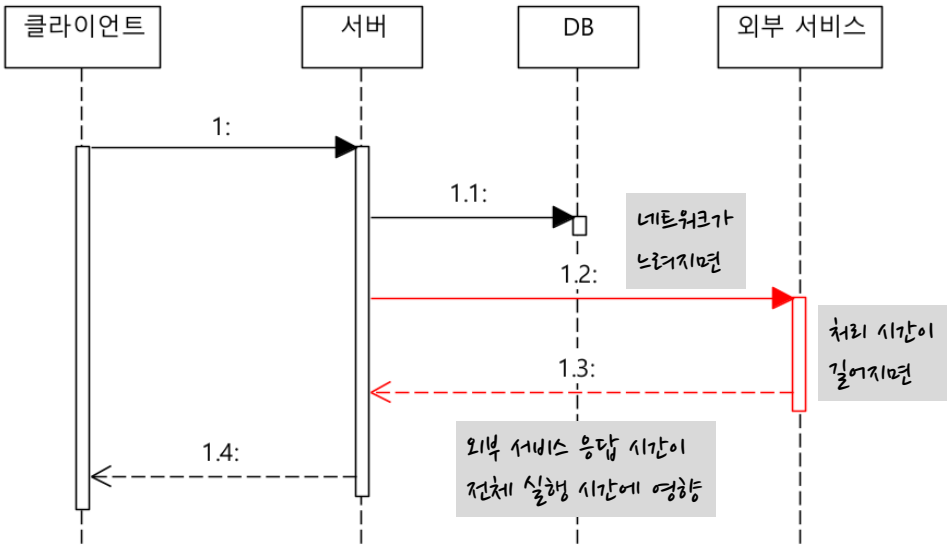
# 초식 : 초보자용
# 웹 서비스 성능 올리기 - 외부 연동 비동기

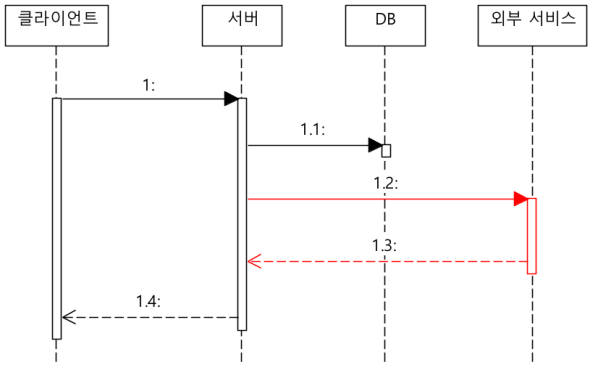
최범균

## 외부 연동과 성능



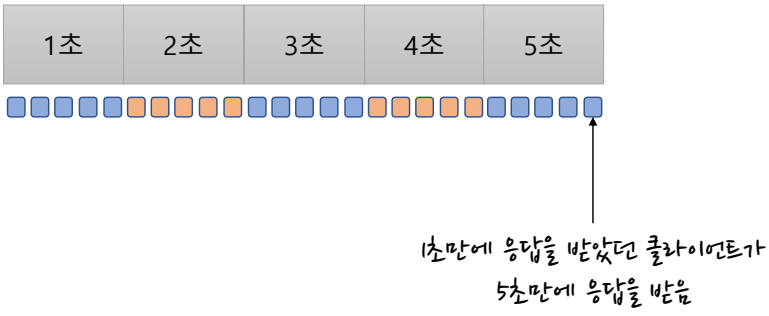
# 외부 서비스의 응답 시간과 성능 저하

- 예: 서버 쓰레드풀 5개, 평균 응답 시간 0.2초
  - 응답 시간에서 외부 연동이 차지하는 시간이 0.1초
  - 1개 쓰레드가 초당 5개 요청 처리
  - 5개 쓰레드가 초당 25개 요청 처리
  - 25 TPS
- 외부 연동 0.1초 → 0.9초 증가하면
  - 응답 시간 1초로 증가
  - 1개 쓰레드가 초당 1개 요청 처리
  - 5개 쓰레드가 초당 5개 요청 처리
  - 5 TPS




# 줄 서는 상황 발생

- 외부 서비스 성능 저하 상황에서 25개 클라이언트가 요청하면

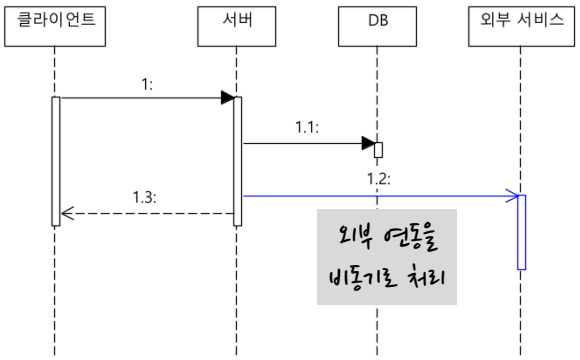


# 그런데 외부 연동을 즉시 해야 하나?

- 예1: 로그인하면 포인트 적립
  - 로그인 로직에서 외부 포인트 적립 서비스 호출
  - 로그인에 성공한 뒤, 수 초 이내 포인트 적립 서비스 호출
- 예2: 주문 취소하면 푸시 발송
  - 주문 취소 로직에서 외부 푸시 발송 API 호출
  - 주문 취소 후, 수십 초 이내 외부 푸시 발송 API 호출

