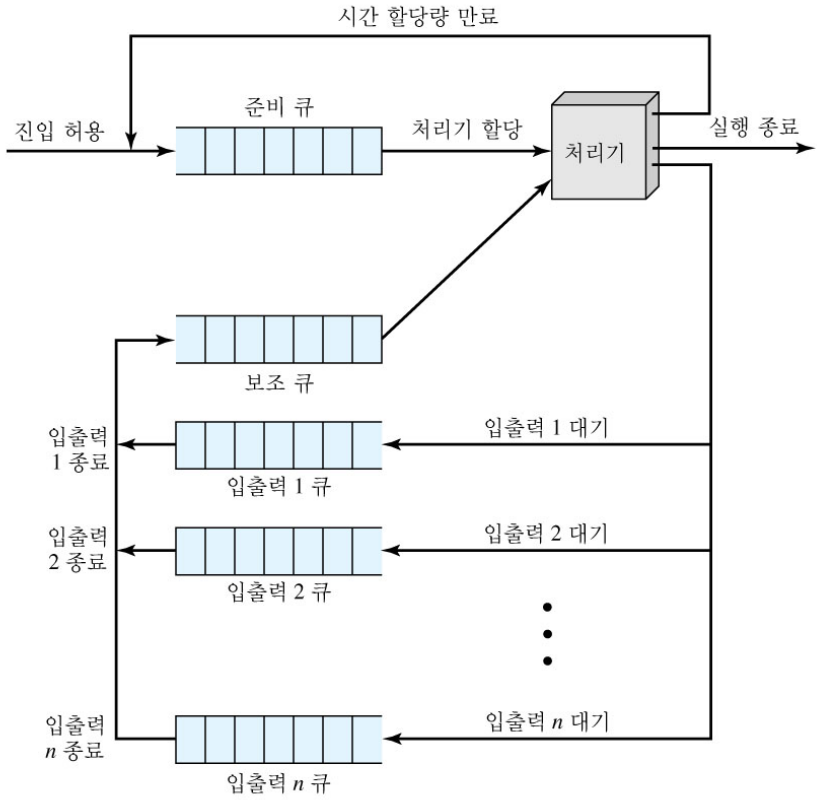


2) RR(Round-Robin) 스케줄링

- 성능 이슈
  - ➢공평하고, 기아 현상 없고, 구현이 쉬움
  - ➢시간 할당량 길이가 성능에 영향을 미침
    - ✓너무 짧다면: 잦은 스케줄링과 스레드 교환으로 전체 스케줄링 오버헤드가 커짐
    - ✓너무 길다면: FCFS와 비슷해 짐
  - ➢균형된 처리율
    - ✓늦게 도착한 짧은 스레드는 FCFS보다 빨리 완료되고, 긴 스레드는 SJF보다 빨리 완료됨
  - ➢I/O 집중 스레드보다 CPU 집중 스레드를 우대함
    - ✓CPU 집중 스레드는 시간 할당량을 전부 소비한 후 선점 → 준비 큐 대기
    - ✓I/O 집중 스레드는 처리기를 잠깐 사용한 후, 입출력 요구 → 블록 → 준비 큐 대기
    - ✓I/O 집중 스레드 성능 저하, 입출력 장치의 사용률 저하, 스레드간 응답시간의 편차 증가

2) RR(Round-Robin) 스케줄링

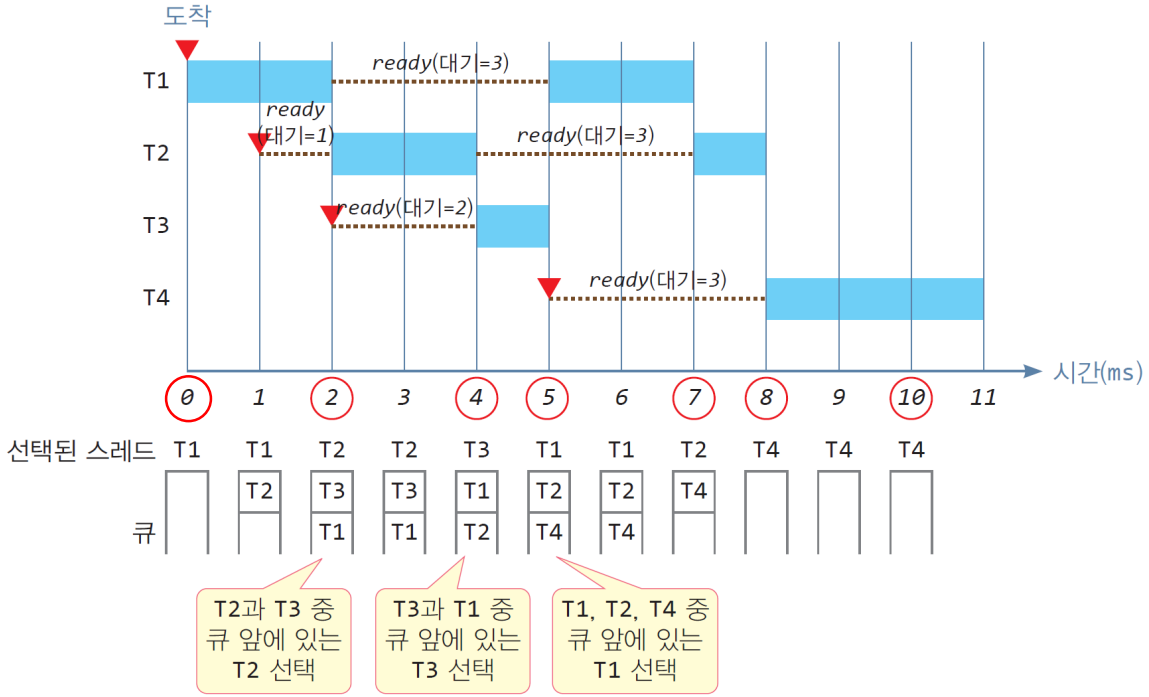
- I/O 집중 스레드보다 CPU 집중 스레드를 우대하는 해결 방안
  - ➢가상(virtual) RR 스케줄링



2) RR(Round-Robin) 스케줄링

- RR 스케줄링 사례(타임 슬라이스: 2ms일 때)

