

Spring Boot Seminar - 2

Wafflestudio Rookies seminar 2024


Sangmin Kim, 2024

Seminar - 2 목차


- Assignment 1 리뷰
- 동시성 처리 - Server 관점
- 동시성 처리 - Server Cluster 관점
- MySQL Transaction


Assignment 1 리뷰

```
21 + @RequestParam date: String,  
22 + @RequestParam mealType: MealType,  
23 + ): ResponseEntity<MealPlanResponse> {  
24 +     val mealPlan = mealPlanService.viewMealPlan(date, mealType)
```

 **sanggggg** 2 days ago Author ...

Date 와 같은 요청 문자열에 대한 파싱은 도메인 영역인 Service 보다 Controller 에 책임을 두는게 좋을 것 같습니다.



 Reply...

Resolve conversation

Assignment 1 리뷰

```
103 +         return allMenuWithReviews.sortedBy {  
104 +             if (it.reviews.isEmpty()) {  
105 +                 0.0  
106 +             } else {  
107 +                 it.reviews.map { review -> review.rating }.average()  
108 +             }  
109 +         }  
110 +     }
```



sanggggg 2 days ago

Author ...

데이터를 정렬하는 방법은 이것 처럼 서버단에서 소팅 하는 방법과, SQL 단에서 (~=RDB 단에서) 소팅하는 방법이 존재합니다.

- DB Sorting: DB Index 를 사용할 수 있어서 빠른 정렬 가능, 엄청 큰 데이터에 대한 소팅도 문제 없다. 서버 리소스를 적게 쓴다.
- Server Sorting: 엄청 큰 데이터는 서버 메모리를 많이 쓰면서 성능상 문제가 있을 수 있다. DB 리소스를 적게 쓴다.

보통 Index 를 태워서 정렬하는 DB 가 성능상 대부분 좋습니다. (Index 는 다음 세미나 설명 예정)

물론 특별한 케이스에서는 DB 부하를 최소화 하고 싶어서 서버 소팅을 할 수도 있지만...

DB 에서 정렬을 하도록 하는 방법은 아래와 같은 메서드를 레포지토리에 정의하면 사용할 수 있습니다.

```
interface MenuRepository : JpaRepository<MenuEntity, String> {  
  
    @Query("SELECT m FROM menus m LEFT JOIN m.reviews r GROUP BY m.id ORDER BY AVG(r.rating) DESC"  
    fun findAllOrderByRatingDesc(): List<MenuEntity>  
}
```

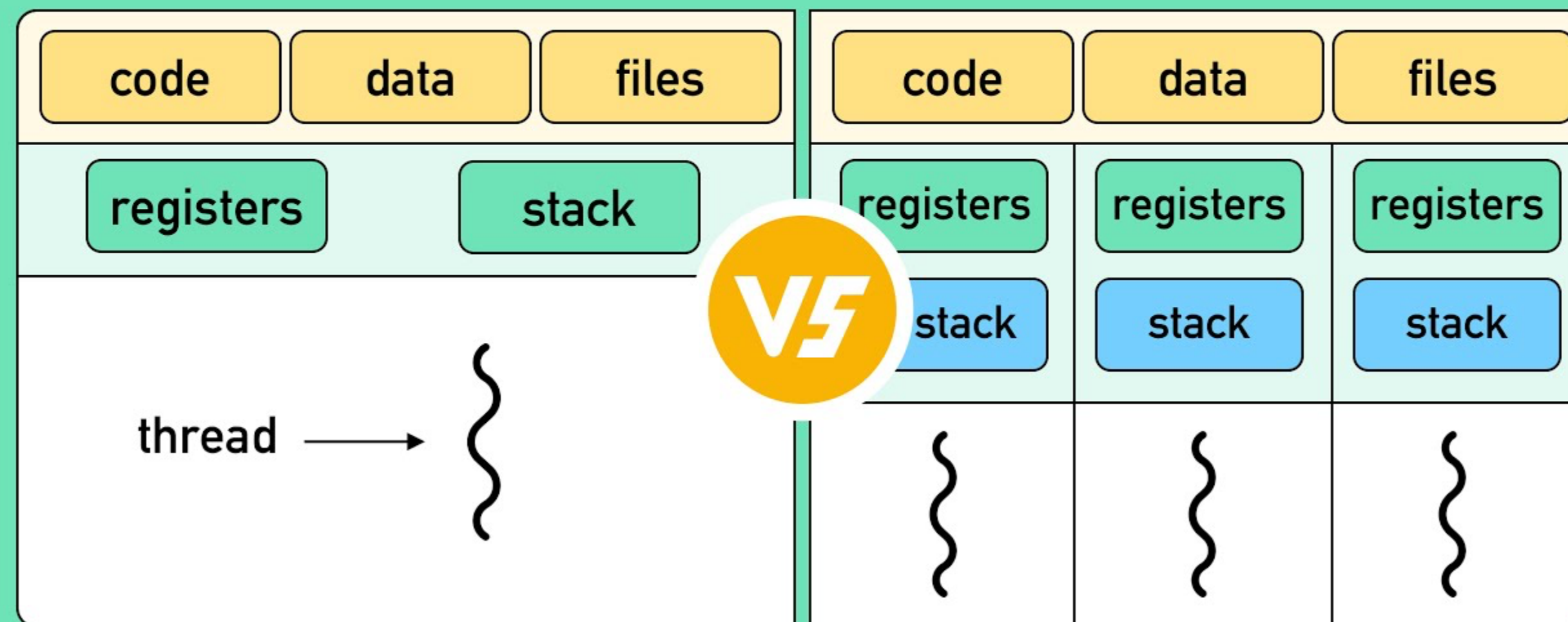


Reply...

