1장. 사용자 수에 따른 규모 확장성

캐시 사용시 데이터의 일관성은 어떻게 유지할까?
캐시 SPOF(Single Point Of Failure)를 방지하기 위한 방법

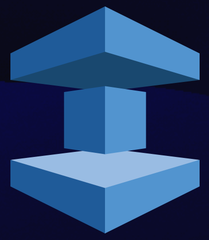
제이 (jay-so)

목차

1. 캐시 사용 예시

1. 캐시 사용 예시

다양한 캐시의 종류



AWS ElastiChache

Redis

Memcached

Apache Ignite

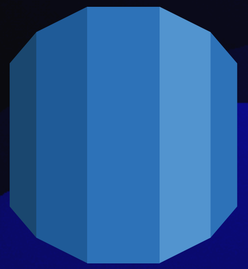
1. 캐시 사용 예시

Redis 캐시를 사용하는 예시

Redis

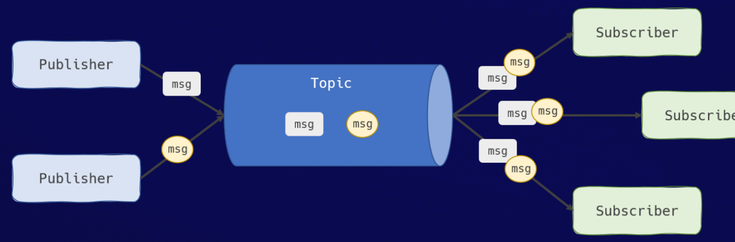
JWT

JWT의 Refresh 토큰 저장 용도



DB에서 자주 읽는 데이터를
저장하는 용도

세션을 저장 용도



Message Pub/Sub 용도

2. 캐시 사용시 유의할 점

2. 캐시를 사용시 유의할 점(고려할 사항)

1. 데이터 갱신은 자주 일어나지 않지만, 조회는 자주 일어나는가?

2. 별도의 RDBMS의 영속으로 기록할 필요가 없거나, RDBMS에서 기록된 내용되는 로직이 별도로 있는가?

3. 캐시에 보관된 데이터는 언제 만료되어야 하는가?

- 만료 정책이 없으면, 해당 데이터는 캐시에 계속 남게 되어 적절한 만료 기한 설정이 필요합니다.

4. 데이터의 일관성은 어떻게 유지할 것인가?

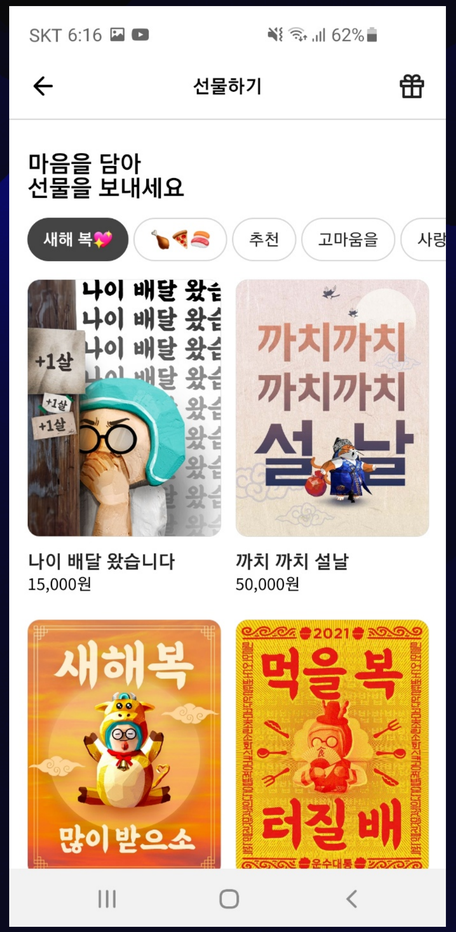
- DB에 저장하는 연산과 캐시를 갱신하는 연산이 단일 트랜잭션으로 처리되지 않으면, 데이터의 일관성은 깨질 수 있습니다.

