멀티프로세서 스케줄링 요약

1. 멀티프로세서 구조

멀티코어 프로세서는 더 많은 CPU 코어를 활용해 성능을 높이지만, 프로그램이 병렬로 실행되지 않으면 성능 개선에 한계가 있음.

운영체제는 여러 CPU에 작업을 효율적으로 분배해야 함.

2. 캐시 일관성 문제 (Cache Coherence)

여러 CPU가 메모리와 캐시를 공유할 때, 캐시에 저장된 값과 메모리 간의 불일치 문제가 발생.

이를 해결하기 위해 버스 스누핑과 같은 기법으로 캐시 일관성을 유지함.

3. 동기화의 필요성

여러 CPU가 동일한 데이터에 접근할 때 락(mutex) 등 동기화 기법을 사용해 데이터 정합성을 보장해야 함.

락을 사용하지 않으면 데이터 손실 및 충돌 가능성 있음.

4. 캐시 친화성 (Cache Affinity)

프로세스가 같은 CPU에서 계속 실행되면, 캐시 재사용으로 성능이 향상됨.

그러나 프로세스가 여러 CPU를 옮겨 다니면 캐시 초기화가 반복되어 성능이 저하됨.

스케줄링 방식

1. 단일 큐 멀티프로세서 스케줄링 (SQMS)

단일 큐에서 모든 CPU가 작업을 가져감.

장점: 구현이 단순하며 기존 스케줄러를 그대로 활용 가능.

단점: 성능 저하(확장성 부족) 및 캐시 친화성 문제 발생.

2. 멀티 큐 멀티프로세서 스케줄링 (MQMS)

CPU마다 별도의 큐를 운영하여 작업 분배.

장점: 확장성이 높고 캐시 친화성 유지에 유리함.

단점: 작업의 불균형(Load Imbalance) 문제 발생.

해결법: 작업을 이주(Migration) 시켜 균형을 맞춤.

Work Stealing: 한 큐가 다른 큐에서 작업을 훔쳐 균형을 맞춤.

Linux 스케줄러 종류

1. O(1) 스케줄러

우선순위 기반 스케줄러로 빠르게 결정.