스레드	도착 시간	실행 시간(ms)
T1	0	4
T2	1	3
T3	2	1
T4	5	3



평균 대기 시간 :
(3 + 4 + 2 + 3)/4 = 12/4 = 3ms

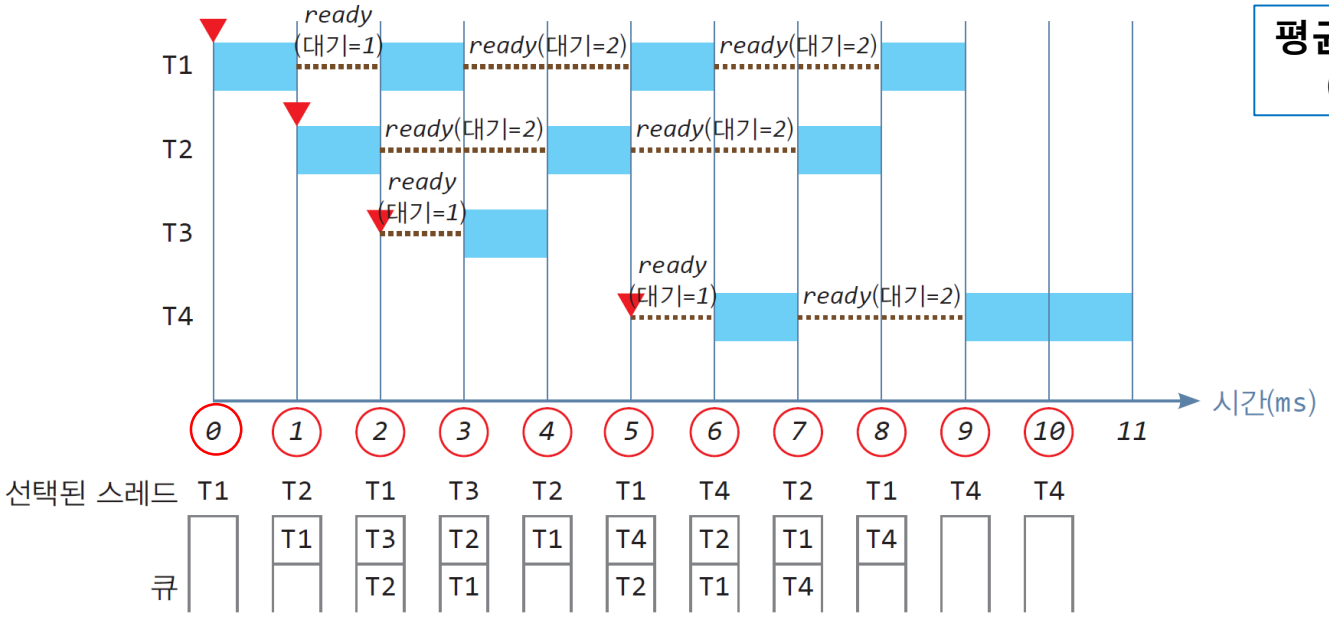
스케줄링 횟수: 7번

평균 반환 시간 :
(7 + 7 + 3 + 6)/4 = 23/4 = 5.75ms

2) RR(Round-Robin) 스케줄링

- RR 스케줄링 사례(타임 슬라이스: 1ms일 때)

스레드	도착 시간	실행 시간(ms)
T1	0	4
T2	1	3
T3	2	1
T4	5	3



평균 대기 시간 :
(5 + 4 + 1 + 3)/4 = 13/4 = 3.25ms

스케줄링 횟수: 11번

평균 반환 시간 :
(8 + 7 + 2 + 6)/4 = 23/4 = 5.75ms

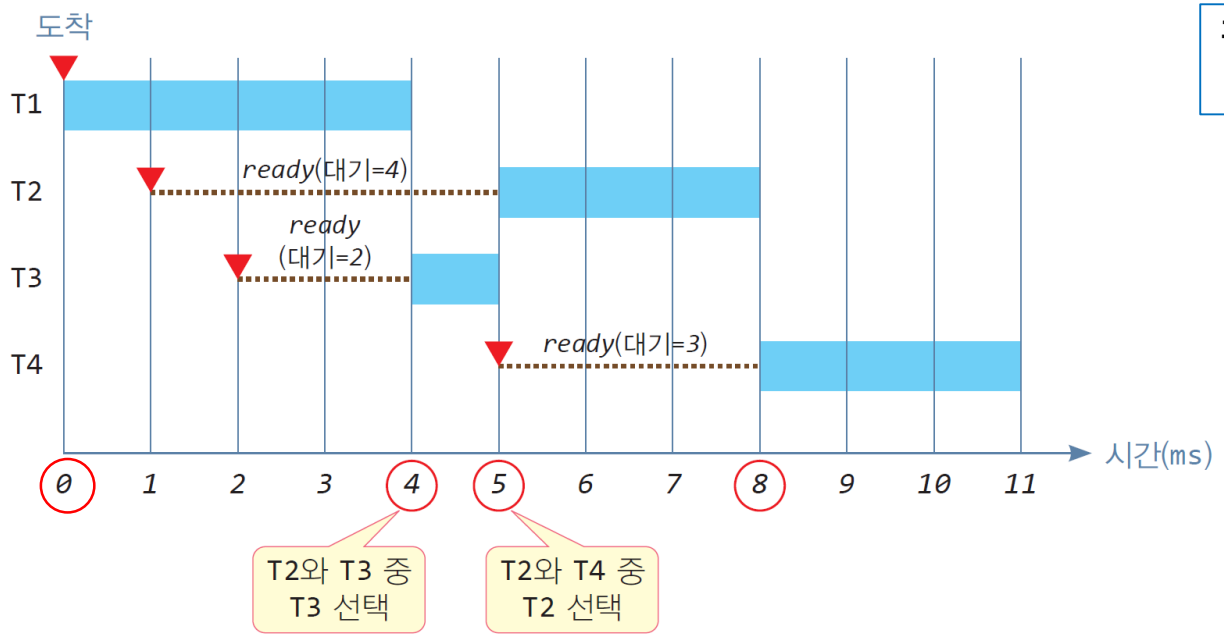
3) SJF(Shortest Job First) 스케줄링

- 알고리즘
  - ➢준비 큐에 대기하는 스레드 중 수행 시간(s)이 가장 짧은 스레드 선택
  - ➢FCFS에서 수행 시간이 긴 스레드만 우대하는 편향성을 완화시키는 접근법
  - ➢평균 대기 시간을 최소화함
- 선택 함수: min(s)
- 스케줄링 타입: 비선점형
- 기아: 발생 가능
  - ➢수행 시간이 짧은 스레드가 계속 도착하면, 수행 시간이 긴 스레드는 수행 기회를 언제 얻을 지 예측할 수 없음
- 문제점
  - ➢수행 시간 예측의 어려움

