Using Redis

Client-side caching in Redis

어플리케이션 메모리 내부에 응답 값을 직접 저장해서 네트워크 왕복을 줄이고, 빠르게 응답을 확인 할 수 있는 방식이다.

일관성 유지

클라이언트에서 저장하고 있기 때문에 일관성을 유지해야한다.

Redis Client-side caching은Tracking이라고 부르고, 2가지 모드를 제공한다.

1. Default Mode
    - 클라이언트가 Access한 키를 기억하고, 동일 키가 수정시 서버에서 무효화 메시지를 보낸다.
        - 클라이언트 목록 : 무효화 테이블
    - 클라이언트가 많아 질 수록 서버에 많은 부담을 갖게 된다.
2. BroadCast Mode
    - 클라이언트가 Access한 키를 기억하지 않는다.

Redis Pipelining

개별 명령에 대한 응답을 기다리지 않고 여러 명령을 내려 성능 향상시키는 기술

클라이언트-서버 모들을 따르고 TCP를 기반으로 하고 있기 때문에 IO에 대한 네트워크 병목 현상이 존재한다.

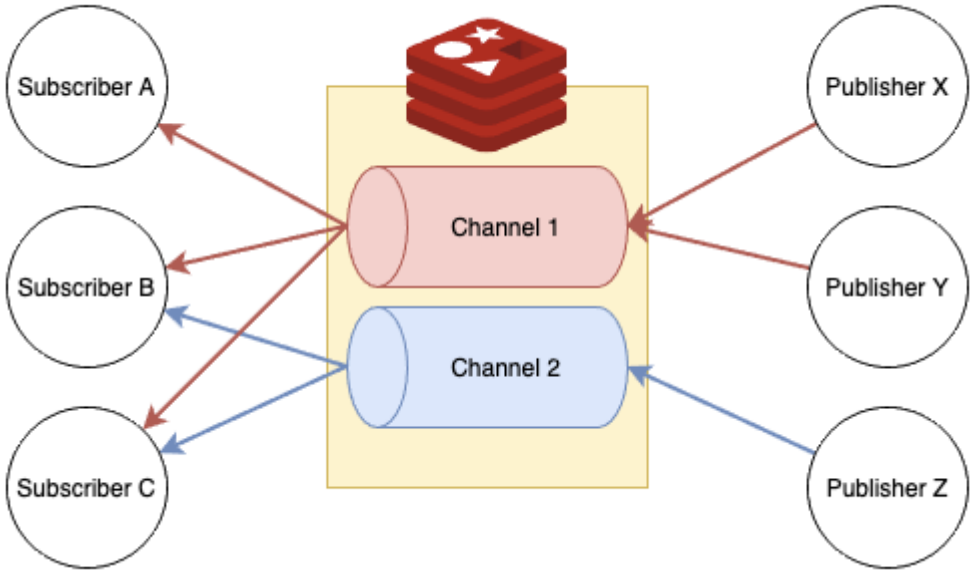
- RTT
    - 클라이언트 → 서버 → 클라이언트 왕복 시간
    - RTT 시간이 250ms 일 경우, 최대 초당 4개의 요청 처리 가능

Pipelining API 를 사용하면, 클라이언트의 여러 개의 작업을 쌓아서 한 번에 전송 할 수 있게 된다.

```
Client: INCR X
Server: 1
Client: INCR X
Server: 2
Client: INCR X
Server: 3
Client: INCR X
Server: 4
```

```
Client: INCR X
Client: INCR X
Client: INCR X
Client: INCR X
Server: 1
Server: 2
Server: 3
Server: 4
```

Pub/Sub



- 일반 Pub/Sub 모델을 redis에서도 지원
- Publisher 와 Subscriber는 서로 누군지 알 수 없고, 채널을 통해 메시지를 송/수신 한다.
- 다른 이벤트 큐와 다르게 redis에서 메시지를 저장하고 있지 않는다.
  - Publih 한 순간 대기중인 Subscriber에게만 메시지 전달
- 구독할 채널이름으로 패턴 매칭을 지원한다. `PSUBSCRIBE te*`

Redis keyspace notifications

Pub/Sub 구조를 사용하여, 데이터의 변경에 대한 메시지를 받을 수 있다.

기본적으로 사용시 CPU를 많이 사용하기 때문에 비활성화 되어 있다.

→ redis.conf를 수정하거나 config set 명령을 통해 활성화 시켜줘야 한다.

`notify-keyspace-events`

Timing of expired event

redis에서는 TTL이 0이 되는 순간 키가 삭제되는 것이 아니다.
→ 즉 키가 만료되었다고 하여 expired event를 바로 수신할 수 있는 것은 아니다.

- redis에서 키 만료 방식
  1. 해당 키를 조회하고, 해당 키가 만료되었다는 것을 발견했을 때 만료
  2. 해당 키를 조회하지 않더라도, 백그라운드에서 삭제
     a. 100m마다 redis cron이 activeExpireCycle(Slow) 실행
     b. redis event 전 beforesleep에서 activeExpireCycle(Fast) 실행

Events in a cluster

클러스터 모드에서는 모든 노드에 이벤트가 BroadCast 되지 않는다.

모든 키스페이스 이벤트를 수신하기 위해서는 클라이언트가 각 노드를 수신해야한다.

Transactions

redis에서도 RDS와 마찬가지로 Transaction을 제공

- 모든 명령을 직렬화되어 순차적으로 실행
- MULTI
  - redis에서 트랜잭션을 시작 커맨드
  - 이후 커맨드는 바로 실행되지 않고 큐에 쌓이게 된다.
- EXEC
  - queue에 쌓여있는 명령어를 일괄적으로 실행한다.
- DISCARD

    - queue에 쌓여있는 명령어를 폐기한다.
- WATCH
    - Optimistic Lock 기반으로 동작

유의 사항

1. EXEC 중 잘못된 자료구조의 명령어가 있더라도 정상 명령어는 모두 적용된다.

```
127.0.0.1:6379> multi
OK
127.0.0.1:6379(TX)> set SringTest str
QUEUED
127.0.0.1:6379(TX)> set listTest list
QUEUED
127.0.0.1:6379(TX)> lpush listTest 123
QUEUED
127.0.0.1:6379(TX)> exec
1) OK
2) OK
3) (error) WRONGTYPE Operation against a key holding the wrong kind of value
127.0.0.1:6379> get listTest
"list"
```

2. 아예 잘못된 명령어를 실행했을 경우 DISCARD 된다.

```
127.0.0.1:6379(TX)> set hello hi
QUEUED
127.0.0.1:6379(TX)> sett hello wer
(error) ERR unknown command `sett`, with args beginning with: `hello`, `wer`,
127.0.0.1:6379(TX)> exec
(error) EXECABORT Transaction discarded because of previous errors.
127.0.0.1:6379> get hello
(nil)
```

3. WATCH 커맨드는 Optimistic Lock으로 1번 변경할 수 있도록 제한한다.
    a. 하지만 하나의 트랜잭션 안에서는 여러번 가능하다. (마지막 값으로 변경)

```
127.0.0.1:6379> set str string
OK
127.0.0.1:6379> watch str
OK
127.0.0.1:6379> multi
OK
127.0.0.1:6379(TX)> set str str1
QUEUED
```

```
127.0.0.1:6379(TX)> set str str2
QUEUED
127.0.0.1:6379(TX)> set str str3
QUEUED
127.0.0.1:6379(TX)> exec
1) OK
2) OK
3) OK
127.0.0.1:6379> get str
"str3"
```