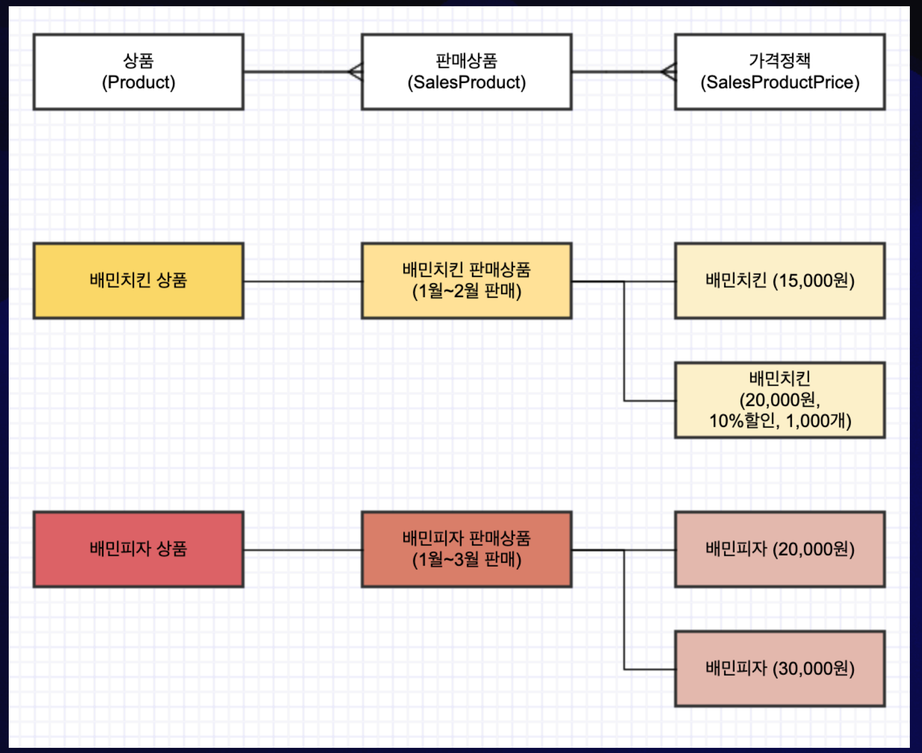
5. 장애에 대해서는 어떻게 대처할 것인가?

- 캐시 서버를 한 대만, 두는 경우, 해당 캐시 서버는 단일 장애 지점(SPOF)가 되어 버릴 가능성이 큽니다.

2. 캐시를 사용시 유의할 점(고려할 사항) - 예시

상품권 재고관리를 위한 시스템 설계(우아한 형제들 선물하기 시스템)





2. 캐시를 사용시 유의할 점(고려할 사항) - 예시

상품권 재고관리를 위한 시스템 설계(우아한 형제들 선물하기 시스템)

재고 관리 요구사항 (3가지)

1. 상품권의 권종별로 전체 재고 수량과 인당 재고 수량이 관리되어야 한다.

- 전체 재고는 종류별로 관리되고, 한 사람당 구입 횟수를 제한하며 전체 종류에서 구입 가능한 횟수를 제한해야 함

2. 상품은 전체 재고량을 초과하여 판매하면 안된다.

- 종류별로 제한된 재고량은 절대 초과되어서 판매되면 안됨

3. 판매가 시작된 상품의 전체 재고 수량은 감소시킬 수 없다.

- 판매가 한번 시작된 상품의 경우, 재고량 수정이 가능하나 최초 설정된 재고량 이상을 설정할 수 없음

2. 캐시를 사용시 유의할 점(고려할 사항) - 예시

상품권 재고관리를 위한 시스템 설계(우아한 형제들 선물하기 시스템)

설계 고려 사항 (4가지)

1. 전체 재고량의 관리와 트랜잭션이 일어나는 재고 사용량은 분리하여 저장한다.

- 전체 재고량의 경우, RDBMS에 저장하여 관리하고 트랜잭션이 일어나는 재고 사용량의 관리는 연산 속도가 빠른 인 메모리 DB를 사용함

2. 재고 사용량의 증가와 감소 시 동시성 이슈는 없어야 한다.

- 단일 스레드에서 연산처리를 하는 Redis 사용하여 동시성 이슈를 해결함

3. 재고 사용량 데이터는 유실되어서는 안된다.

