**IOCP**

접속자 수 만큼 스레드를 만드는 방식에서는 io처리를 위한 전담 스레드를 생성해야하고 스레드가 늘어나면 컨텍스트 스위칭이 많아져 비효율 적이다.

이런 단점을 극복하기 위해 하나의 스레드가 하나 이상의 사용자로부터 요구를 처리할 수 있도록 하게 해주는 방식인데 여러 개의 스레드를 사용하지만 두당 하나의 스레드가 필요한 방식은 아니라는 것이다.

IOCP란 특정 입출력 객체와 관련된 일종의 IO 작업 결과 큐라고 생각할 수 있는데 IOCP 객체가 별도로 생성되고 이 객체와 입출력 객체 주으이 하나가 연결되어 비동기 작업이 수행되면 운영체제가 이 큐에 그 비동기의 결과를 집어넣게 된다.

IOCP 객체 내부 thread pool을 사용하여 스레드의 생성 파괴 오버헤드를 없앰

적은 쉬의 스레드로 많은 연결을 관리

멀티스레드에서 동작하며 콜백함수들을 멀티스레드에서 동시에 동작해서 성능을 높이는 것이 iocp 성능의 비결

논블로킹 io

Select io

Overlapped io event, callback

**TCP vs UDP**

TCP는 데이터의 경계를 구분하지않으며 신뢰성 있는 데이터 전송을 보장합니다. 하지만 데이터 의 수신 여부를 확인해야 하기 때문에 전송속도가 느리다는 단점이 있습니다.

이에 반해 UDP는 데이터의 경계를 구분하고 신뢰성이 없는 데이터를 전송합니다.

그리고 데이터의 수신 여부를 확인 하지 않기 때문에 TCP보다 전송속도가 빠릅니다.

**Process & Thread**

* Process

1. 컴퓨터에서 연속적으로 실행되고 있는 컴퓨터 프로그램
2. 각각 독립된 메모리영역을 할당 받는다.
3. 기본적으로 프로세스당 최소 1개의 스레드를 가지고 있다.
4. 각 프로세스는 별도의 주소 공간에서 실행되며, 한 프로세스는 다른 프로세스의 변수나 자료구조에 접글할 수 없다.

* Thread

1. 스레드는 프로세스 내에서 각각 스택만 따로 할당받고 코드,데이터,힙 영역은 공유한다.
2. 스레드는 한 프로세스 내에서 동작되는 여러 실행의 흐름으로, 프로세스 내의 주소 공간이나 자원들을 같은 프로세스 내에 스레드끼리 공유하며 실행된다.
3. 같은 프로세스 안에 있는 여러 스레드들은 같은 힙공간을 공유하지만 프로세스는 다른 프로세스의 메머리에 직접 접근할 수 없다.
4. 각각의 스레드는 별도의 레지스터와 스택을 갖고 있지만 힙 메모리는 서로 읽고 쓸 수 있다.
5. 한 스레드가 프로세스 자원을 변경하면 다른 이웃 스레드도 그 변경 결과를 즉시 볼 수 있다.

**멀티 프로세스 & 멀티 스레드**

* 멀티 프로세스

1. 하나의 응용프로그램을 여러 개의 프로세스로 구성하여 각 프로세스가 하나의 작업을 처리하도록 하는 것

* 장점

여러 개의 자식 프로세스 중 하나에 문제가 발생하면 그 자식 프로세스만 죽는 것 이상으로 영향이 없다.

* 단점

컨텍스트 스위칭과정에서 캐쉬 메모리 초기화 등 무거운 작업이 짆애되고 많은 시간이 소모되는 등의 오버헤드가 발생한다.

프로세스는 각각의 독립된 메모리 영역을 할당 받았기 때문에 프로세스 간에 공유하는 메모리가 없어 컨텍스트 스위칭이 발생하면 캐쉬에 있는 모든 데이터를 모두 리셋하고 다시 캐쉬 정보를 불러와야한다.

