다중 처리기 스케줄링

다중 처리기 스케줄링에 대한 접근 방법

1. **비대칭 다중처리** → 마스터 서버

   하나의 프로세서가 모든 스케줄링 결정과 I/O 처리 그리고 다른 시스템의 활동을 취급하는 것

   하나의 코어만 시스템 자료구조에 접근하여 **자료 공유의 필요성이 없어**지기에 간단하다. 하지만 모든 스케줄링 결정을 맡고 있는 마스터 서버가 일을 못 하면 **전체 시스템 성능을 저하할 수 있는 병목**이 될 수 있다.

2. **대칭 다중 처리 (Symmetric MulitiProcessing, SMP)**

   거의 모든 최신 운영체제는 SMP를 사용한다. 대칭 다중 처리는 각 프로세서는 스스로 스케줄링을 할 수 있다. **각 프로세서의 스케줄러가 준비 큐를 검사**하고 **실행할 스레드를 선택**하여 스케줄링이 진행된다.

   스케줄 대상이 되는 스레드가 담기는 준비큐 전략

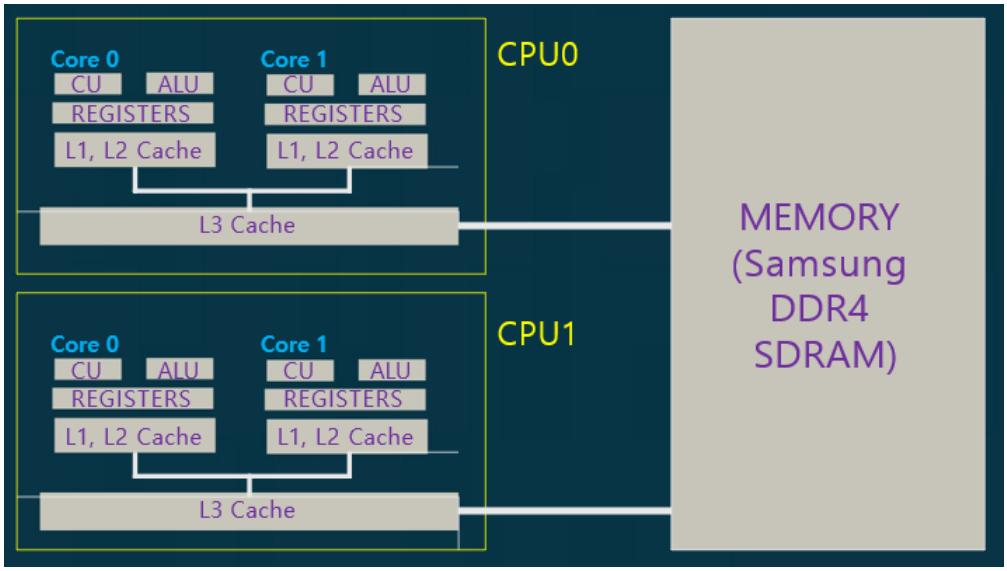
   - 공통 준비 큐
   - 각 프로세서별로 스레드 큐

다중 코어 프로세서

CPU는 프로세서 중 가장 중요한 역할을 하는 하드웨어이다. 이 **CPU 안에는 물리적인 계산**을 **실제로 진행하는 유닛을 코어(Core)**라고 한다.

CPU 안에는 한개 이상의 코어가 들어갈 수 있고 최근 하드웨어들은 **점점 많은 코어**를 탑재하는 추세이다.

한 프로세서에 여러 코어가 장착되어진 것을 다중 코어 프로세서라고 한다. **각 코어는 구조적인 상태를 유지하고 있어서 운영체제 입장에서는 개별적인 논리적 CPU처럼 보이게 된다.**