동시성 처리 - Server 관점

쓰레드와 프로세스

| Process vs. Thread



동시성 처리 - Server 관점

멀티 쓰레딩

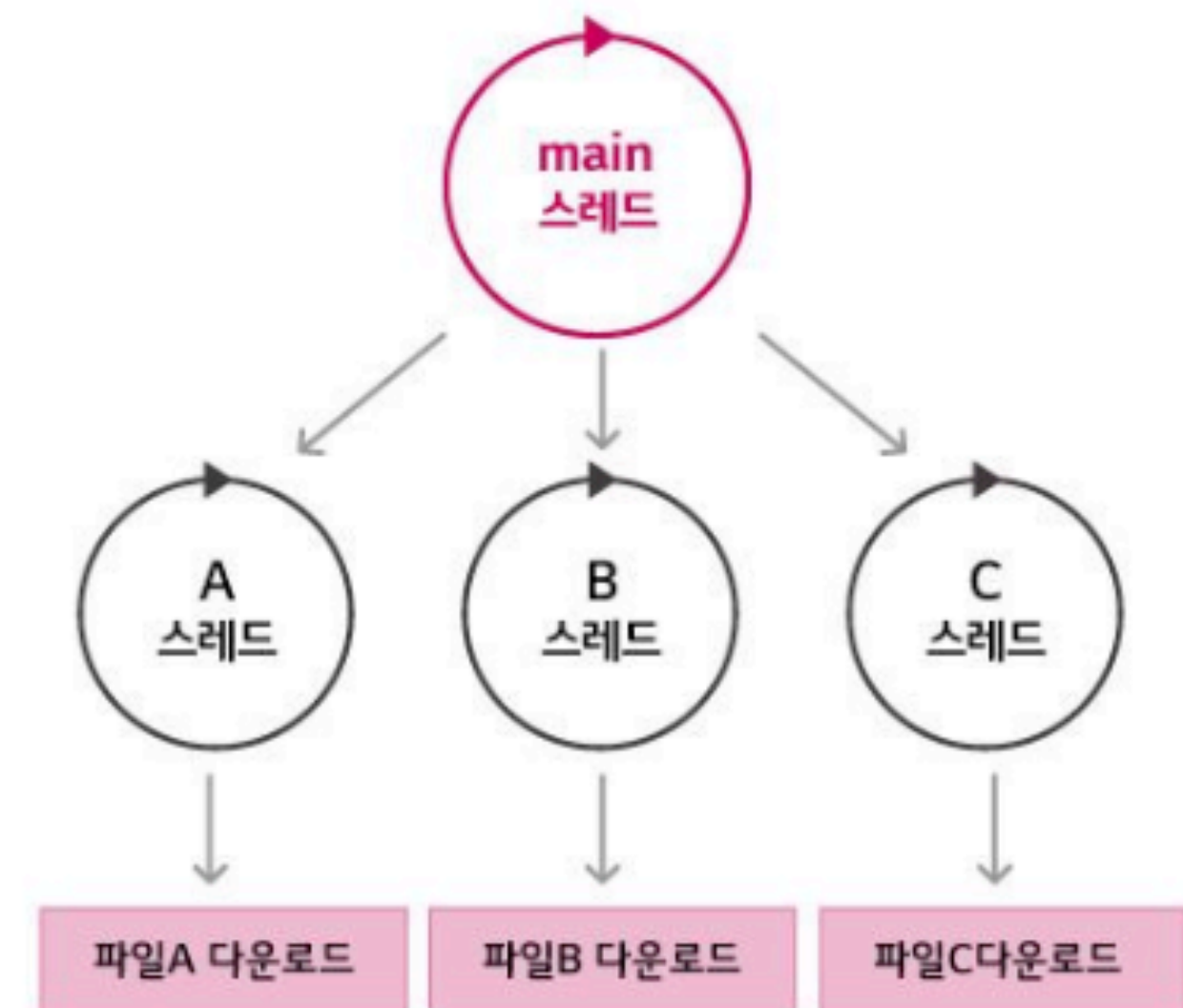
- 컴퓨팅 리소스의 효율적인 활용 (Multi-Core CPU)
- 블로킹 작업의 병렬 처리 (Network I/O)

싱글 스레드



순차 실행(Sequential)

멀티 스레드



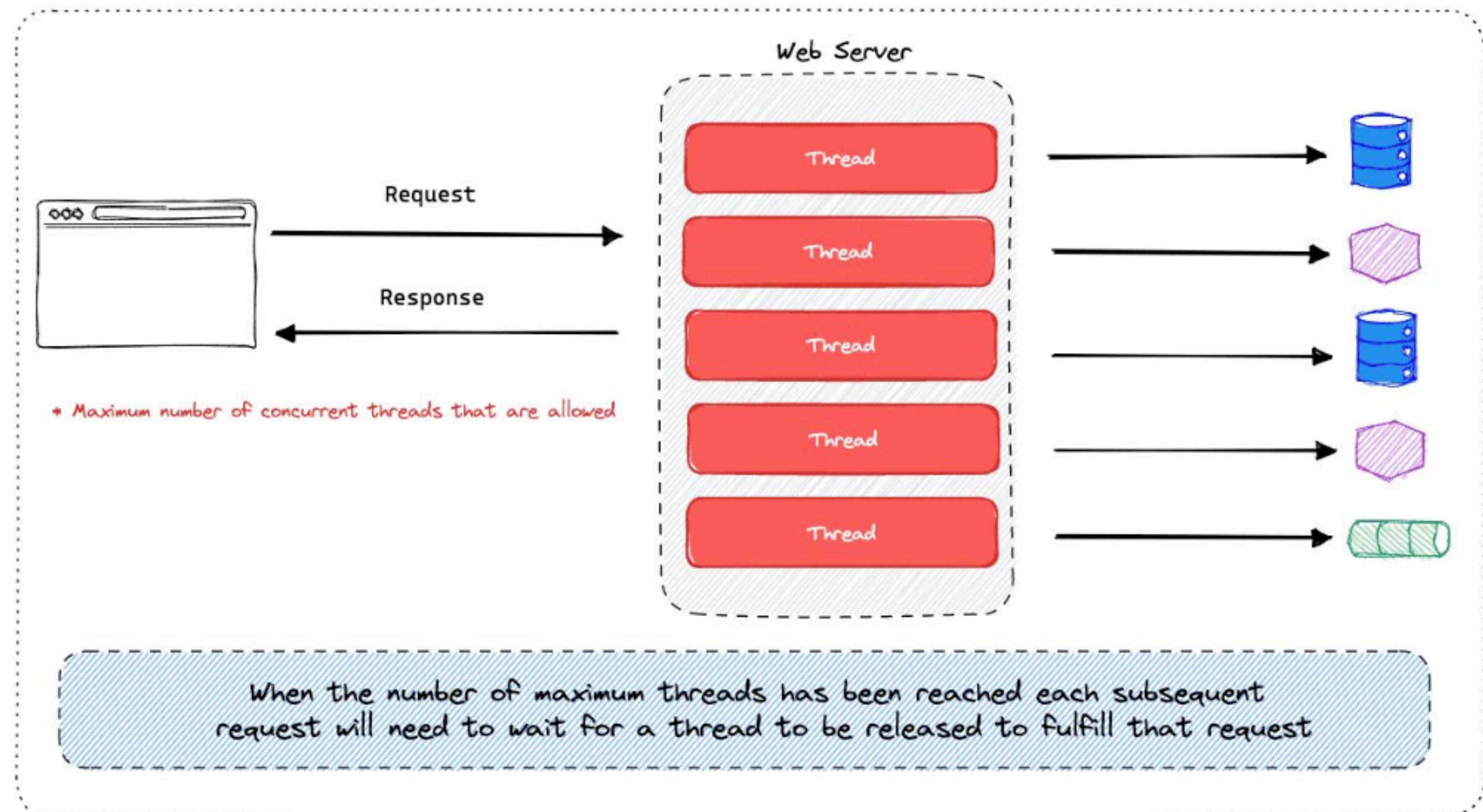
병행 실행(Concurrent)