3) SJF(Shortest Job First) 스케줄링

- SJF 스케줄링 사례

스레드	도착 시간	실행 시간(ms)
T1	0	4
T2	1	3
T3	2	1
T4	5	3



평균 대기 시간 :
(0 + 4 + 2 + 3)/4 = 9/4 = 2.25ms

스케줄링 횟수: 4회

평균 반환 시간 :
(4 + 7 + 3 + 6)/4 = 20/4 = 5.0ms

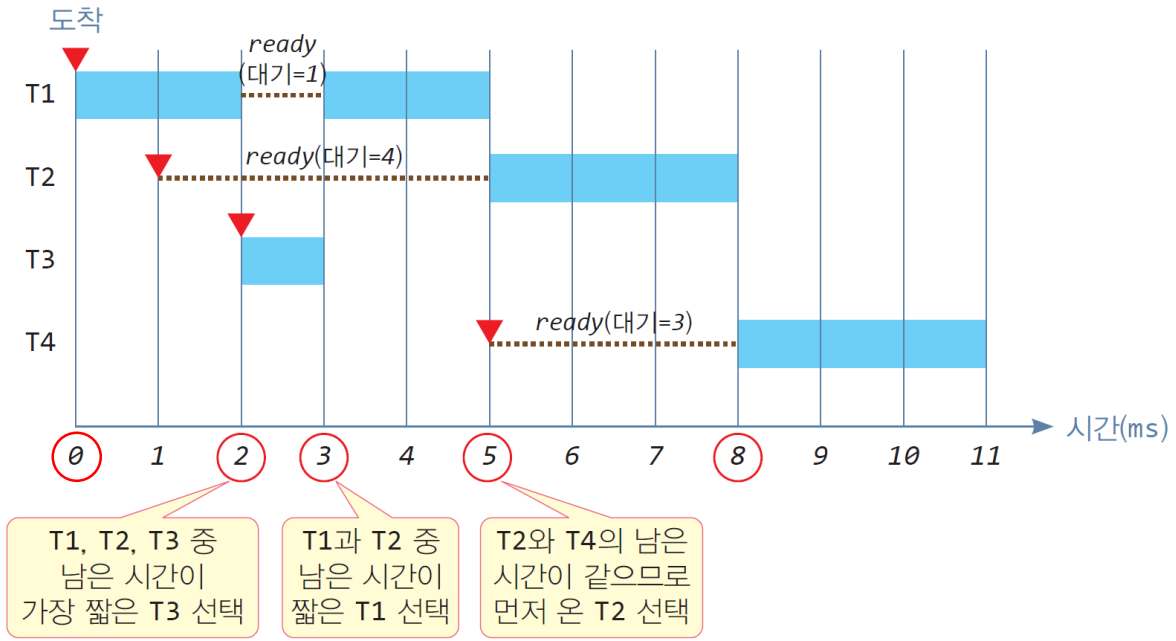
4) SRTF(Shortest Remaining Time First) 스케줄링

- 알고리즘
  - ➢준비 큐에 도착한 스레드 중 남아있는 수행 시간이 가장 짧은 스레드 선택
  - ➢SJF의 선점형 버전으로, 평균 대기 시간을 최소화 함
- 선택 함수: min(s-e)
- 스케줄링 타입: 선점형
- 기아: 발생 가능
  - ➢짧은 스레드가 계속 도착하면, 긴 스레드는 수행 기회를 언제 얻을 지 모름
- 문제점
  - ➢수행 시간 예측의 어려움

