typora-copy-images-to: ..\img

Blocking/Non-Blocking, SYNC/ASYNC

개념

Synchronous

- 요청과 그 결과가 동시에 일어난다는 뜻이며, 어떤 객체 또는 함수 내부에서 다른 함수를 호출했을 때 이 **함수의 결과를 호출한 쪽에서 처리**하면 동기입니다.

```
Scanner sc = new Scanner(System.in);
int num = sc.nextInt();
```

Asynchronous

- 요청과 그 결과가 동시에 일어나지 않는다는 뜻이며 동기와 달리 어떤 객체 또는 함수 내부에서 다른 함수를 호출했을 때 이 **함수의 결과를 호출한 쪽에서 처리하지 않으면 비동기**입니다.
-

```
setTimeout( foo, 3000);

function foo(){
    console.log("2");
}
console.log("1");
```

- 여기서 foo()함수를 setTimeout()함수의 **callback**함수라 부른다.
- **콜백 함수**: 비동기 방식에서 **어떤 Event(foo)가 발생 했을 때 수행해야 할 함수**를 의미.
- 이처럼 비동기 방식에서는 함수를 호출한 쪽에서 수행 결과를 직접 처리하지 않고 콜백 함수를 통해 수행 결과를 처리한다.

Blocking

- **블로킹**은 자신의 수행결과가 끝날 때까지 제어권을 갖고 있는 것을 의미한다.
    - 해당 프로세스의 작업은 중단되고 **WAIT**상태가 된다.

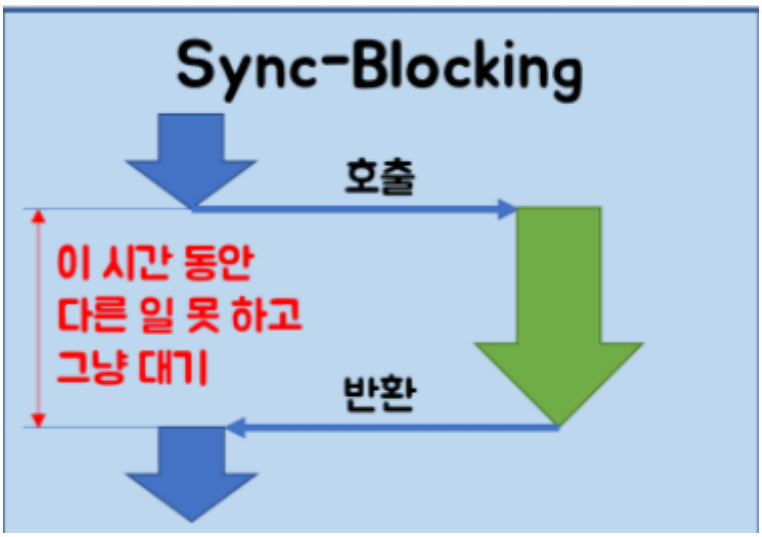
Non-Blocking

- **논블로킹**은 자신이 호출되었을 때 제어권을 바로 자신을 호출한 쪽으로 넘기며, 자신을 호출한 쪽에서 다른 일을 할 수 있도록 하는 것을 의미합니다
    - 해당 프로세스의 가 호출하면 바로 응답을 준다. (작업이 끝났는지 여부와 관련없이)

사용

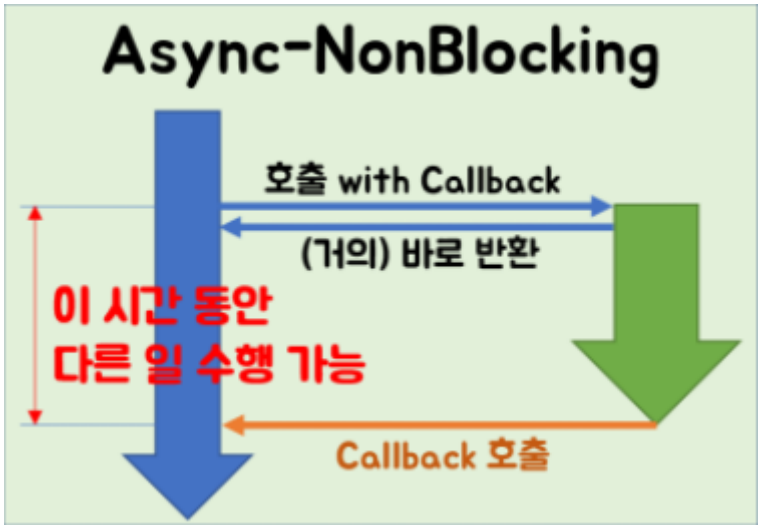
	Blocking	Non-blocking
Synchronous	Read/write	Read/wirte (O_NONBLOCK)
Asynchronous	i/O multiplexing (select/poll)	AIO