동시성 처리 - Server 관점

스프링 MVC의 쓰레드 사용 방식

- 1 Thread / 1 Request (Tomcat)
- 필요에 의해 각 Request 에 대응하는 여러 쓰레드를 생성 가능

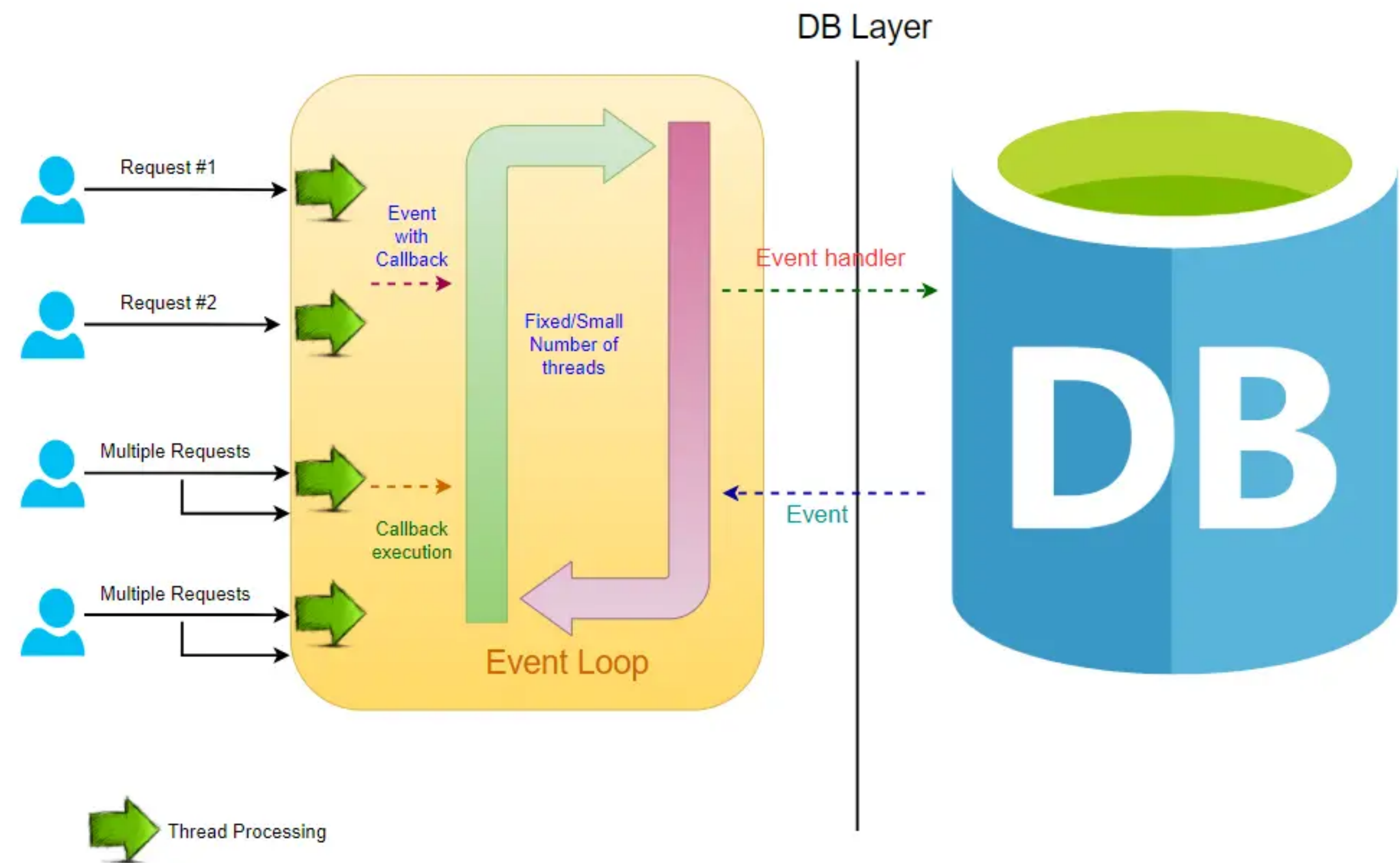
Thread Per Request



동시성 처리 - Server 관점

