

NIO, 비동기 IO 관련

🕒 Created	@Jan 24, 2021 9:40 PM
⋮ Property	
≡ Property 1	
⋮ Tags	

New IO

배경

- 기존 IO는 병목현상에 취약(queue 처리방식, 처리속도 < 데이터 진입 속도)
- NIO는 스레드 생성이 계속 발생하는 걸 막을 수 있음. but, 데이터 처리과정 자체가 어려우면 NIO에서도 병목 현상이 발생
- NIO 단점으로 버퍼를 어딘가에 생성해줘야함 → 대용량 데이터 처리시 그 크기만큼의 버퍼 생성이 필요해진다.

(re) Blocking VS Non-Blocking

- Blocking
 - 모든 스레드가 병렬 구조(계층 X)
 - 채널, 스트림 구성시 송수신 받기 위해 스레드를 계속 중지 상태로 두어야함
 - 주고 받는 데이터가 없을 경우 스레드가 멈춰있어 사용자 수에 맞게 스레드를 늘려줘야함 → 사용자가 많은 경우 스레드 풀의 스레드도 늘려줘야 해서 오버헤드 발생할 수 있음
 - NIO에서는 스레드를 인터럽트로 빠져나올 수 있음
(IO에서는 무조건 스트림을 닫아야지 블로킹 탈출함)
- Non-Blocking
 - 네트워크 소켓 채널은 NIO을 지원하지만 스레드가 하나인 이상 병렬 처리는 못함
→ 시간이 오래 걸리는 작업은 논블로킹에서도 별도 스레드 생성 필요
 - 여러 채널에서 발생하는 유형별 이벤트를 돌아가면서 처리함 → 하나의 스레드가 모든 스레드 IO를 관리하는 멀티플렉서 역할을 함.

NIO에서 버퍼

- 입출력 데이터를 항상 버퍼(Buffer)에 저장되고 데이터 타입에 따른 버퍼가 존재
- 버퍼형식은 크게 2가지로 분류
 - Direct buffer : OS의 메모리에 있고 크기가 크고 생성 시간이 느리나 성능이 높음
 - Non-direct buffer : JVM의 힙 메모리에 있고 크기가 작고 생성시간이 빠르나 성능이 낮음
- 버퍼를 읽고 쓰는 일반적인 프로세스 : 버퍼에 데이터 쓰기 → `buffer.flip()` → 버퍼에서 데이터 읽기 → `buffer.clear()` or `buffer.compact()`

IO모델

- 블로킹 : 애플리케이션 실행 시 운영체제 대기 큐에 들어가면서 요청에 대한 system call이 완료된 후에 응답을 보낸다.
- 논블로킹 : 애플리케이션 실행 시 운영체제 대기 큐에 들어가지 않고, 실행 여부와 관계없이 바로 응답을 보낸다. 바로 응답하기 힘든 경우, 에러를 반환하는데 정상데이터를 받을 때까지 계속해서 요청을 다시 보낸다.

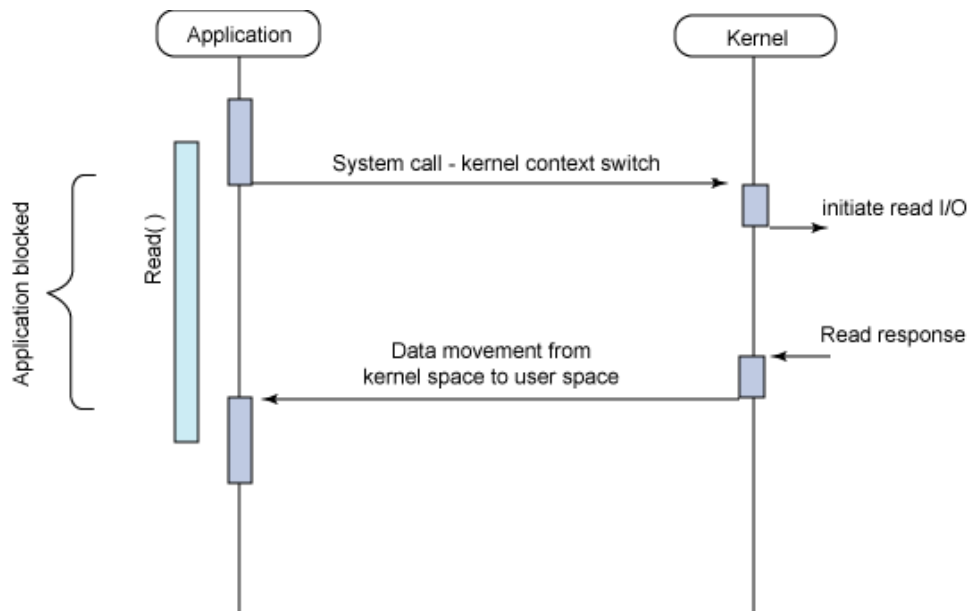
	Blocking	Non-blocking
Synchronous	Read/write	Read/write (O_NONBLOCK)
Asynchronous	i/O multiplexing (select/poll)	AIO

IO이벤트 통지모델은 논블로킹에서 제기된 문제를 해결하기 위해 고안되었다. IO 이벤트를 통지하는 방법은 크게 동기형 통지모델과 비동기형 통지모델로 나눌 수 있다.