Read/Write

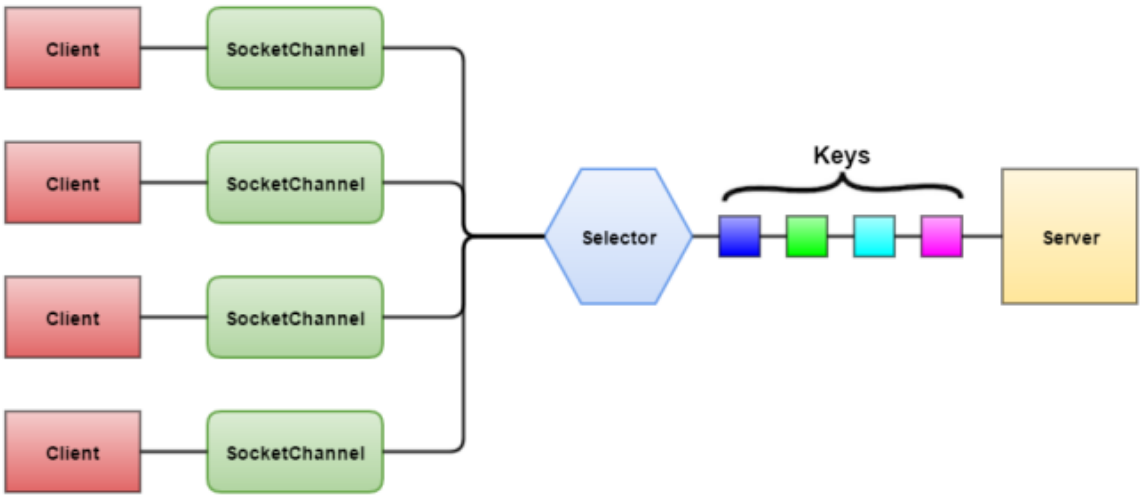


-

AIO



IO multiplexing (sellect / poll)



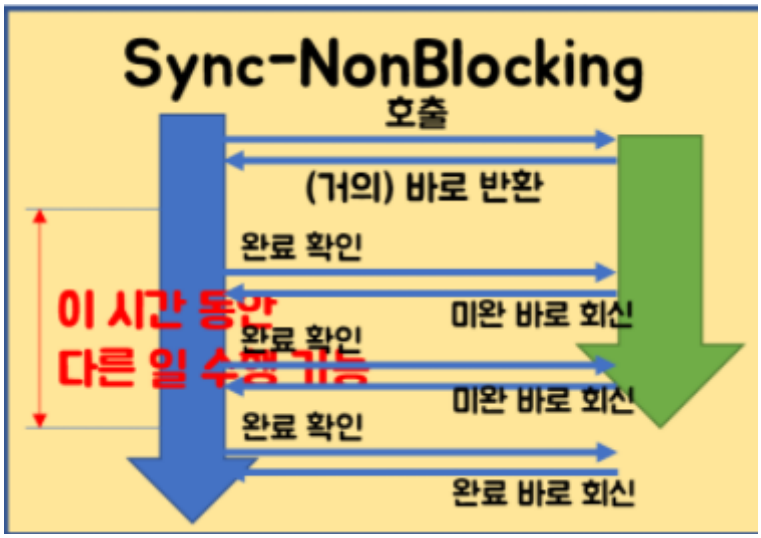
- **select**
  - 등록된 file descriptor를 하나하나 체크를 해야하고 커널과 유저 공간 사이에 여러번의 데이터 복사가 있음.
  - 관리 file descriptor 수에 제한이 있음.
  - 사용 쉽고 지원 OS가 많아 이식성 좋음.

  file descriptor를 하나 하나에 체크하기 때문에 O(n)의 계산량이 필요합니다. 따라서 관리하는 file descriptor의 수가 증가하면 성능이 떨어진다.
  또한 관리 수가 한정되어 있기 때문에 그 수를 초과하면 사용할 수 없다.
- **poll**
  - 관리 file descriptor 무제한.
  - 좀더 low level의 처리로 system call의 호출이 select보다 적음. 이식성 나쁨.
  - 접속수가 늘어나면 오히려 fd당 체크 마스크의 크기가 select는 3bit인데 비해, poll은 64bit정도이므로 양이 많아지면 성능이 select보다 떨어짐.
- **epoll**
  - 관리 fd의 수는 무제한.
  - select, poll과 달리, fd의 상태가 kernel 에서 관리하므로 상태가 바뀐것만을 직접 통지 , fd_set 복사가 필요없음.
  - 일일이 fd 세트를 kernel 에 보낼 필요가 없음.
  - kernel이 fd를 관리하고 있기 때문에 커널과 유저스페이스 간의 통신 오버헤드가 대폭 줄어듬.