4) SRTF(Shortest Remaining Time First) 스케줄링

- SRTF 스케줄링 사례

스레드	도착 시간	실행 시간(ms)
T1	0	4
T2	1	3
T3	2	1
T4	5	3



평균 대기 시간 :
(1 + 4 + 0 + 3)/4 = 8/4 = 2ms

스케줄링 횟수: 5회

평균 반환 시간 :
(5 + 7 + 1 + 6)/4 = 19/4 = 4.75ms

5) MLFQ(Multi-level Feedback Queue) 스케줄링

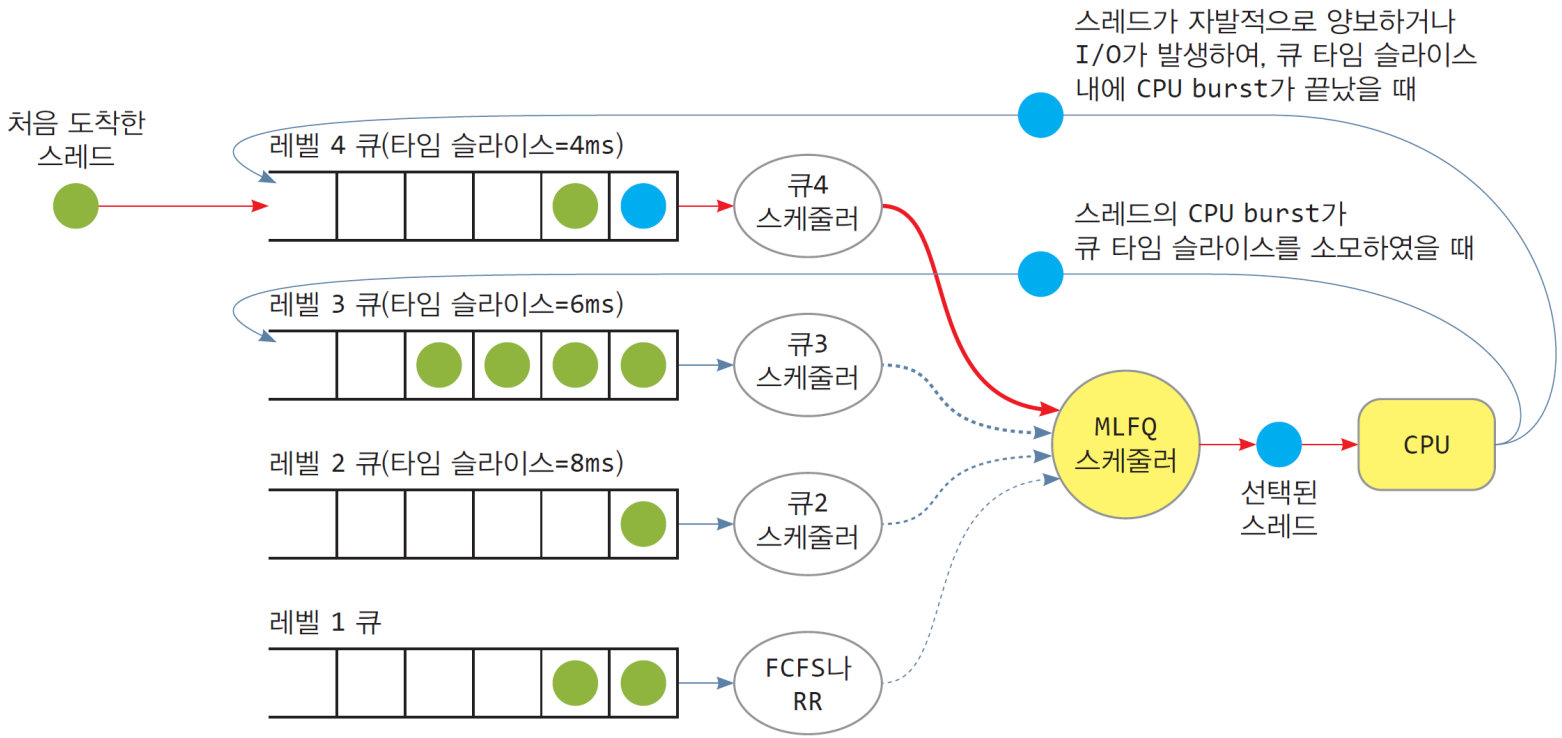
- 알고리즘
  - ➢n개의 우선순위 레벨 큐
  - ➢큐마다 시간 할당량 설정, 낮은 레벨의 큐일수록 더 큰 시간 할당량 설정
  - ➢스레드는 도착 시 가장 높은 우선순위 레벨 큐에 삽입
  - ➢가장 높은 우선순위 레벨 큐에서 스레드 선택함
    - ✓만일 비어 있으면, 그 다음 우선순위 큐에서 스레드 선택
    - ✓스레드가 해당 큐의 시간 할당량을 초과하여 수행하면 강제로 다음 우선순위의 큐로 이동시킴
    - ✓스레드가 자발적 또는 입출력으로 수행을 중단한 경우, 현재 큐 끝에 삽입
  - ➢특정 큐에 있는 시간이 오래되면 기아를 막기 위해 바로 상위 레벨 큐로 이동
  - ➢최하위 레벨 큐는 FCFS 스케줄링 함

5) MLFQ(Multi-level Feedback Queue) 스케줄링

- 선택 함수: 각 큐의 시간 할당량
- 스케줄링 타입: 선점형
- 기아: 발생하지 않음
  - ➢큐에 대기하는 시간에 오래되면, 더 높은 레벨의 큐로 이동시킴(에이징 기법)
- 성능 이슈
  - ➢짧거나 입출력이 빈번한 스레드, 혹은 대화식 스레드를 높은 레벨의 큐에서 빨리 수행

5) MLFQ(Multi-level Feedback Queue) 스케줄링

- 4개의 우선순위 레벨을 가진 MLFQ 스케줄링

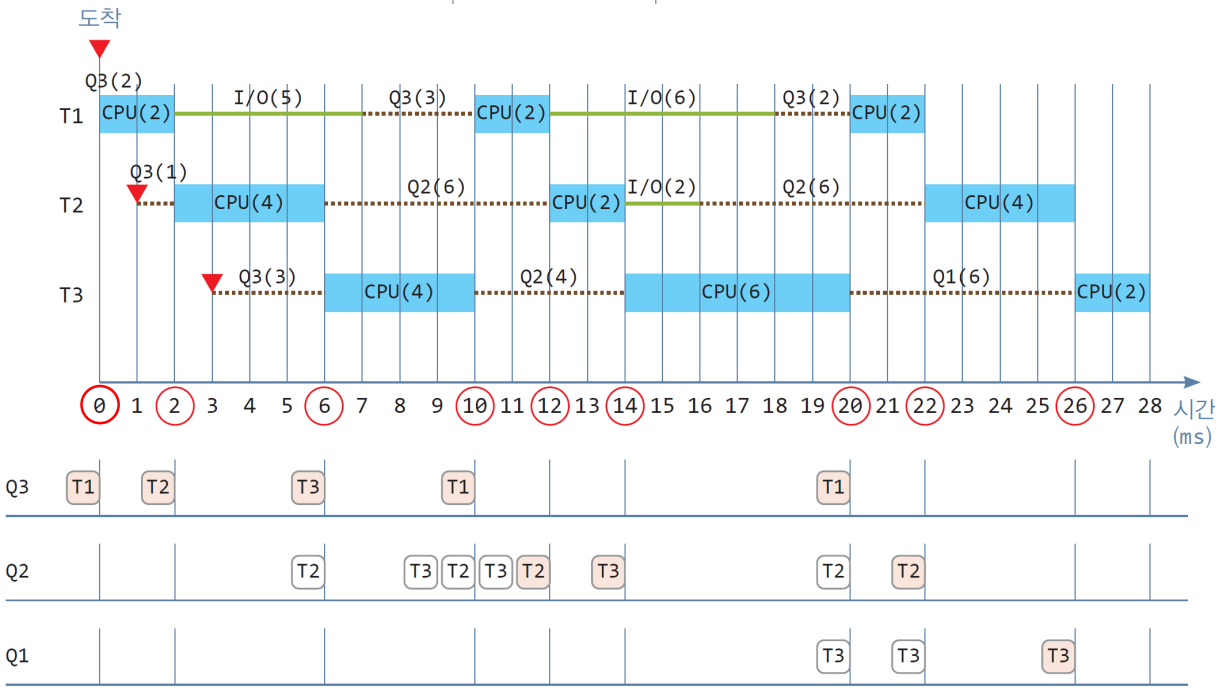


5) MLFQ(Multi-level Feedback Queue) 스케줄링

- MLFQ 스케줄링 수행 사례(3개의 큐, 3개의 스레드)