- 동기 : 시스템콜을 기다린다. (notify를 사용자 프로세스가 담당) 시스템의 반환을 기다리는 동안 대기 큐에 머무는 것이 필수는 아니다. (블로킹은 필수로 머물러야함)

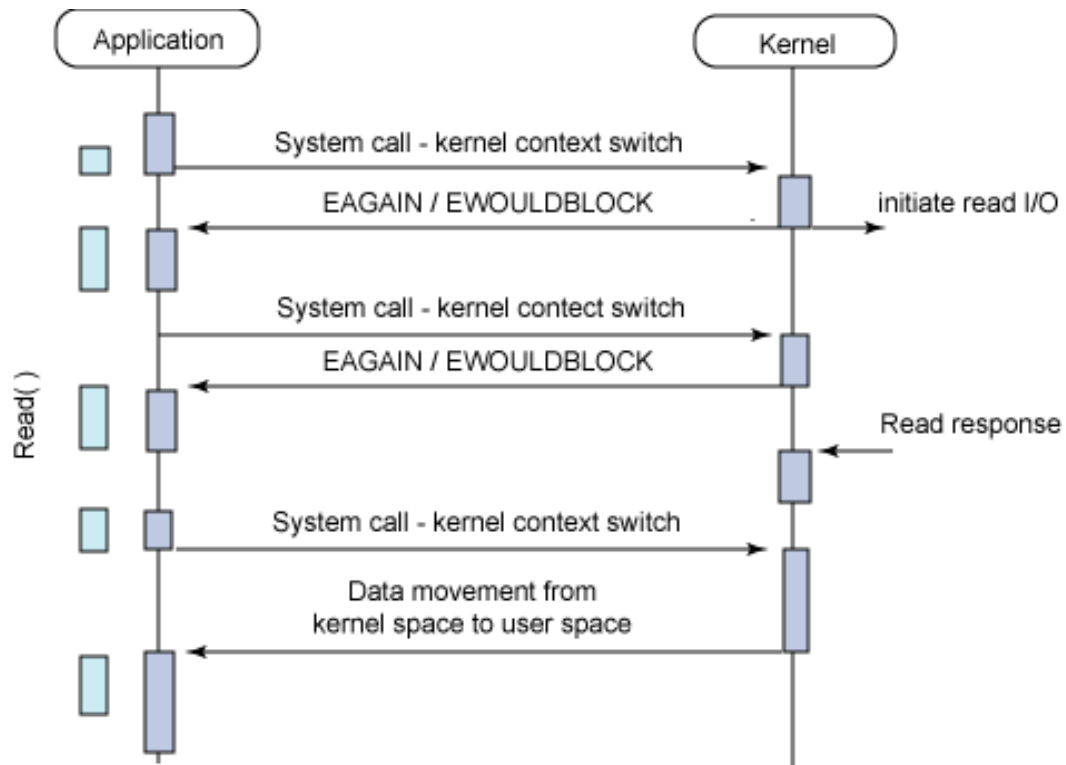
- 비동기 : 시스템콜을 기다리지 않는다. (notify를 커널이 담당) 요청에 대해 처리완료 여부에 관련없이 응답하고 다음코드를 돌린다. 이후에 운영체제에서 처리완료여부를 알려주고 응답한다.