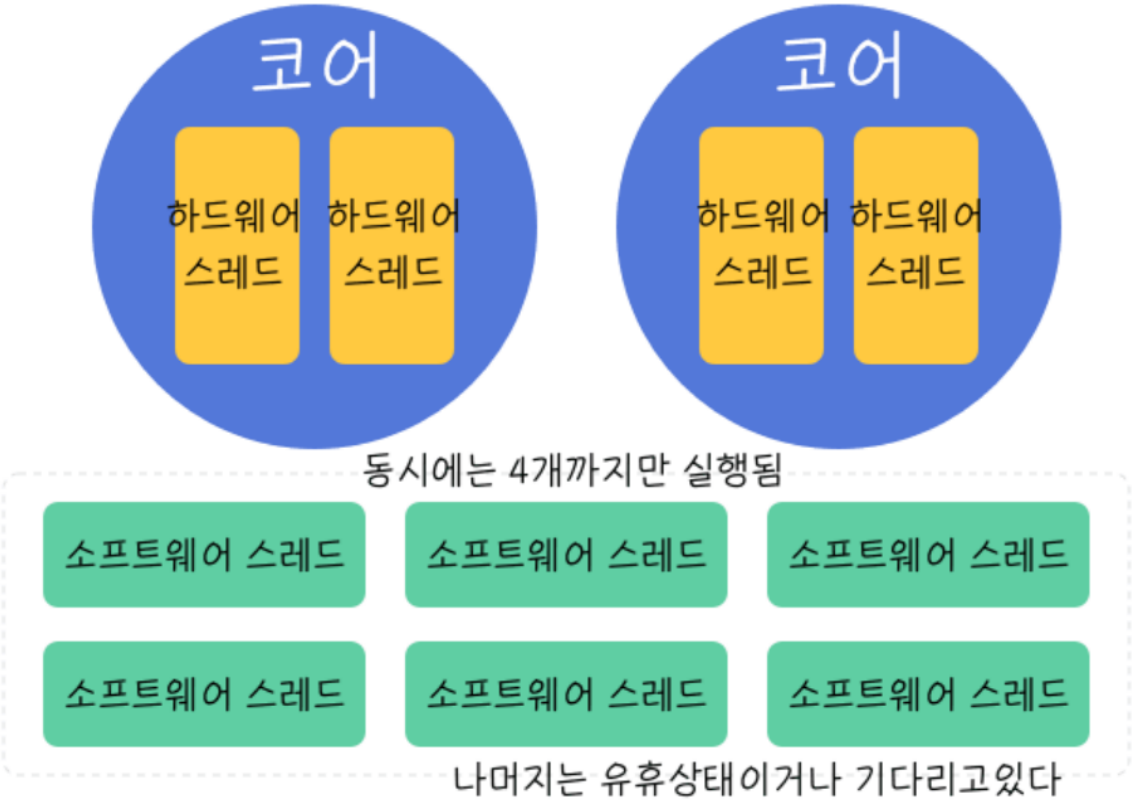


출처 : https://velog.io/@kanamycine/멘토링-1.-멀티코어-프로세서

하드웨어 스레드와 소프트웨어 스레드

코어안에는 **하드웨어 스레드**가 장착되어 있는데 이 또한 각 코어에 여러개의 하드웨어 스레드가 장착될 수 있고, 이는 **멀티 스레드 처리 코어**라고 말한다.

- 소프트웨어 스레드 → 우리가 계속 말하던 스레드
- 하드웨어 스레드 → OS가 스케줄 해줄 수 있는 최소 단위의 일



출처 : https://juneyr.dev/thread

이 하드웨어 스레드는 명령어 포인터 및 레지스터 집합과 같은 **구조적 상태를 유지**한다.

그래서 운영체제 입장에서는 소프트웨어 스래드를 실행할 수 있는 논리적 CPU로 보인다. 이를 **칩 다중 스레딩(Chip Multi-Threading, CMT)**이라고 한다.

아래 그림은 각 코어에 2개의 하드웨어 스레드가 있다. 운영체제 입장에서는 8개의 CPU가 있는 것으로 인지한다.

하나의 코어는 오직 하나의 스레드만 실행할 수 있다. 그래서 코어속 여러개의 하드웨어 스래드는 사실 동시에 실행되는 **병렬성**을 가지는 것이 아닌 시간에 따라 교대로 실행되어 동시에 실행되는 것처럼 보이는 **동시성**을 띄고 있다.

메모리를 기다리는 동안 하나의 하드웨어 스레드가 중단되면 코어가 다른 스레드로 전환할 수 있다.

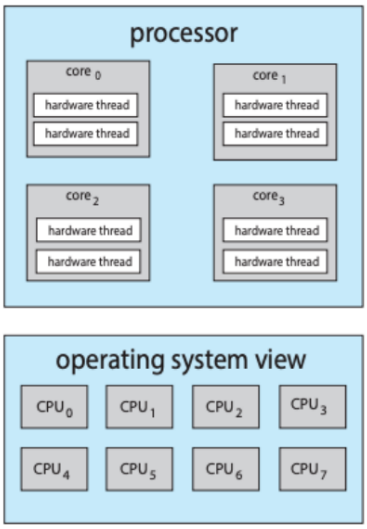


Figure 5.14 Chip multithreading.

다중 스레드화

일반적으로 프로세서가 다중 스레드화 하는 데에는 2가지 방법이 있다.

1. **거친 다중 스레딩 (coarse-grained)**

   매 사이클이 아닌 메모리 스톨과 같은 긴 지연시간을 가진 이벤트가 발생하면 그때 스래드를 변경한다.

2. **세밀한 스레딩 (fine-grained)**

   매 사이클마다 스레드를 변경한다.