프로세스는 각각의 독립된 메모리 영역을 할당 받았기 때문에 하나의 프로그램에 속하는 프로세스들 사이의 변수를 공유할 수 없다.

**멀티 프로세스 & 멀티 스레드**

* 멀티 스레드

하나의 응용프로그램을 여러 개의 스레드로 구성하고 각 스레드로 하나의 작업을 처리하도록 하는 것

* 장점

프로세스를 생성하여 자원을 할당하는 시스템 콜이 줄어들어 자원을 효율저긍로 관리할 수 있다.

스레드 간 데이터를 주고 받는 것이 간단하고 자원소모가 줄어든다.

스레드 간 작업량이 작아 컨텍스트 스위칭이 빠르다.

간단한 통신 방법으로 인해 프로그램 응답시간이 단축된다.

스레드는 프로세스 내의 스택 영역을 제외한 모든 메모리를 공유하기 때문에 통신의 부담이 적다.

* 단점

디버깅이 까다롭다.

단일 프로세스 시스템의 경우 효과를 기대하기 어렵다.

다른 프로세스에서 스레드를 제어할 수 없다.

멀티 스레드의 경우 자원 공유의 문제가 발생한다.(동기화 문제)

하나의 스레드에 문제가 발생하면 전체 프로세스가 영향을 받는다.

**Thread - Safe**

스레드 세이프란 여러 스레드가 동시에 사용되어도 안전하다는 것을 의미하며 특정 함수나 변수가 여러 스레드에서 호출되어도 하나의 스레드에서 호출했을 때와 같은 결과가 보장되어야 한다.

만약 함수가 전역 변수를 참조하게 된다면 그 함수는 thread safe하지 않은 결과가 나올 수 있다.

* Thread-safe 방식

1. 어떤 함수가 한 스레드에 의해 호출되어 실행 중일 때, 다른 스레드가 그 함수를 호출하더라도 그 결과가 각각에게 올바로 주어져야 한다.
2. 공유 자원의 사용을 최대한 줄여 각각의 스레드에서만 접근 가능한 저장소들을 사용하여 동시 접근을 막는다.
3. 공유 자원을 꼭 사용해야 할 경우 해당 자원의 접근을 세마포어 등의 락으로 통제한다.

**Context Switching**

멀티 프로세스 환경에서 cpu가 어떤 하나의 프로세스를 실행하고 있는 상태에서 인터럽트 요청에 의해 다음 우선 순위의 프로세스가 실행되어야 할 때 기존 프로세스의 상태 또는 레지스터의 값을 저장하고 cpu가 다음 프로세스를 실행하도록 프로세스의 상태 또는 레지스터의 값을 교체하는 작업을 컨텍스트 스위칭이라고 한다.

* 발생 시기

1. 입출력을 요청
2. CPU 사용시간이 만료
3. 자식 프로세스를 만들 때
4. 인터럽트 처리를 기다릴 때

와 같은 인터럽트 요청이 와야한다.

* 프로세스가 스레드보다 컨텍스트 스위칭에 더 많은 비용이 드는 이유

스택 영역을 제외한 모든 메모리를 공유하기에 컨텍스트 스위칭 발생 시에 스택 영역만 변경을 진행하면 된다.

프로세스는 공유하는 메모리가 없기에 캐시 메모리 초기화 등의 무거운 작업을 해야한다.

**동기 & 비동기**

* 동기

요청을 보낸 후 응답을 받아야지만 다음 동작이 이루어지는 방식이다.

설계가 매우 간단하고 직관적이지만 결과가 주어질 때까지 대기해야 한다는 단점이 있다.

* 비동기

요청을 보낸 후 응답과는 상관없이 다음방식이 동작하는 방식이다.

동기보다 복잡하지만 결과가 주어지는데 시간이 걸리더라도 그 시간 동아 ㄴ다른 작업을 할 수 잇으므로 자원을 효율적으로 사용할 수 있다는 장점이 있다.