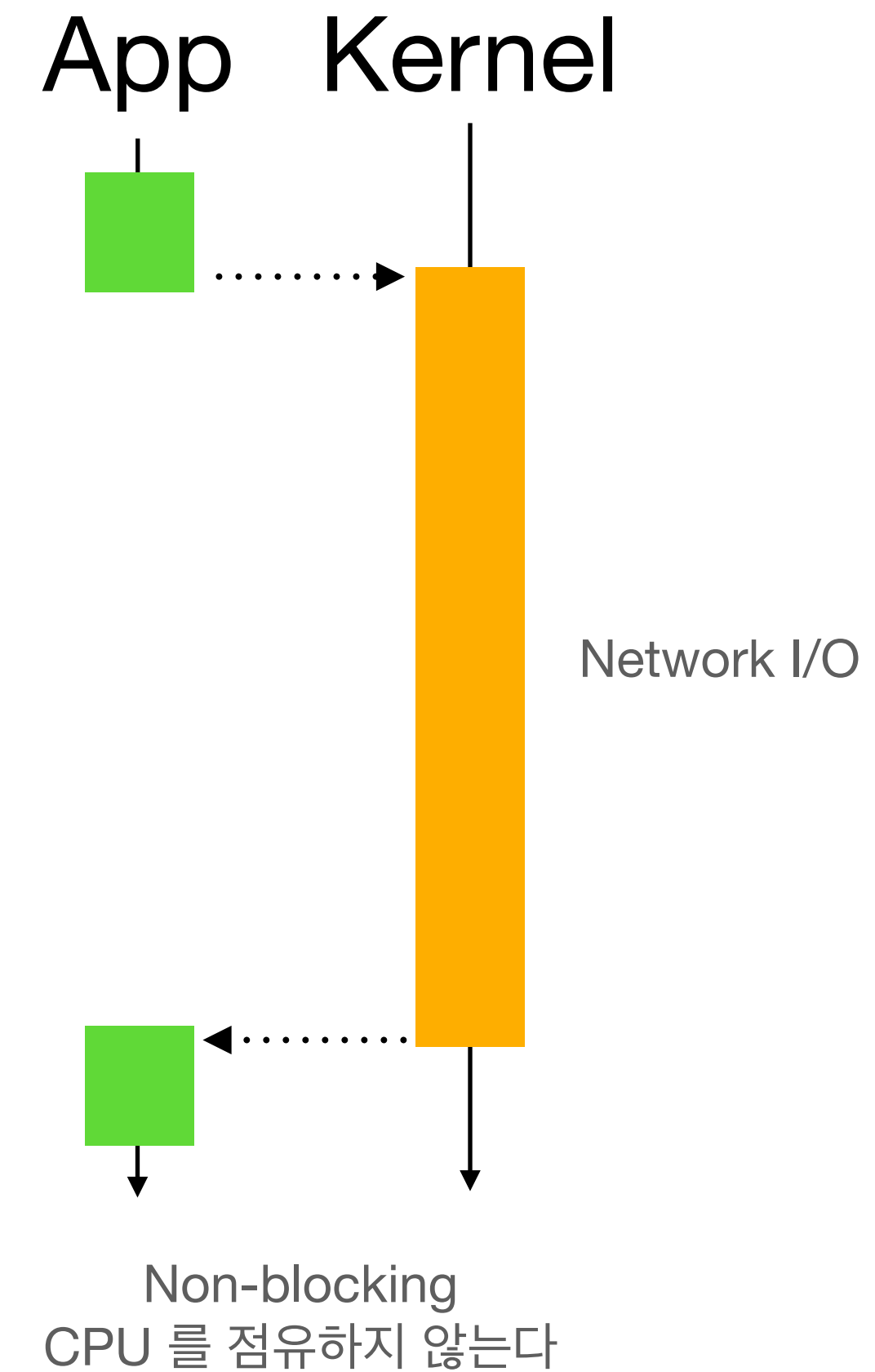
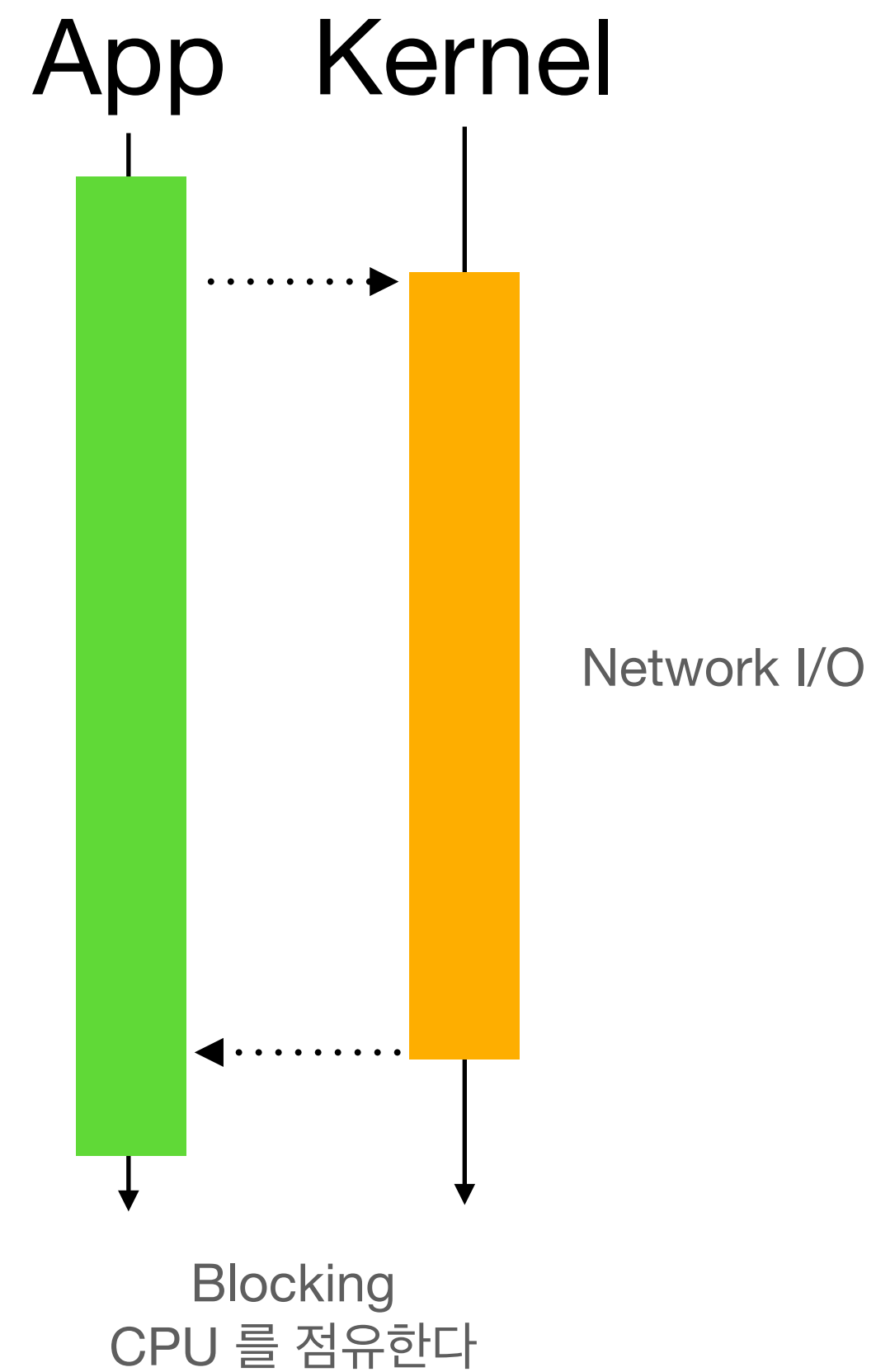
스프링 Webflux 의 쓰레드 사용 방식

- N Thread : M Request
- 작은 개수의 Thread 로 처리 가능
- I/O 로 인한 병목이 큰 경우 유용 (Non-blocking I/O)