동기 블로킹



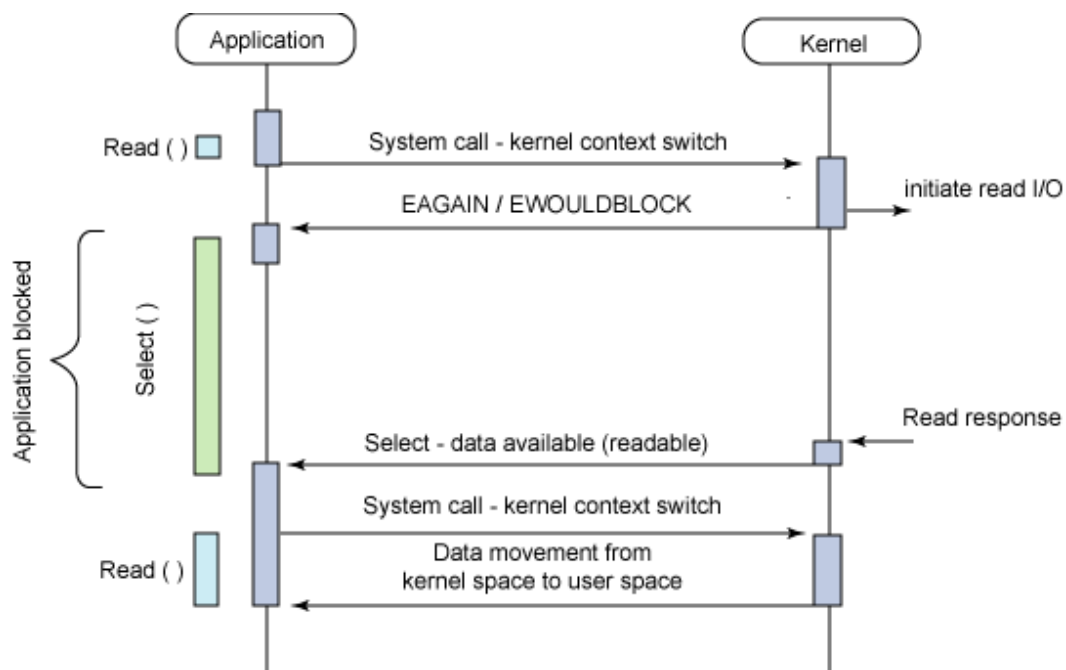
- 프로그램이 블로킹을 일으키는 시스템 함수를 호출
- 한 작업당 한 번의 사용자-커널사이의 문맥교환 발생
- 정지된 프로그램은 CPU를 사용하지 않고 커널의 응답을 대기
- 프로그램 관점에서 보면 마치 처리로직이 오래걸리는 것 같지만, 사실은 커널의 일을 기다리느라 블록되어 있는 것이다. 이게 개선 포인트

동기 논블로킹



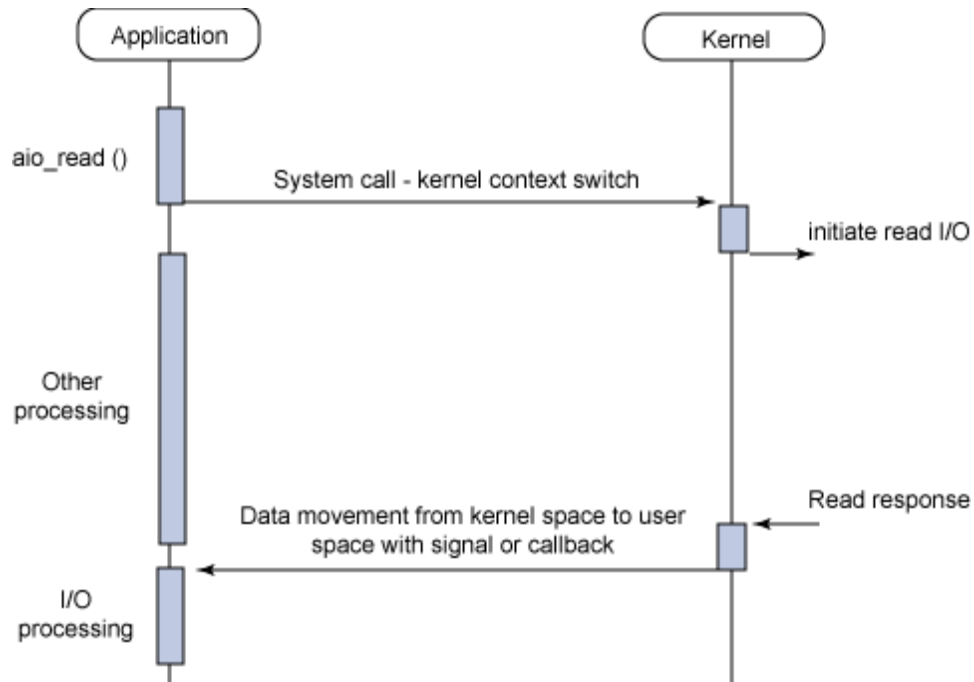
- 동기블로킹의 개선안이지만 비효율적이다. 왜냐하면 위에서 정리했듯이 논블로킹방식은 정상데이터가 올 때 까지 계속 시스템콜을 하며 문맥교환을 한다.
- IO 지연(latency) 초래한다.

비동기 블로킹



- IO는 논블로킹이고 알림(notify)가 블로킹인 방식이다.
- select() 시스템함수 호출이 사용자프로세스를 블로킹한다.
- 비효율적이다.

비동기 논블로킹



- 시스템콜이 즉시 IO개시 여부를 반환한다. 사용자프로세스는 다른일을 할 수 있고 (CPU는 다른 업무를 볼 수 있다), IO는 백그라운드에서 처리된다.
- IO 응답이 도착하면 신호나 콜백으로 IO전달을 완료한다.

※ 참고자료

- <https://adrian0220.tistory.com/144>
- <https://jongmin92.github.io/2019/03/03/Java/java-nio/>
- <https://sjh836.tistory.com/109>
- <https://ju3un.github.io/network-basic-1/>