- 인 메모리 DB는 휘발성 데이터로 데이터 손실이 일어날 수 있으므로, 재고 사용량 데이터를 RDB에 싱크함

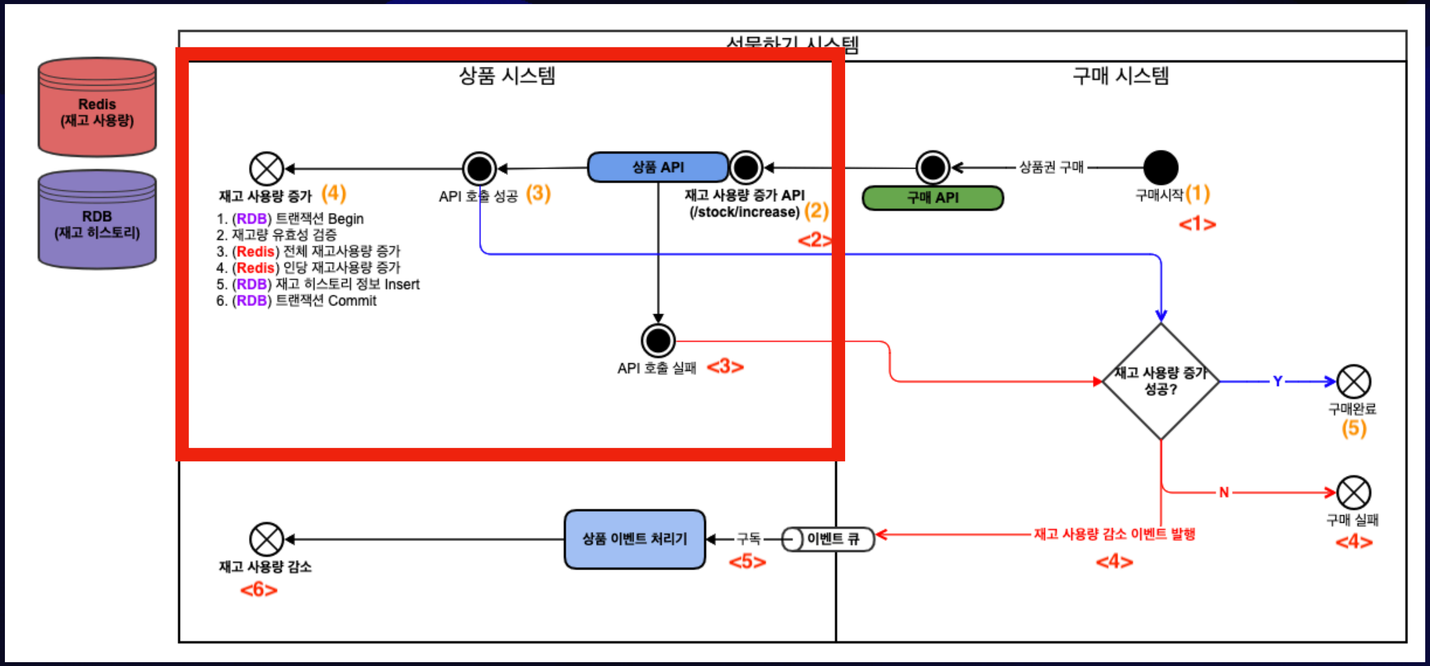
4. 재고 사용량의 관리는 Redis의 Set 자료구조에 구매번호를 저장하여 관리한다.

- 구매번호는 유니크한 값이며, Redis의 Set 자료구조를 통해 중복을 피하기 위함

2. 캐시를 사용시 유의할 점(고려할 사항) - 예시

데이터의 일관성은 어떻게 유지할 것인가?

재고량 증가



구매 시스템에서 구매 완료 후,
상품 시스템 API 호출

1. (RDB) 트랜잭션 시작 - Trasaction Begin
2. 현재 구매가 가능한 상태인지, 유효성 검증 과정을 거침
3. (Redis)구매가 가능할 경우 - Redis에 구매 번호를 추가
4. (Redis) 인당 재고 사용량 증가
5. (RDB) 구매 히스터리에 변경된 구매 정보를 입력
6. (RDB) 트랜잭션 커밋(Commit)

2. 캐시를 사용시 유의할 점(고려할 사항) - 예시

데이터의 일관성은 어떻게 유지할 것인가?

구매 시스템에서 구매 완료 후,
상품 시스템 API 호출

1. (RDB) 트랜잭션 시작 - Trasaction Begin
2. 현재 구매가 가능한 상태인지, 유효성 검증 과정을 거침
3. (Redis)구매가 가능할 경우 - Redis에 구매 번호를 추가
4. (Redis) 인당 재고 사용량 증가
4. 구매가 가능할 경우 RDB 구매 정보를 입력
5. (RDB) 트랜잭션 커밋(Commit)

"만약 RDBMS 트랜잭션이 실패되어
롤백(Roll Back)이 된다면?"