동시성 처리 - Server 관점

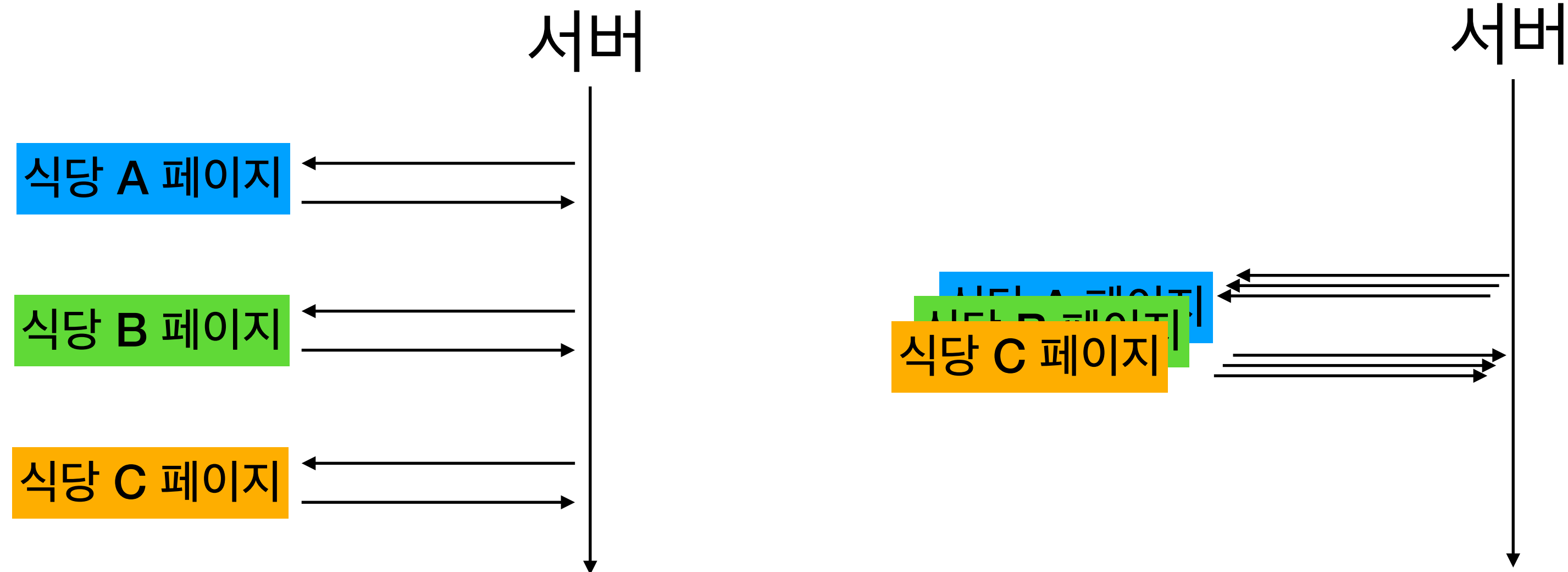
Non-blocking vs Blocking



동시성 처리 - Server 관점

동시 처리 예시

- 식단 크롤링 예시



동시성 처리 - Server 관점

예시 코드

```
private val threads = Executors.newFixedThreadPool( nThreads: 4)

@PostMapping("/api/v1/batch") new *
fun batchSyncMenus(): ResponseEntity<Unit> {
    val tasks = restaurants.map {
        threads.submit { fetchRestaurantMenus(it) }
    }
    val datas = tasks.map { it.get() }
    // menuService.syncMenus(datas.flatten())
    return ResponseEntity.noContent().build<Unit>()
}

@PostMapping("/api/v1/sequential") new *
fun sequentialSyncMenus(): ResponseEntity<Unit> {
    restaurants.forEach {
        fetchRestaurantMenus(it)
    }
    return ResponseEntity.noContent().build<Unit>()
}
```

```
@Test new *
fun test() {
    var start = System.currentTimeMillis()
    mvc.perform(
        MockMvcRequestBuilders
            .post( urlTemplate: "/api/v1/batch")
    ).andReturn()
    val durationBatch = System.currentTimeMillis() - start
    mvc.perform(
        MockMvcRequestBuilders
            .post( urlTemplate: "/api/v1/sequential")
    ).andReturn()
    val durationSequential = System.currentTimeMillis() - start - durationBatch
    println("Batch: $durationBatch ms")
    println("Sequential: $durationSequential ms")
}
```

✓ Tests passed: 1 of 1 test

Batch: 1071 ms

Sequential: 4021 ms

동시성 처리 - Server 관점

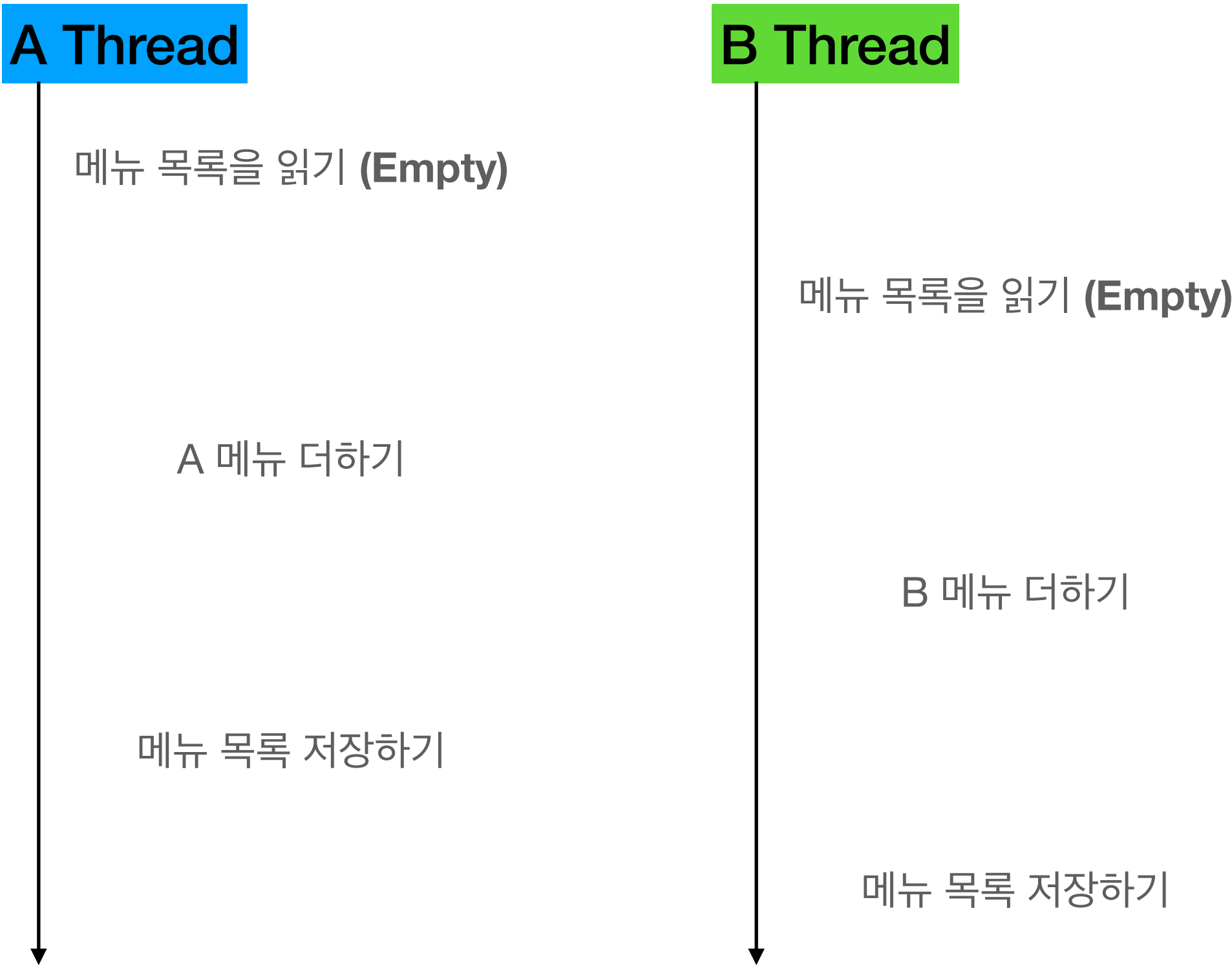
Sync 문제

- 동일한 리소스에 같이 접근한 경우
 - 저 판 뒤집어야지 (파란 면 올라오게 해야지)
 - 저 판 뒤집어야지 (파란 면 올라오게 해야지)
 - 결과물은 다시 빨간면