큐 타임 슬라이스 : Q3=4ms, Q2=6ms, Q1=무제한

스레드	도착 시간	실행 시간(ms) (CPU-I/O-CPU-I/O-CPU)
T1	0	2 - 5 - 2 - 6 - 2
T2	1	6 - 2 - 4
T3	3	12



평균 대기 시간 :
(5 + 13 + 13)/3 = 31/3 = 10.3ms

스케줄링 횟수: 9회

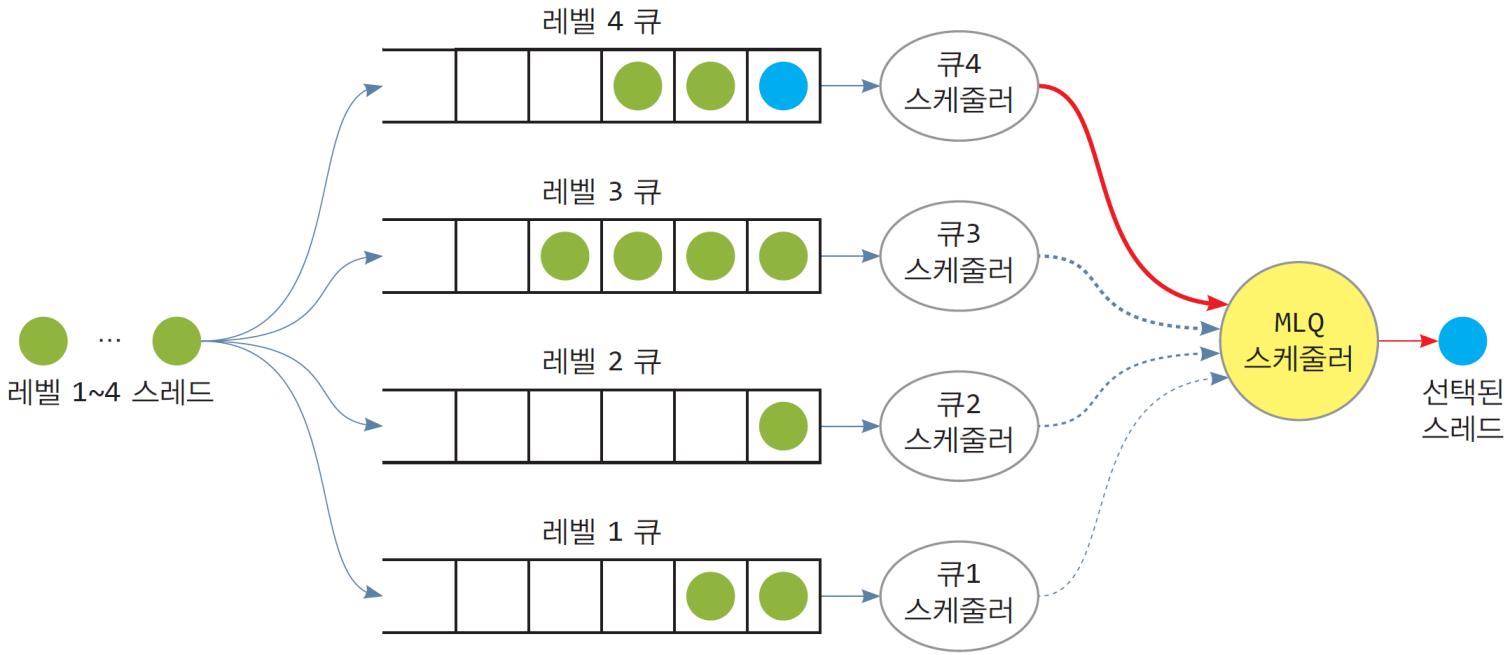
평균 반환 시간 :
(22 + 25 + 25)/3 = 72/3 = 24ms

6) 우선순위 기반 스케줄링

- 알고리즘
  - ➢모든 스레드에 고정 우선순위 할당, 종료 때까지 바뀌지 않음
  - ➢도착하는 스레드는 자신의 우선순위 준비 큐에 삽입(MLQ), 각 큐에서 FCFS 스케줄링
  - ➢가장 높은 우선순위의 준비 큐에서 스레드 선택
- 선택 함수: 가장 높은 우선순위 스레드 선택
- 스케줄링 타입: 선점형
  - ➢더 높은 우선순위의 스레드가 도착할 때 현재 스레드 강제 중단하고 스케줄링
- 스레드 우선순위: 있음
- 기아: 발생 가능
  - ➢높은 우선순위의 스레드가 계속 도착하는 경우
  - ➢큐 대기 시간에 비례하여 일시적으로 우선순위를 높이는 에이징 방법으로 해결 가능
- 특징
  - ➢스레드별 고정 우선순위를 가지는 실시간 시스템에서 사용