RDBMS는 트랜잭션이
커밋되지 않은 상태

Redis에 쓰여진 데이터와
RDBMS와의 데이터 사이에 불일치 발생

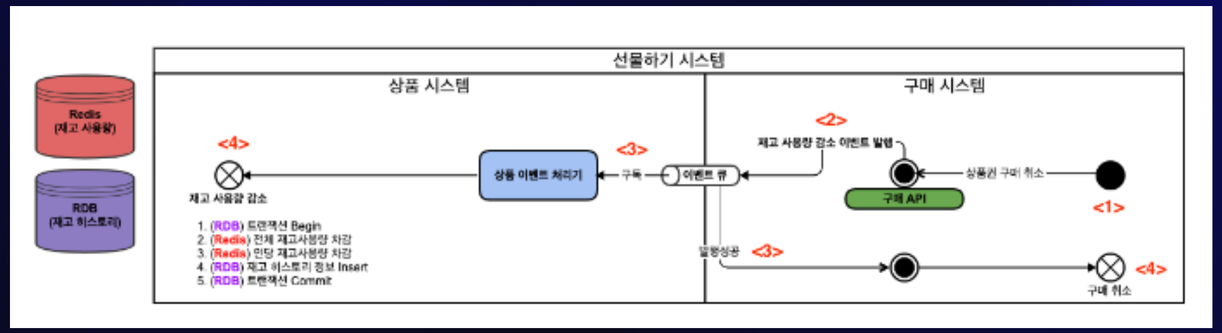
2. 캐시를 사용시 유의할 점(고려할 사항) - 예시

데이터의 일관성은 어떻게 유지할 것인가?

재고량 감소

구매 프로세스가 시작되어, 먼저 재고 사용량이 증가 되었다가
구매 취소가 되거나, API 호출 등의 이유로 구매가 완료되지 않을때

증가된 Redis의 재고 사용량
(전체 재고량, 인당 재고 사용량)을
차감하기 위해서 비동기로 이벤트가 발생되어 처리함(복구)



1. (RDB) 트랜잭션 시작 - Trasaction Begin
2. 현재 구매가 가능한 상태인지, 유효성 검증 과정을 거침
3. (Redis)구매가 가능할 경우 - Redis에 구매 번호를 추가
4. (Redis) 인당 재고 사용량 증가
5. (RDB) 구매 히스토리에 취소된 데이터 정보 입력
6. (RDB) 트랜잭션 커밋(Commit)

2. 캐시를 사용시 유의할 점(고려할 사항) - 예시

데이터의 일관성은 어떻게 유지할 것인가?

재고량 감소 - 비동기 처리 이유(3가지)

1. 전체 재고량을 관리하고, 재고 사용량을 증가 혹은 차감 시키는 방식을 사용한다.

재고 사용량 증가가 동기로 처리되는 이유: 한 번에 하나의 요청만을 처리하여 재고가 중복 판매가 되는 문제를 방지함
재고 사용량 감소가 비동기로 처리하는 이유: 여러 사용자의 요청을 동시에 받아서 처리할 수 있기 때문

- 전체 재고량을 관리하여, 재고 시스템에서 실제 재고 사용량보다 더 많이 사용되는 판매 상황을 방지하고, 동시에 여러 사용자의 취소를 허용함으로 재고 시스탬을 효율적으로 관리할 수 있음

2. 재고 사용량 증가 방식을 동기 방식으로 처리함으로써, 절대 재고가 더 팔리는 일은 발생되지 않는다.