동시성 처리 - Server 관점

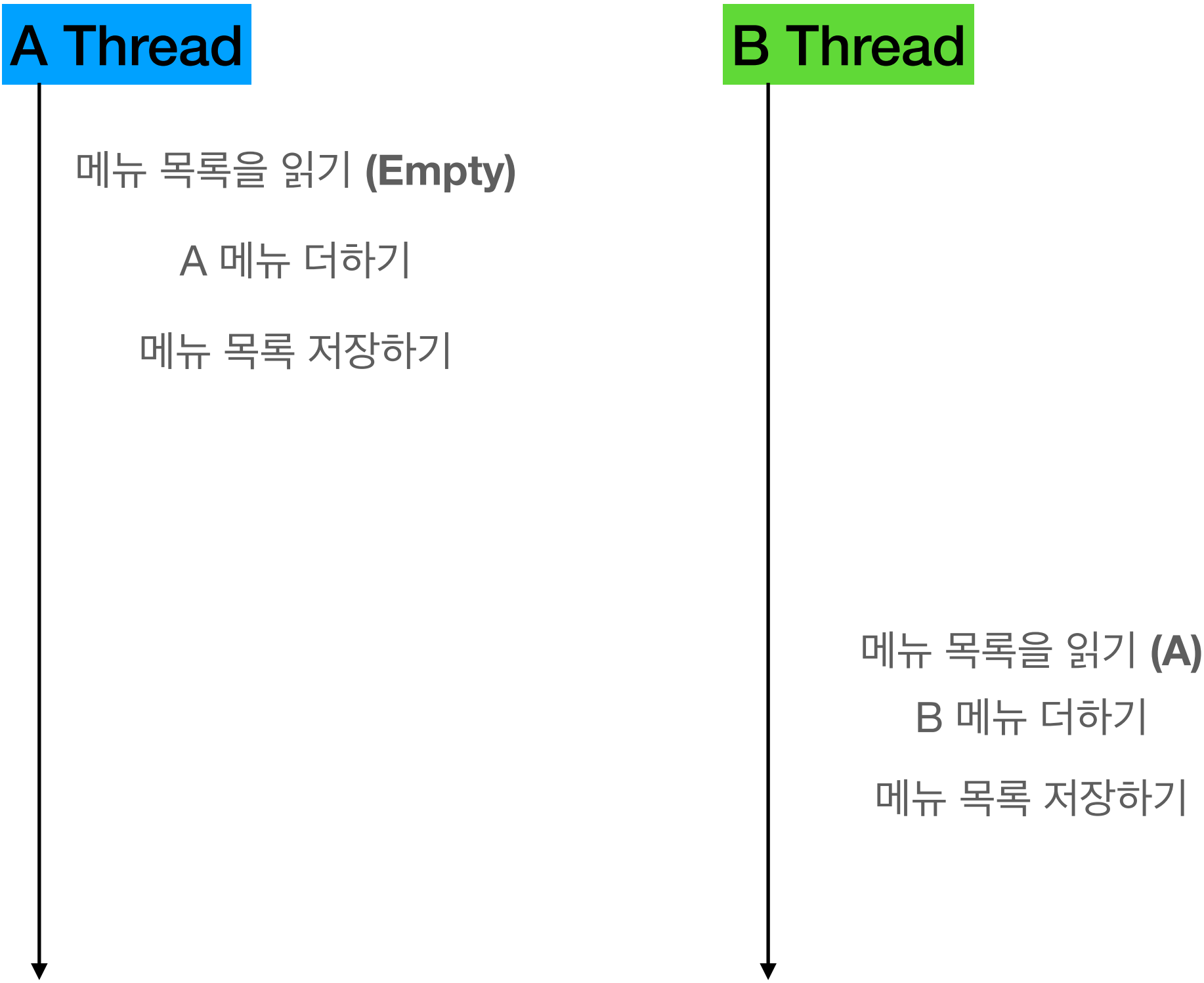
Sync 문제



최종 결과: B

동시성 처리 - Server 관점

Sync 문제



최종 결과: A, B

동시성 처리 - Server 관점

Sync 문제

- Lock 을 통한 독점적 접근이 필요
- Java의 `@Synchronized`
- 🤔 “서버” 가 아닌 코드라면 중요할지도...?
 - 그러면 서버는 뭐가 중요한가요?

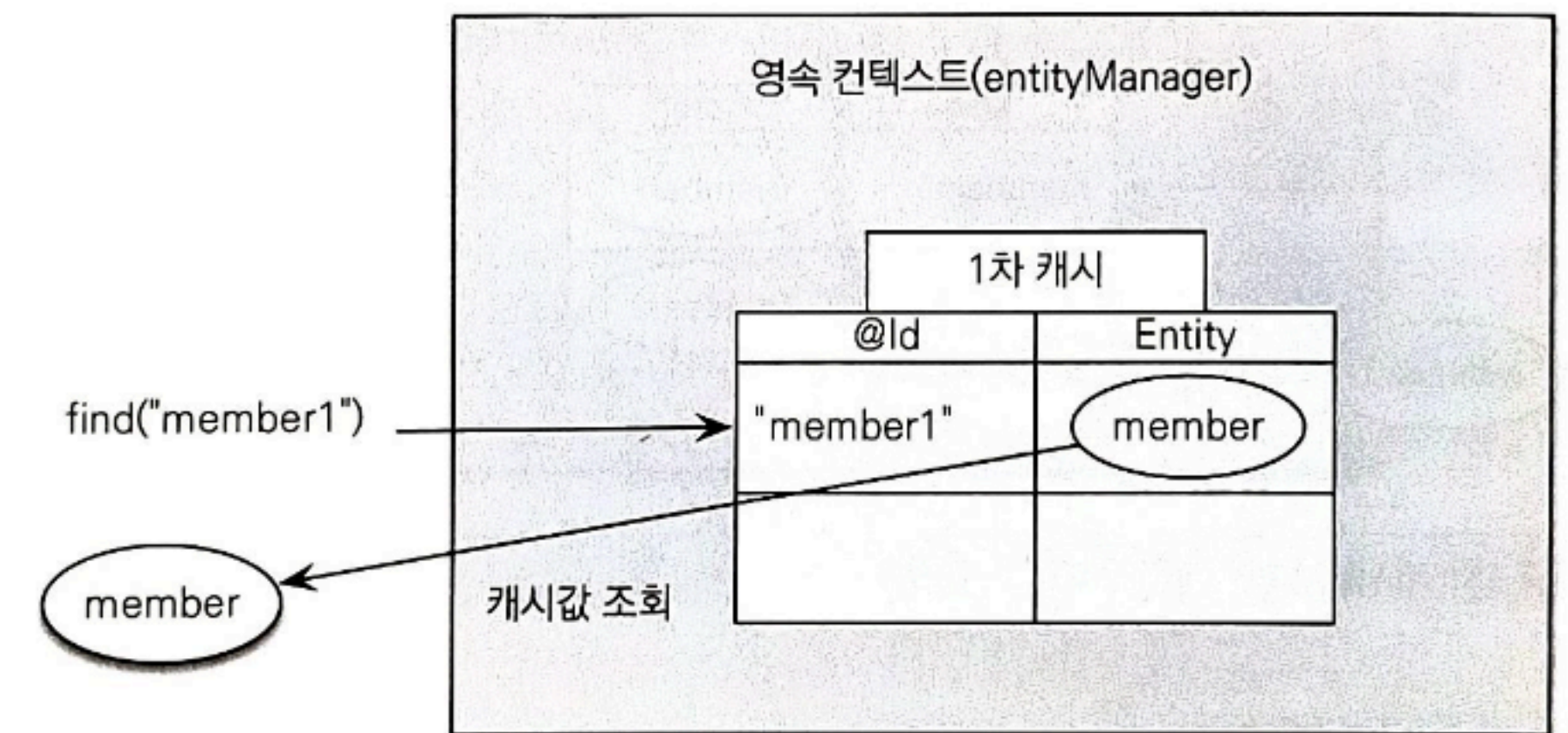
```
var pad = true

@synchronized new *
fun flipToFalse() {
    if (pad) {
        pad = !pad
    }
}
```