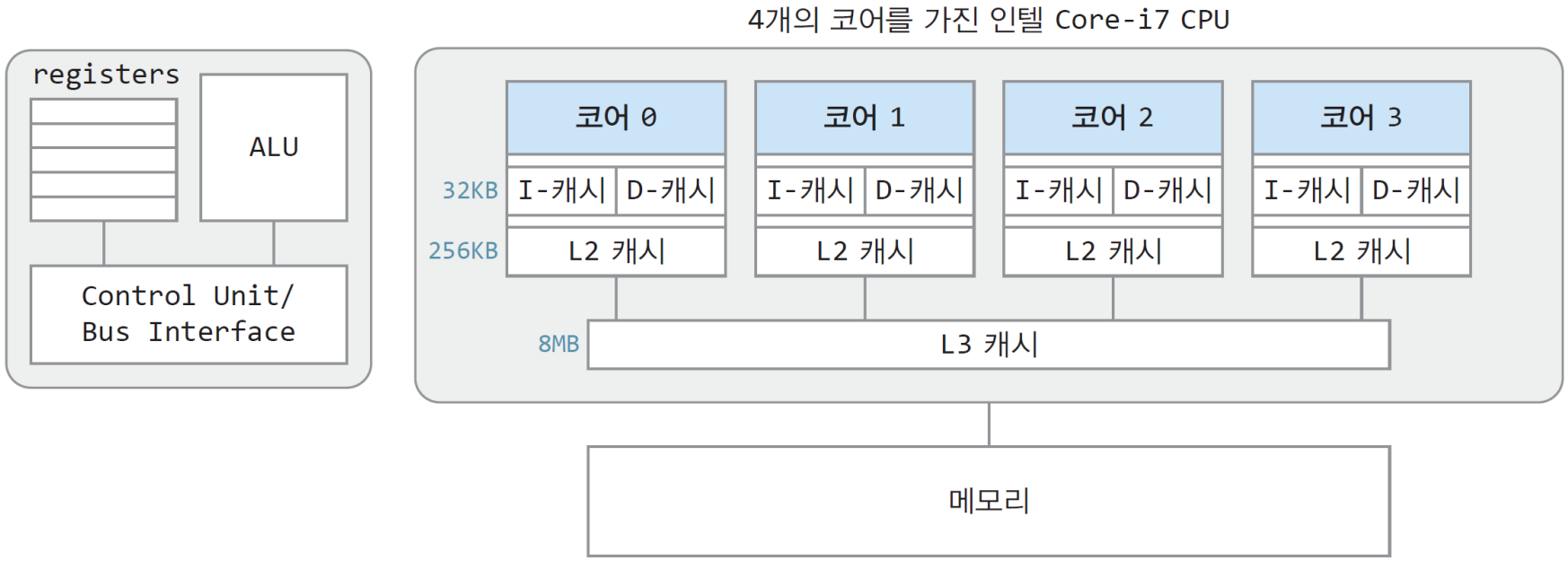
6) 우선순위 기반 스케줄링

- MLQ(Multo-level Queue) 스케줄링
  - ➢ 우선순위[RQ_i] < 우선순위[RQ_j] for i < j



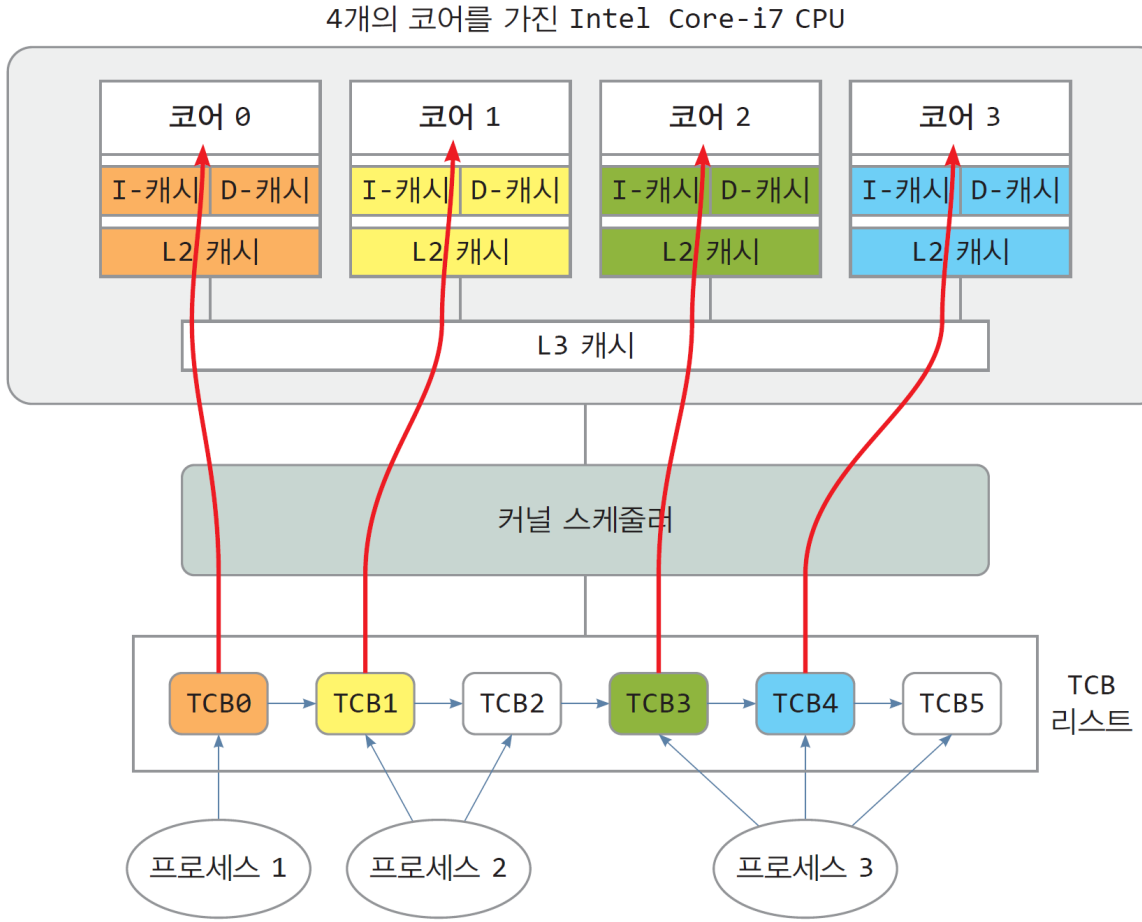
4. 멀티코어 CPU에서의 스케줄링

- 멀티코어 CPU



1) 멀티코어 CPU와 병렬처리

- 4개의 코어를 가진 CPU에서 멀티스레딩

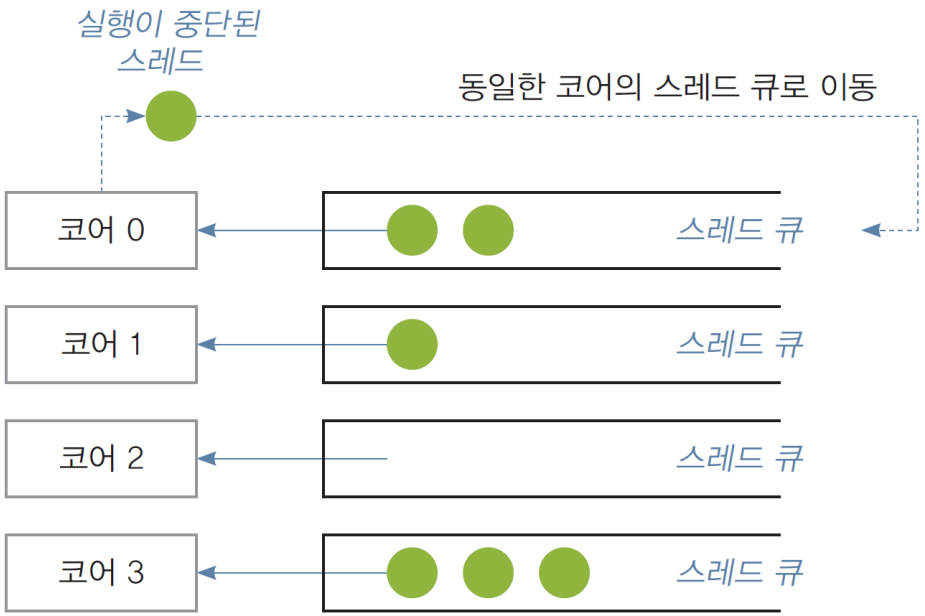


2) 멀티코어 시스템에서의 CPU 스케줄링

- 멀티코어 시스템에서 싱글 코어 CPU 스케줄링 알고리즘을 사용할 때 문제점
  - ➢(문제 1) 스레드 교환 후 오버헤드 문제