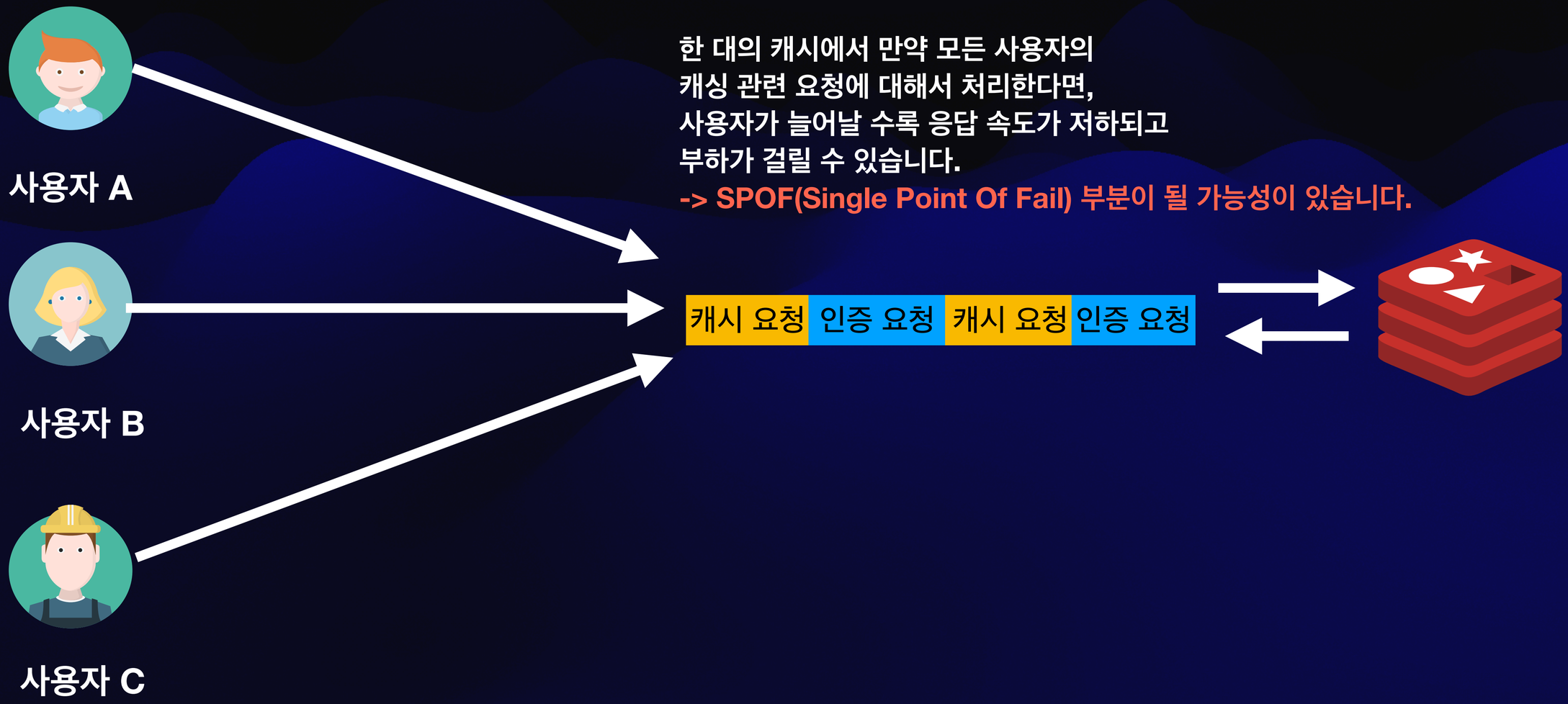
- 재고 사용량 증가 방식은 동기 방식이므로, 하나의 재고가 두 번 팔리는 일은 없기 때문

3. Redis의 Set 자료구조를 사용함으로써, 재고 사용량 차감에 대한 잘못된 구매 번호의 이벤트가 발생되어도 재고 사용량 차감에는 영향을 미치지 않는다.

- Redis의 Set을 사용하면, 같은 구매 번호에 대한 중복된 요청이 들어와도 Set의 특성 상 유니크한 값으로만 저장되기 때문에, 잘못된 구매 번호로 인한 중복 차감을 방지할 수 있기 때문