동시성 처리 - Server 관점

심화) JPA 와 Thread

- JPA 의 EntityManager (영속성 컨텍스트) 들은 Thread 에 묶이는 정보들이다. (각 요청마다 하나씩 생성된다)
- Why?



동시성 처리 - Server 관점

심화) JPA 와 Thread

- 멀티 쓰레드 상황에서 JPA 의 동작

```
private val threads = Executors.newFixedThreadPool( nThreads: 1)

@Transactional
@Test
fun `다른 스레드에서 조회한 엔티티를 가지고, 현재 스레드에서 영속성 컨텍스트 관련 기능을 사용할 수 없다`() {
    val songFromCurrentThread : SongEntity = songRepository.findById( id: 1L).get()

    assertDoesNotThrow {
        println(songFromCurrentThread.album.title) // 영속성 컨텍스트를 통해 album을 lazy load
    }

    val songFromTheOtherThread : SongEntity! = threads.submit<SongEntity> { songRepository.findById( id: 1L).get() }.get()

    assertThrows<LazyInitializationException> {
        println(songFromTheOtherThread.album.title) // 영속성 컨텍스트를 통해 album을 lazy load하려 하지만 다른 스레드(영속성 컨텍스트)에서 조회한 것이기 때문에 에러 발생
    }
}
```

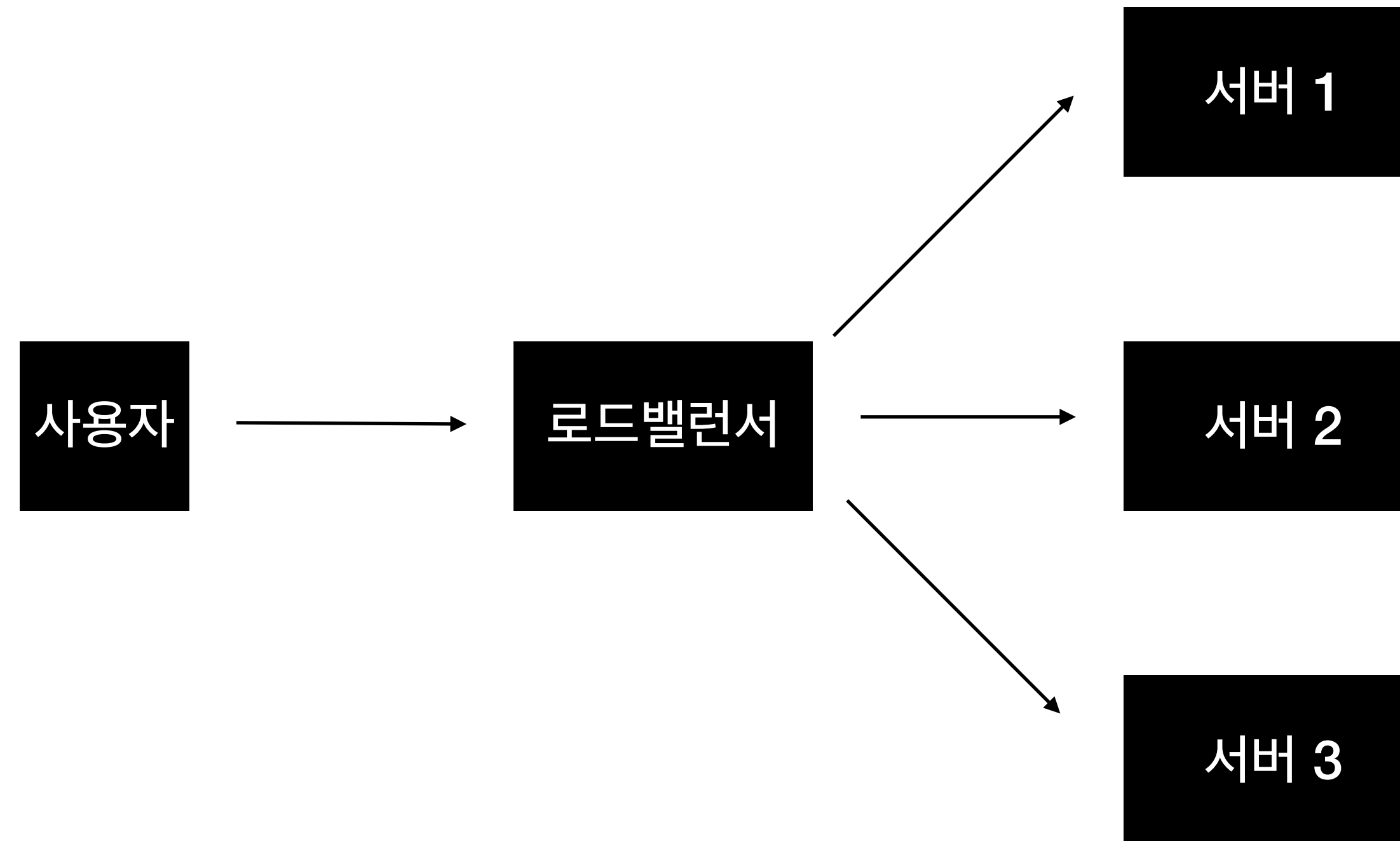
동시성 처리 - Server 관점

Synchronize 한계점

- 서버 (JVM) 내에 존재하는 리소스에 대해서만 Sync 를 맞출 수 있다.
- JVM 외부, 노드 외부의 DB 와 소통 한다.
- DB 에 대한 Sync 를 맞출 수 있으려면 어떻게 해야할까...?

동시성 처리 - Server Cluster 관점

Server Cluster



동시성 처리 - Server Cluster 관점

Server Cluster

- 서버 내부의 Thread 를 통해 병렬적인 처리를 하는 것 처럼
- 서버 자체를 N 개로 늘려 병렬적인 처리를 할 수 있다.
 - e.g. 사용자 트래픽이 몰린다면? 서버를 늘린다.
- 여러 서버들의 집단을 형성
- 물론 모든 서버가 동일하지 않고, 각각의 서버가 다른 역할을 하게 할 수도 있고...