    - ✓이전에 수행된 적이 없는 코어에 스레드가 배치될 때,
    - ✓스레드 교환 후, 수행 중에 새로운 스레드의 코드와 데이터가 캐시에 채워지는 긴 경과 시간 필요
  - ➢(문제 2) 코어별 부하 불균형 문제
    - ✓스레드를 무작위로 코어에 할당하면, 코어마다 처리할 스레드 수의 불균형 발생

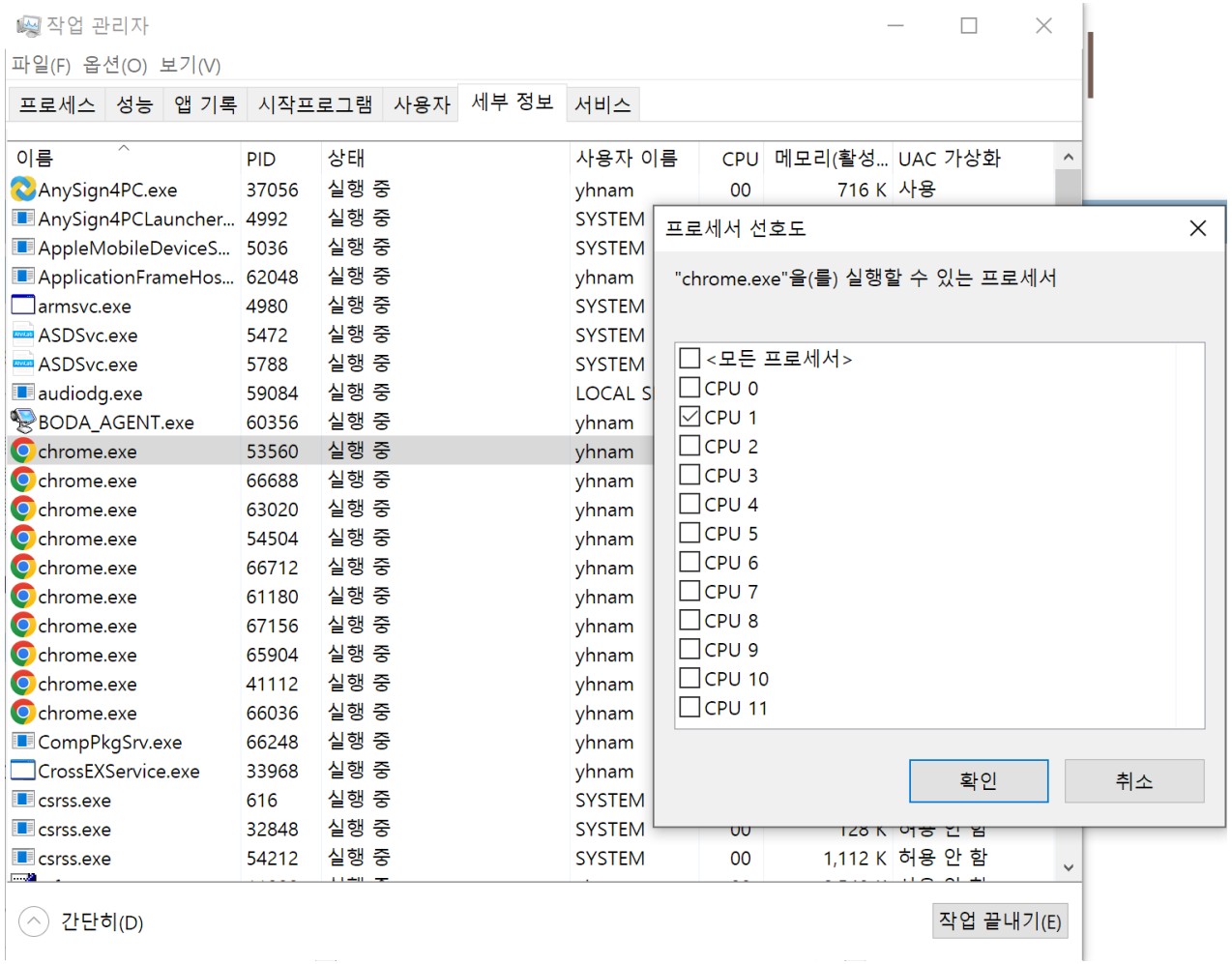
2) 멀티코어 시스템에서의 CPU 스케줄링

- 스레드 교환 후 오버헤드 문제 해결
  - ➢코어 친화성(core affinity) 적용
    - ✓스레드를 동일한 코어에서만 수행하도록 스케줄링
    - ✓코어 친화성(Core affinity), 캐시 친화성(cache affinity)라고도 부름
  - ➢코어 당 스레드 큐 사용