결과적으로 다중 스레드 다중 코어 프로세스는 아래 그림과 같이 두 개의 다른 스케줄링 단계가 필요하다

1. 하드웨어 스래드에 어떤 소프트웨어 스레드들을 실행시킬 지 결정
2. 각 코어가 실행할 하드웨어 스래드 결정

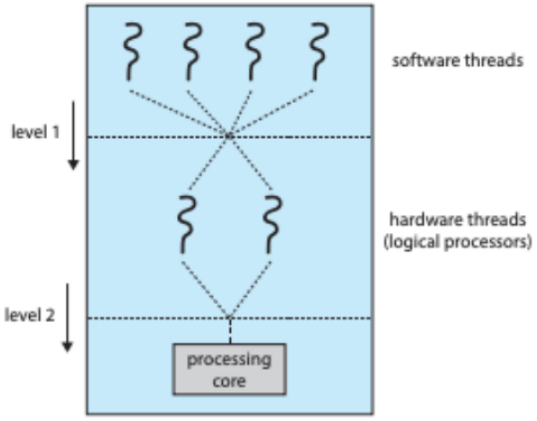


Figure 5.15 Two levels of scheduling.

부하 균등화 (Load Balancing)

SMP 시스템 중 각 프로세서가 실행할 스레드를 결정하기위해 각자 준비 큐를 가지고 있는 시스템의 경우에는 **부화 균등화 기능**이 필요하다.

부화 균등화는 SMP 시스템이 모든 프로세서 사이에 부하가 고르게 배분되도록 도와주는 것이다. 이를 위한 방법으로는 2가지가 있다.

1. **push migraion**

   특정 태스크가 주기적으로 각 프로세서의 부하 여부를 검사하고 과부하인 처리기에서 덜 바쁜 처리기로 스레드를 이동(push) 시키는 방식

2. **pull migration**

   쉬고 있는 프로세서가 바쁜 프로세서를 기다리고 있는 프로세스를 가지고(pull)오는 방식

처리기 선호도 (Processor Affinity)

캐쉬 등의 리소스 관련 문제로 인해 스레드는 한 프로세서에서 다른 프로세서로 이주시키지 않고 대신 같은 프로세서에서 계속 실행시키면서 이용하려한다. 이를 **프로세서 선호도**라고 한다.

프로세서 선호도 형태

- **약한 선호도(soft affinity)** : 운영체제는 프로세스를 원하는 특정 처리기에 실행시키려고 시도를 하나 프로세스가 처리기 사이에 이주하는 것이 가능ㅎ다.
- **강한 선호도(hard affinity)** : 운영체제는 시스템 콜을 사용해서 자신이 처리될 특정 처리기를 명시한다.

많은 시스템에서 둘 다 모두 지원하고 있다.

위에서 말했던 Load Balancing 부하 균등화는 Processor Affinity 처리기 선호도를 상쇄한다.

동일한 프로세서에서 스레드를 계속 실행시키면 캐시 메모리에 있는 데이터를 활용할 수 있다. 하지만 부하 균동화를 통해 스레드를 이동시키면 이 이점이 사라진다.

그래서 자신의 문맥에 맞춰서 잘 활용하자

이기종 다중처리

모바일 시스템에서는 다중 코어 아키텍처가 채택되어 있지만, 일부 시스템은 전력 소비를 조장하는 기능을 포함하여 클록 속도 및 전력 관리 측면에서 차이가 나는 코어를 사용하여 설계된다.

big 코어는 고성능이지만 많은 에너지를 사용하여 대화형 응용 프로그램을 할당하고,

little 코어는 더 적은 에너지를 사용하지만 저성능이여서 백그라운드 배치 프로그램을 할당한다.

실시간 시스템 (Real-Time System)

실시간 시스템에서는 **작업 수행이 요청되었을 때 이를 제한된 시간 안에 처리해서** 결과를 내주어야한다.

이 제한된 시간안에 처리하는 방식에 따라 2가지 실시간 시스템으로 분류할 수 있다.

1. **연성 실시간 시스템 (Soft Real-Time System)**

실시간 프로세스가 실시간이 아닌 프로세스들에 우선권을 가진다는 것을 보장

하지만 이것이 스케줄 되는 시점에 대해서는 아무런 보장이 없다.

2. **경성 실시간 시스템 (Hard Real-Time System)**

   태스크를 반드시 마감시간까지 서비스를 받을 수 있게끔 보장한다.

   마감시간까지 서비스를 받지 못 하면 해당 태스크가 실행되지 않은 것과 같은 결과를 도출한다.