3. 단일 캐시 사용시 유의할 점

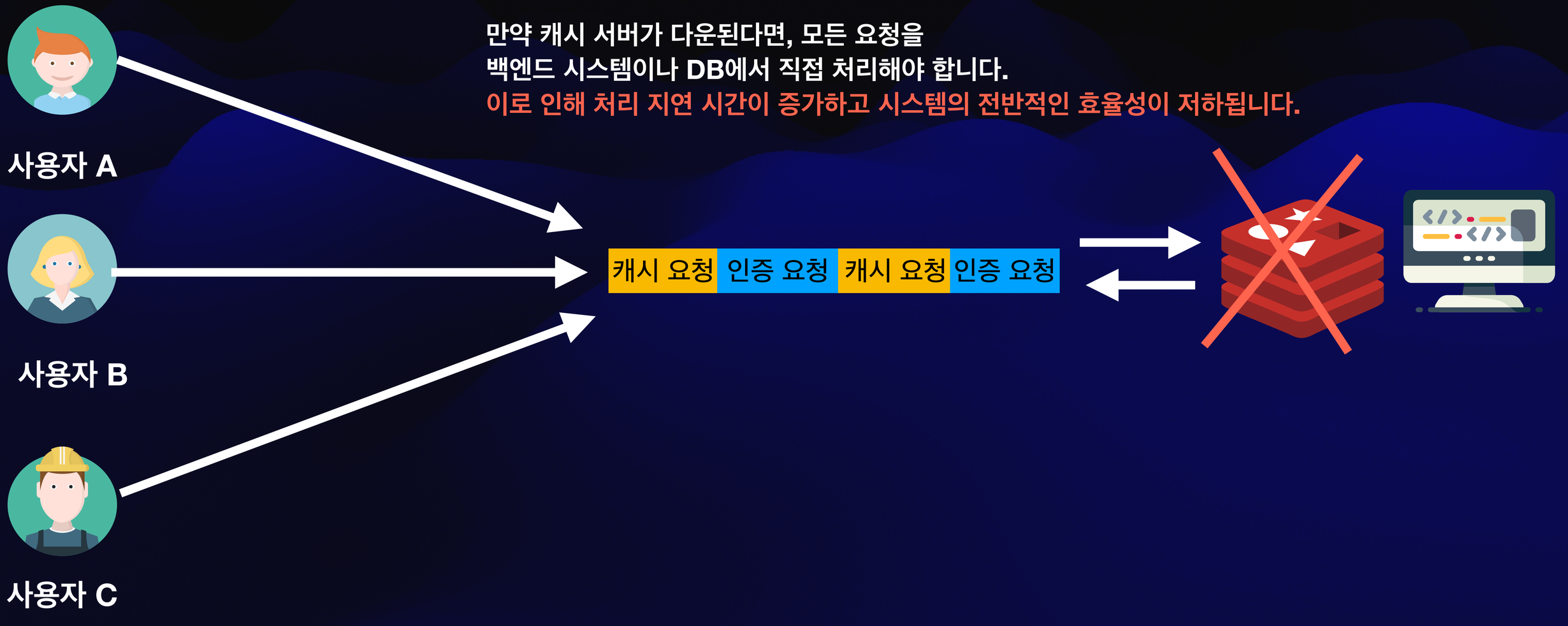
3. 단일 캐시 사용시 유의할 점



한 대의 캐시에서 만약 모든 사용자의
캐싱 관련 요청에 대해서 처리한다면,
사용자가 늘어날 수록 응답 속도가 저하되고
부하가 걸릴 수 있습니다.
-> SPOF(Single Point Of Fail) 부분이 될 가능성이 있습니다.

3. 단일 캐시 사용시 유의할 점

만약, 해당 캐시 서버가 다운된다면?



만약 캐시 서버가 다운된다면, 모든 요청을
백엔드 시스템이나 DB에서 직접 처리해야 합니다.
이로 인해 처리 지연 시간이 증가하고 시스템의 전반적인 효율성이 저하됩니다.

4. 단일 캐시의 SPOF를 방지하기 위한 방법

4. 단일 캐시의 SPOF를 방지하기 위한 방법

단일 캐시의 SPOF를 방지하기 위한 방법으로는 다음과 같은 예가 있습니다.

단일 캐시의 SPOF를 방지하기 위한 방법 (3가지)

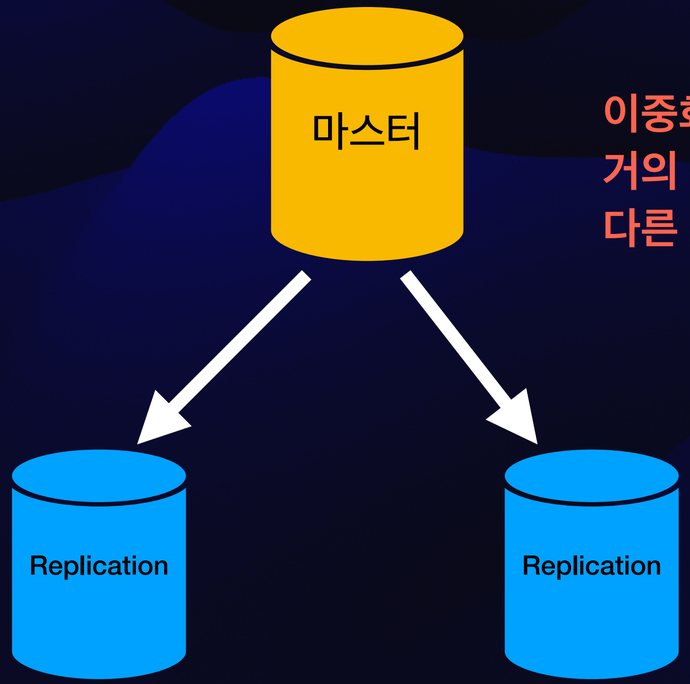
1. 레플리케이션(Replication)