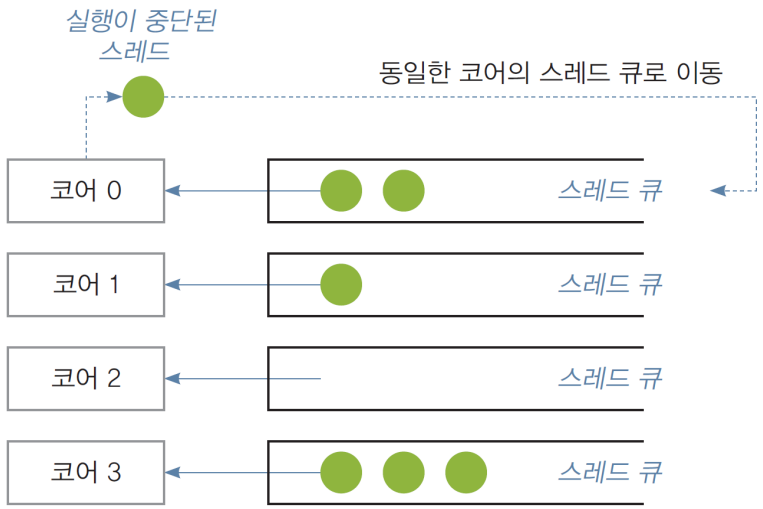
2) 멀티코어 시스템에서의 CPU 스케줄링

- Windows 사례: 작업관리자에서 코어 친화성 지정



2) 멀티코어 시스템에서의 CPU 스케줄링

- 코어별 부하 불균형 문제 해결
  - ➢부하 균등화(load balancing) 기법으로 해결
  - 1) 푸시 마이그레이션(push migration) 기법
    - ✓감시 스레드가, 짧거나 빈 큐를 가진 코어에 다른 큐의 스레드를 옮겨놓는 기법
  - 2) 풀 마이그레이션(pull migration) 기법
    - ✓코어가 처리할 스레드가 없게 되면, 다른 코어의 스레드 큐에서 자신의 큐로 가져와 수행시키는 기법



3) 멀티코어 시스템에서의 스케줄링 알고리즘

- 부하 공유(load sharing)
  - ➢모든 코어들에게 부하가 공평하게 분산되도록 함
    - ✓서로 다른 프로세스에 속한 모든 스레드들은 단일 전역 큐를 통하여 어떠한 코어에도 할당함
    - ✓먼저 들어온 스레드 우선 방식(FCFS)
  - ➢단점
    - ✓단일 전역 큐는 상호배제 조건이 유지되도록 관리되어야 함으로 성능의 병목점이 될 수 있음
    - ✓캐시 친화성(affinity)이 떨어져 비효율적이 될 수 있음
    - ✓스레드간 통신이나 동기화가 자주 발생한다면, 성능이 저하됨

3) 멀티코어 시스템에서의 스케줄링 알고리즘

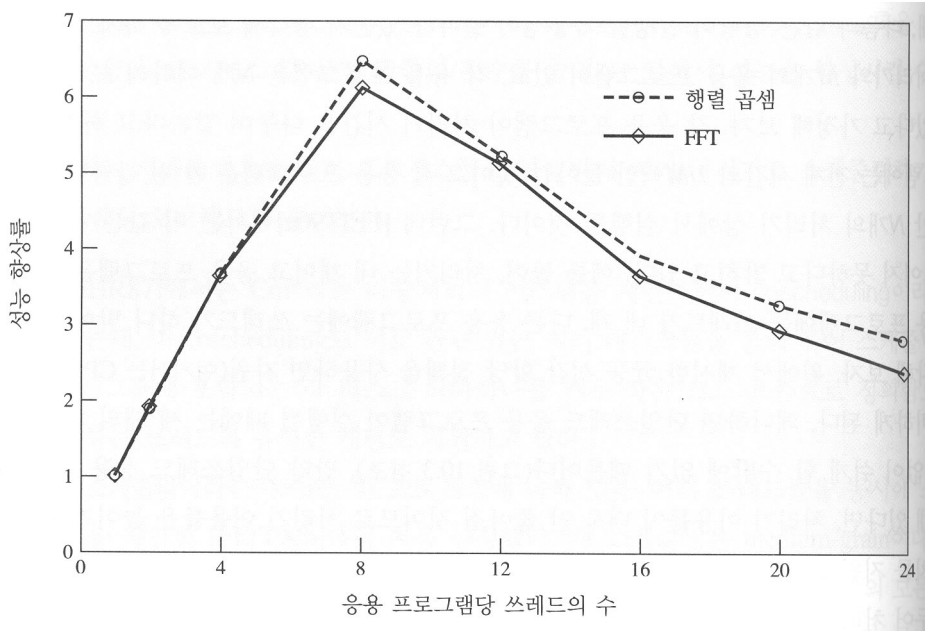
- 갱(gang) 스케줄링
  - ➢서로 관련이 있는 스레드의 집합을 코어 집합 상에 일대일 수행할 수 있도록 스케줄링하는 기법
  - ➢ULT와 KLT를 유지하여야 함
  - ➢장점
    - ✓서로 밀접한 관계가 있는 스레드들이 동시에 수행됨으로써 동기화로 인한 스레드 교환 횟수가 감소함
    - ✓한번에 여러 스레드들을 여러 코어로 스케줄링하기 때문에 스케줄링 오버헤드가 줄어듬
    - ✓멀티스레드 구조를 갖는 응용프로그램의 스케줄링에 효과적임
  - ➢단점
    - ✓스레드 관리가 복잡한

3) 멀티코어 시스템에서의 스케줄링 알고리즘

- 전용(dedicated) 코어 할당 기법
  - ➢응용프로그램을 구성하는 스레드 개수만큼 코어 할당
  - ➢응용프로그램이 종료하면 코어 집합 반환
  - ➢ULT와 KLT를 유지하여야 함
  - ➢문제점
    - ✓입출력 대기나 동기화로 인한 코어 시간 낭비
  - ➢적합한 환경
    - ✓코어 이용률이 성능에 별 영향을 미치지 않는 대단위 병렬 시스템에서 유리
    - ✓프로세스 교환이 큰 오버헤드로 작용하는 경우

3) 멀티코어 시스템에서의 스케줄링 알고리즘

- 전용(dedicated) 코어 할당 기법
  - ➢스레드 수의 변화에 따른 응용 프로그램의 성능 향상률
    - ✓16개의 처리기로 구성된 시스템에서 두 개의 응용프로그램을 구성하는 스레드 개수를 변화시키면서 동시에 수행



3) 멀티코어 시스템에서의 스케줄링 알고리즘

- 멀티코어 스레드 스케줄링 이슈
  - ➢멀티코어 시스템의 OS들은 부하 균형 스케줄링 방식을 채용함
  - ➢멀티코어 구조의 특성이 다양함 → 특성에 맞는 스케줄링 설계는 성능을 향상 시킬 수 있음