퀴즈

- 다중 스레드 다중 코어 프로세서의 스케줄 단계에 대해 설명하라
- 하드웨어 스래드와 소프트웨어 스래드를 설명하라
- 부하균등화와 처리기 선호도의 관계에 대해 설명하라

참조

[Operating System - Chapter 5] CPU 스케줄링

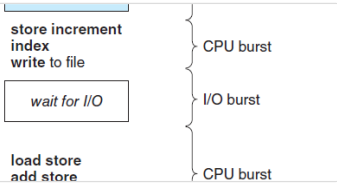
이 포스팅은 공룡책으로 알려진 Operating System Concepts의 5장인 CPU Scheduling를 공부하면서 정리한 포스팅이다.

https://imbf.github.io/computer-science(cs)/2020/10/18/CPU-Scheduling.html

[운영체제 - 공룡책] Chapter 05. CPU Scheduling

CPU 스케줄링은 멀티프로그래밍 운영체제의 기반이다. CPU의 프로세스들을 빈번히 switch하면서, 운영체제는 더 생산적이게 된다. 앞서 현대 운영체제들은 프로세스가 아닌 kernel-level단계의 스레드들을 스케줄링의 단위로 하고 있다. 하지만, "process

https://velog.io/@wilko97/운영체제-공룡책-Chapter-06.-Thread-Concurrency



하드웨어 스레드와 소프트웨어 스레드

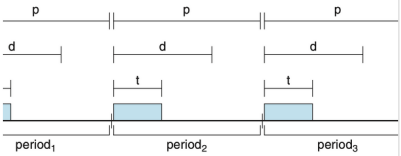
서론 notice : 모던 자바 인 액션 관련 글은 책의 저작권 이슈를 우려해서 삭제했습니다. CompletableFuture의 비동기 처리 를 공부하다가 비동기 작업을 다시 merge하는 작업, 즉 thenCompose 와 thenCombine…

https://juneyr.dev/thread

[Chapter 5. CPU 스케줄링] 실시간 시스템과 실시간 스케줄링 (RM, EDF)

* 본 글은 '운영체제(Operating System: Concepts) 9th edition'의 내용과 2021학년도 1학기에 수강한 '운영체제' 과목 강의 내용을 함께 정리하여 작성하였습니다. 실시간 시스템 (Real-Time System) 실시간 시스템에서는 작업 수행이 요청되었을 때 이를 제한된 시간 안에 처리하여 결과를 내주어야 한다. 이는 연성 실시간 시스

https://eunajung01.tistory.com/67



멀티코어 프로세서

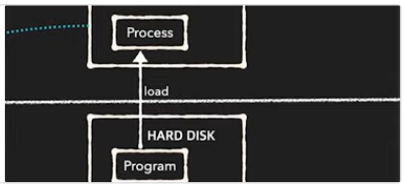
멀티 코어 프로세서 CPU는 두 개 이상의 독립 코어를 단일 집적회로로 이루어진 하나의 패키지로 통합한 것이다. dual-core, triple-core, hexa-core, octa-core, deca-core, dodeca-core (코어의 수 별)각 코어는 슈퍼스

https://velog.io/@kanamycine/멘토링-1.-멀티코어-프로세서

[CS]프로세스와 쓰레드, 멀티프로세스와 멀티쓰레드(+멀티코어)

프로그램 : 피자의 레시피와 같은 개념. 코드 파일로 구현되었고 실행되기 전엔 효력이 없다.프로세스 : 만들어진 피자와 같은 개념. 프로그램이 실행될때 생성됨.** 보조기억장치에 있던 프로그램이 실행되면서 메모리 공간에 할당됨.프로그램을 실행하면 파일은 운영체제에

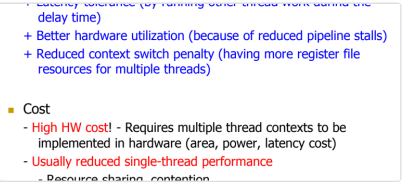
https://velog.io/@crosstar1228/CS프로세스와-쓰레드-멀티프로세스와-멀티쓰레드멀티코어



컴퓨터구조 16 - Multithreading

Thread Instruction stream with state Thread는 자기만의 독립적인 state(context)를 갖는다. 한 process안에 하나이상의 thread를 가질 수 있음. multi threading관점에서는 각각의 thread를 독립적인 program으로 생각한다. processor 입장에서 thread의 흐름(context)을 잘 기억해야된다. 이러한 문제를 해결하다보면 이전의

https://chopby.tistory.com/14



동시성(Concurrency) vs 병렬성(Parallelism)

서점에 가면 다양한 언어별로 동시성을 다루는 책들을 많이 볼 수 있습니다. 프로그래밍을 하다 보면 이러한 동시성 처리가 필요한 경우가 있습니다. 그런데 동시성이라는 말을 종종 병렬성과 혼동되어 사용하는 경우가 있습니다. 그래서 이번에 한번 정리를 해볼까 합니다. 동시성(Concurrency) vs 병렬성(Parallelism) 동시성병렬성

https://seamless.tistory.com/42