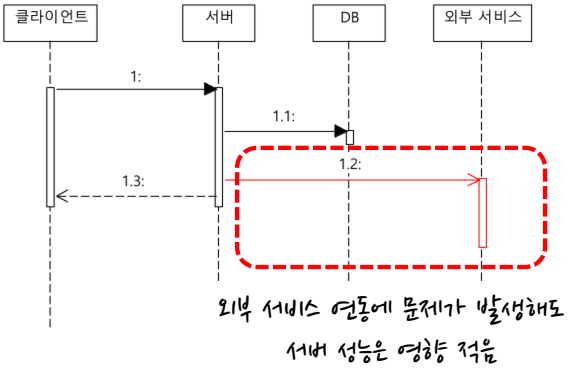
# 외부 연동에 비동기 고려

- 별도 쓰레드, 프로세스로 외부 연동 처리



# 외부 연동을 비동기로 처리하면

- 외부 서비스에 성능 문제가 발생해도 영향 작음
  - 즉, 외부 서비스 상태에 상관없이 응답 시간/TPS를 일정 수준 유지




# 비동기 처리 방식 3가지

- 별도 쓰레드
- 연동 데이터를 DB에 저장하고 별도 쓰레드/프로세스로 처리
- 연동 데이터를 메시징 시스템에 저장하고 별도 쓰레드/프로세스로 처리

# 별도 쓰레드로 실행

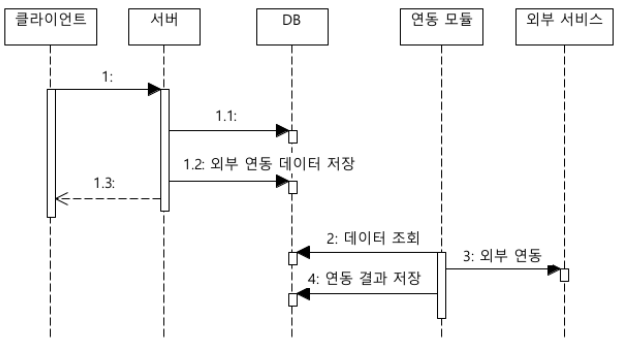
- 외부 연동 코드를 별도 쓰레드로 실행
- (요즘은) 구현이 쉬움
- (보통은) 거의 즉시 실행
- 고민 거리
  - 트랜잭션 처리
  - 서버 재시작시 작업 유실
  - 외부 연동 실패시 재처리
  - 쓰레드 풀 크기
- 재처리 필요성이 낮은 외부 연동에 적합
  - 예: 푸시 발송 등

```
dao.insert(data);
pushService.pushAsync(pushData);
```

```
@Async
public void pushAsync(PushData pushData) {
    pushClient.sendPush(...);
}
```

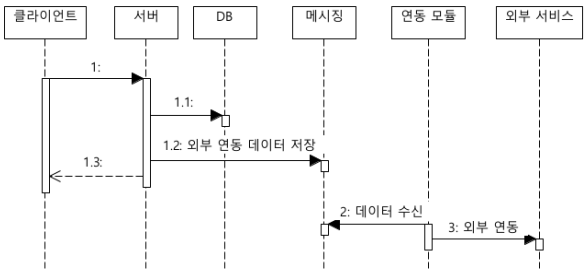
# DB 사용

- 연동 데이터를 DB에 저장
- 별도 쓰레드/프로세스가 연동 처리
  - DB에서 연동 데이터를 조회한 뒤
  - 외부 연동 처리
- DB 트랜잭션 처리 쉬움
- 외부 연동 실패시 재처리 쉬움
- 누락이 없어야 할 연동에 적합
  - 예: 로그인 포인트 적립



# 메시징 시스템 사용

- 연동 데이터를 메시징 시스템에 저장
  - 카프카 같은 곳에 저장
- 별도 쓰레드/프로세스가 연동 처리
  - 메시징 시스템에서 데이터 수신해서
  - 외부 연동 처리
- 대량 데이터 처리에 이점
- 외부 연동 실패시 재처리 용이
- 데이터 유실 가능성 고려




# 정리

- 비동기 처리 → 복잡도 증가
  - 모든 외부 연동을 비동기로 할 필요는 없음
  - 외부 연동이 성능에 영향을 준다면 비동기 연동을 고려
- 재처리/트랜잭션 등 제약이 약하면 별도 쓰레드로 비동기 처리
- 연동/성능에 대한 요구가 높다면 DB + 메시징 조합까지 고려