동시성 처리 - Server Cluster 관점

공유 자원으로서의 데이터 베이스

- 앞의 예시와 동일한 문제
- e.g. 특정 값을 2배 해서 업데이트 해줘.
- 두 요청이 병렬적으로 행해진다면?
 - 2배가 될 수도
 - 4배가 될 수도..

동시성 처리 - Server Cluster 관점

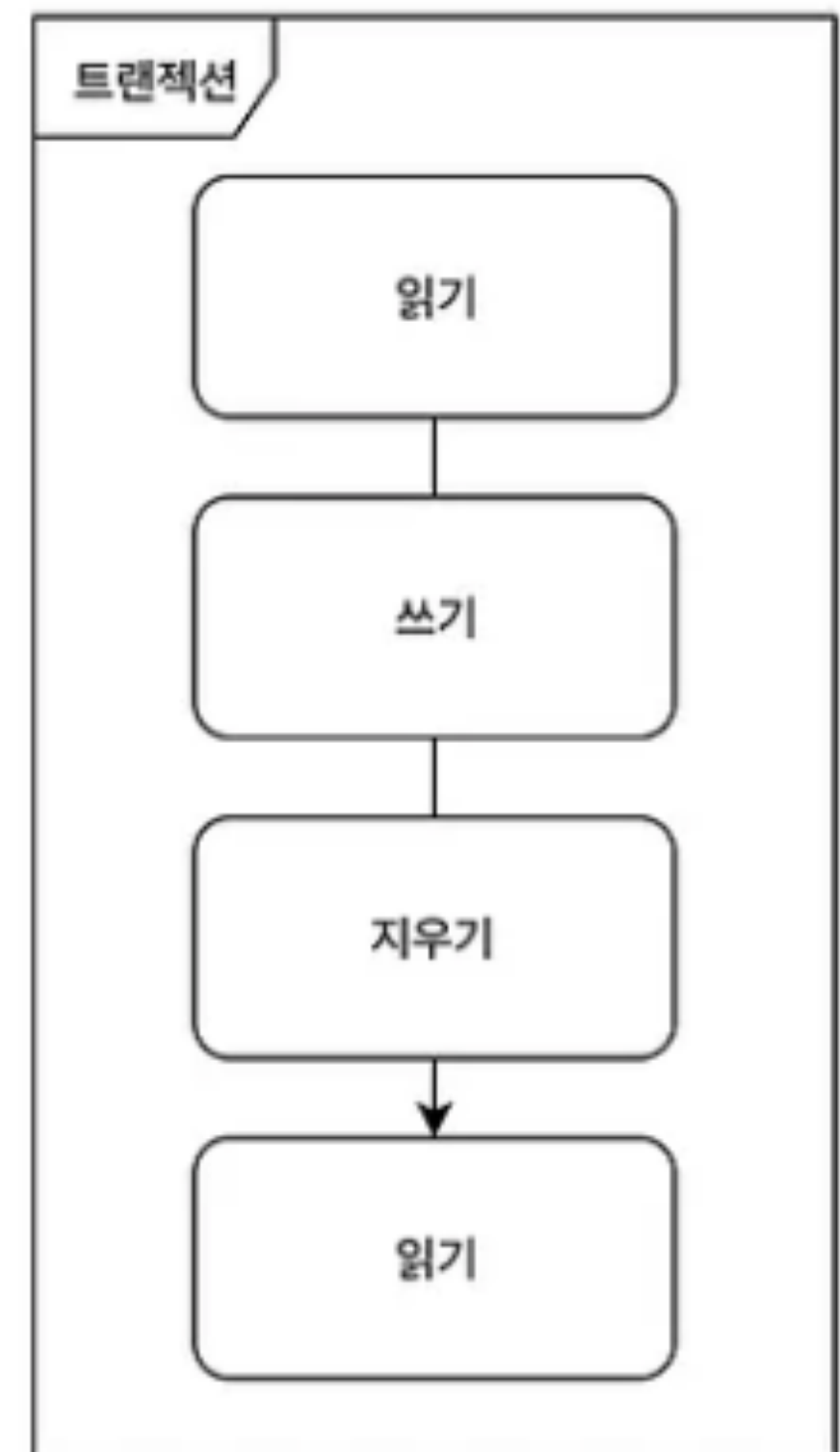
공유 자원으로서의 데이터 베이스

- 자원에 대한 독점을 선언해야 한다
- 자원 종류에 따라
 - Table Lock (테이블 썬로 건들지마)
 - Gap Lock (이 조건들은 건들지마)
 - Row Lock (이 데이터 한 줄은 건들지마)
- 락의 세기에 따라
 - 쓰지마 (Shared Lock)
 - 읽지도마 (Exclusive lock)

MySQL Transaction

공유 자원으로서의 데이터 베이스

- 락을 필요할 때 마다 잘 걸어주기... 생각만 해도 머리 아프다....
- 내가 진행하는 일련의 요청이 Atomic 하게 이뤄진 것 처럼 할 수 있을까?
- 은행 송금 (내 계좌 조회, 내 계좌 -, 남 계좌 조회, 남 계좌 +)
- 티켓 구매 (남은 수량 확인, 결제 진행, 남은 수량 -)



MySQL Transaction

트랜잭션의 특징 ACID



MySQL Transaction

MySQL 의 트랜잭션 구현체

- 트랜잭션 구현은 말만 들어도 매우 까다롭다.
- 터프하게 모든 요청은 트랜잭션 시작 부터 다 X-Lock 걸기
 - 완전무결 But 성능 최악
- 하나도 안걸기...
 - 완전무결 x 성능 최고
- 어떻게 둘의 중간을 잡을까....

```
START TRANSACTION;
```

```
UPDATE country_new  
SET country_id = @updated_ci  
WHERE country_id = @outdated_ci;
```

```
UPDATE dup_countries  
SET country_id = @updated_ci  
WHERE country_id = @outdated_ci;
```

```
UPDATE world_locations  
SET country_id = @updated_ci  
WHERE country_id = @outdated_ci;
```

```
COMMIT;
```

MySQL Transaction

MySQL 의 트랜잭션 구현체

- Q) Transaction 중간에 rollback 되면, 작성한 내용을 다 치워야 한다. 어떻게?
- Q) 다른 Transaction 에서 작성되고 있는 내용이 내 Transaction 에 노출되면 안된다. 어떻게?
- Q) 어떻게 적절한 시점에 락을 걸까...
- *Recap*) JPA 의 *Cache* 무결성의 보장은 이런 *Transaction* 을 통해 느슨하게 진행할 수 있다.

MySQL Transaction

JPA 에서 Transaction 쓰기

- `@Transactional` 만 붙여주면, 해당 함수 호출 동안 Transaction 내부임을 보장한다.

```
@Transactional  @sanggsgg *  
fun createReview(  
    author: User,  
    menuId: String,  
    content: String,  
    rating: Int,  
): Review {
```