2. 분산 캐시

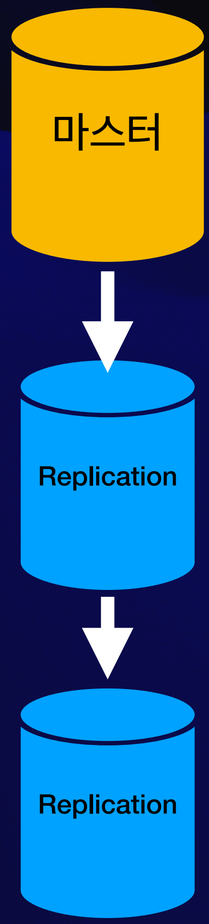
3. 로드 밸런싱

4. 단일 캐시의 SPOF를 방지하기 위한 방법

단일 캐시의 SPOF를 방지하기 위한 방법 - 레플리케이션(Replication)



이중화의 개념으로 데이터를
거의 실시간(비동기 방식)으로
다른 레디스 노드에 복사하는 방식입니다.



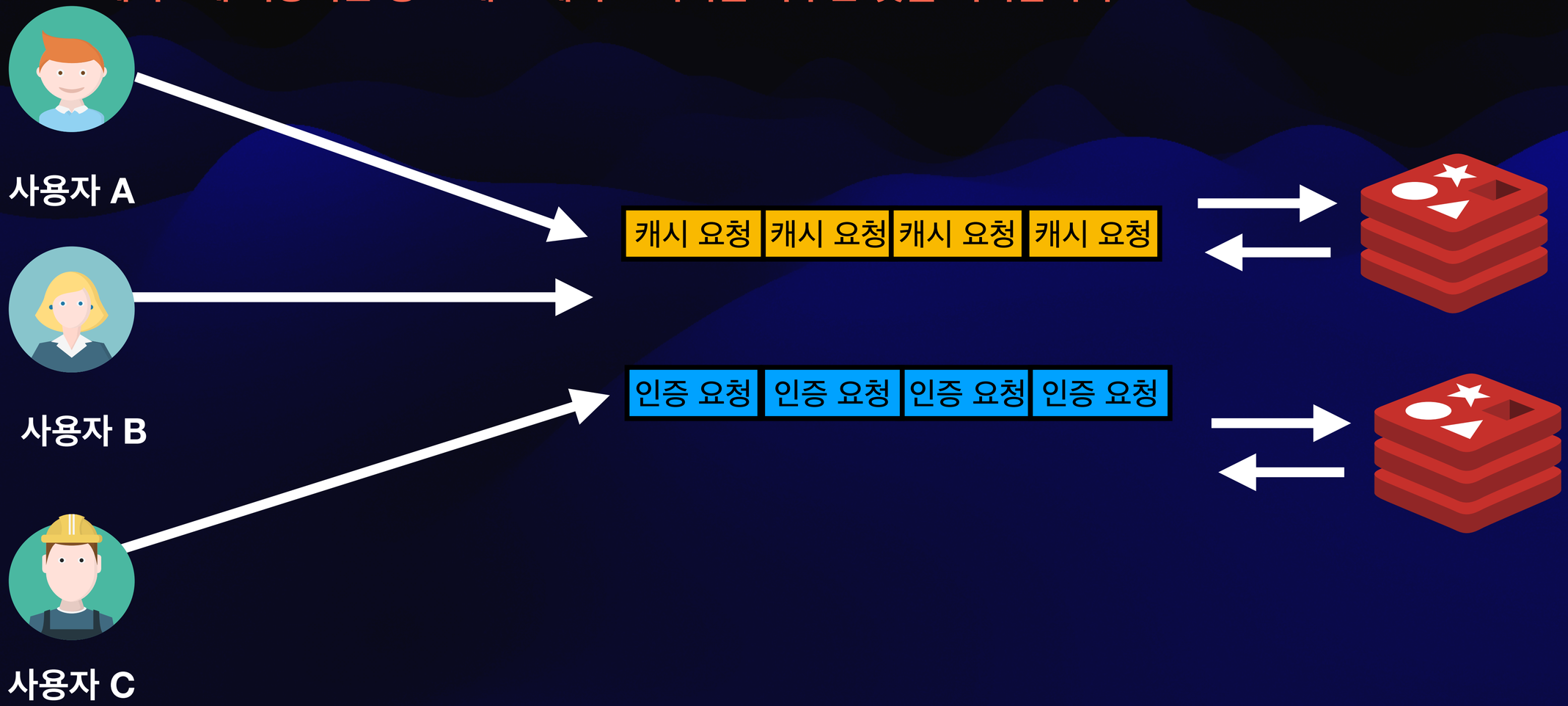
특징

1. 비동기 복제를 합니다.
2. 마스터는 여러 복제 서버를 가질 수 있습니다.
3. 복제 서버는 다시 복제 서버를 가질 수 있습니다.
4. 마스터가 다운되면, 복제 서버 중 하나가 마스터가 될 수 있습니다.