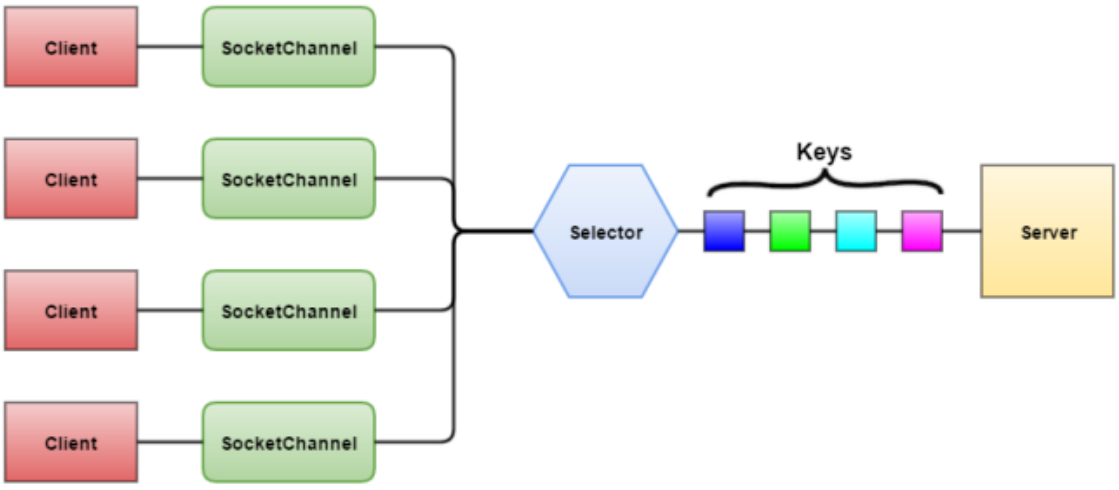
  epoll 모드는 활성 부분만을 염두에두고 순회 및 복사 작업을 줄입니다.

일종의 이벤트 리스너이다. 기존의 multi-threading 방식에서는 client 수가 증가하게 되면, 프 로그램의 성능이 급격히 낮아진다. (증가하는 thread로 인한 메모리, CPU 소비)이를 막기 위해 ?????를 사용하여, 하나의 thread에서 다수의 동시 사용자를 처리할 수 있도록 했다. Non-blocking 모드로 설정된 channel에 Selector를 등록해 놓으면 channel은 연결 요청이 들어오거나 data가 도착한 경우에 ?????에 알리게 된다. 메시지를 받은 ?????는 어떤 기능을 사용할 수 있는지 return하게 된다. 옵션으로 사용할 수 있는 기능은 다음과 같다.

Read/Write (O_NONBLOCK)



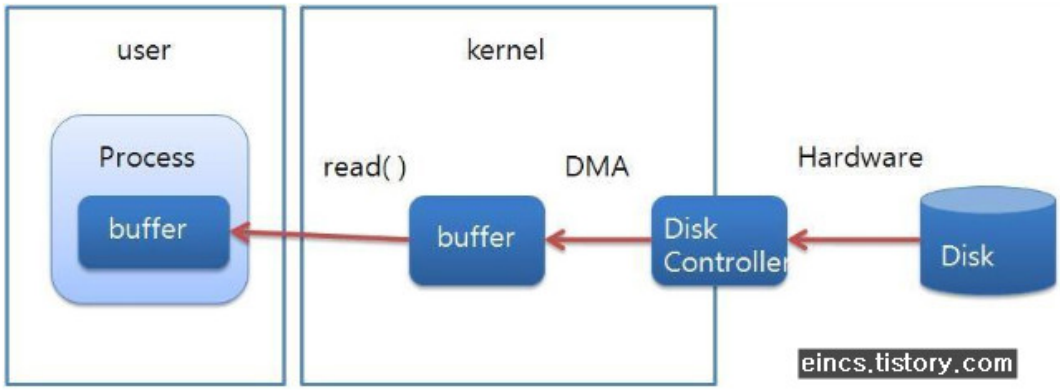
- 가장 노답.... syscall호출 장난 아님



JAVA NIO

기존 java IO 문제점

- IO는 커널 버퍼가 직접 핸들링 할 수 없다.



파일을 읽는 과정

```
Process(JVM)이 file을 읽기 위해 kernel에 명령을 전달
Kernel은 시스템 콜(read())을 사용함
디스크 컨트롤러가 물리적 디스크로 부터 파일을 읽어옴
DMA를 이용하여 kernel 버퍼로 복사
Process(JVM)내부 버퍼로 복사
```

이에 따른 오버헤드

```
JVM 내부 버퍼로 복사할 때, CPU가 관여
복사 Buffer 활용 후 GC 대상이 됨
복사가 진행중인 동안 I/O요청한 Thread가 Blocking
```

- Thread에서 블로킹이 발생
- Thread를 생성하는 시간
- 클라이언트에서 접속할때 thread생성, 접속할때 마다 블로킹 -> threadpool로 해결가능은 하지만 이것대로 클라이언트 수만큼 thread를 가지고 있는 자원낭비

NIO

커널 버퍼를 직접 핸들링 가능-> **DirectBuffer**

syscall을 간접사용하여 c, c++을 통한 Server Program이 가능하여 IO속도를 높일 수 있음. threadpool 문제를 OS에서 제공하는 select()를 통해 해결.

참조

https://victorydntmd.tistory.com/8

https://pjh3749.tistory.com/170

http://eincs.com/2009/08/java-nio-bytebuffer-channel-file/