4. 단일 캐시의 SPOF를 방지하기 위한 방법

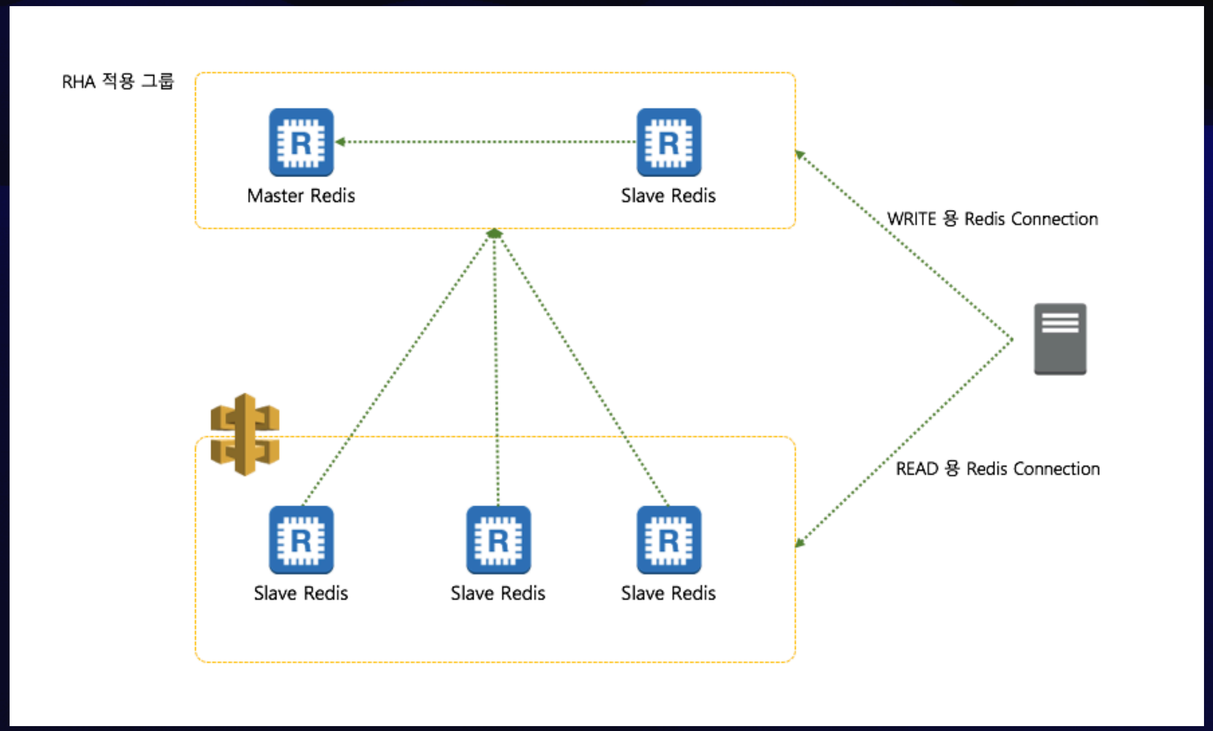
단일 캐시의 SPOF를 방지하기 위한 방법 - 분산 캐시

레디스에 저장되는 용도 대로 레디스 서버를 나누는 것을 의미합니다.



4. 단일 캐시의 SPOF를 방지하기 위한 방법

단일 캐시의 SPOF를 방지하기 위한 방법 - 로드 밸런싱



여러 캐시 서버가 동일한 데이터를 저장하고 있을 때,
로드 밸런서를 사용하여 들어오는 요청을
여러 서버에 균등하게 분배함으로써
단일 서버에 대한 부하를 줄이고,
서버 중 하나가 실패했을 경우에도
시스템의 가용성을 유지할 수 있습니다.

예시

- Write용 Redis
- Master Redis, Slave Redis

- Read용 Readis
- Master Redis, Slave Redis, Slave Redis

5. 참고 자료

5. 참고 자료

우아한 기술 블로그 - 선물하기 시스템의 상품 재고는 어떻게 관리되어질까?

https://techblog.woowahan.com/2709/

티스토리 블로그 - 레디스 리플리케이션

https://mozi.tistory.com/372

티스토리 블로그 - 레디스 분산 서버

https://chagokx2.tistory.com/99

티스토리 블로그 - 레디스 로드밸런싱

https://brunch.co.kr/@alden/29