

# Redis?

Remote Dictionary Server 약자로 in-memory 저장소

데이터 저장소로 디스크가 아닌 메모리를 사용 → 초고속 데이터 저장소

기본적으로 key-value 저장 방식이지만 다양한 collection을 제공

영속적인 데이터 보존 기능 제공

redis는 메인으로 사용하는 데이터베이스보다는 보조적인 수단인 Cache, Message broker 용도로 많이 사용

레디스가 전 세계 데이터베이스에서 6위

## in memory

- 성능은 초당 약 10만회 명령을 실행, 속도가 천배에서 10배 차이
- 두가지 방식으로 메모리 데이터 단점을 커버
  - 데이터를 다른 서버에 실시간으로 복제, 이 기능을 이용하면 master server가 down 되어도 복제서버로 접속해서 서비스를 지속할 수 있음
  - 디스크 쓰기 기능을 제공

## 레디스를 사용하는 분야

### 1. 디지털 트윈

- 수많은 센서데이터를 실시간으로 받아 처리하는데 필요
- 컴퓨터에 현실 속 사물의 쌍둥이를 3D로 만들고, 현실에서 발생하는 데이터를 받아 트윈에 적용하는 기술입니다. 이로써 통합된 모니터링이 가능하고 시뮬레이션이나 데이터 분석도 가능

- 또 시뮬레이션 결과를 현실 사물에 반영하는 것도 가능

## 2. 채팅/메신저/챗봇

- 대화 내용을 임시로 레디스에 저장하고, 사용자 상태정보를 레디스의 pub/sub으로 구현

## 3. 사물인터넷

- 수많은 세션들로부터 데이터를 받아 처리하는데 레디스를 사용

## 4. 세션 스토어

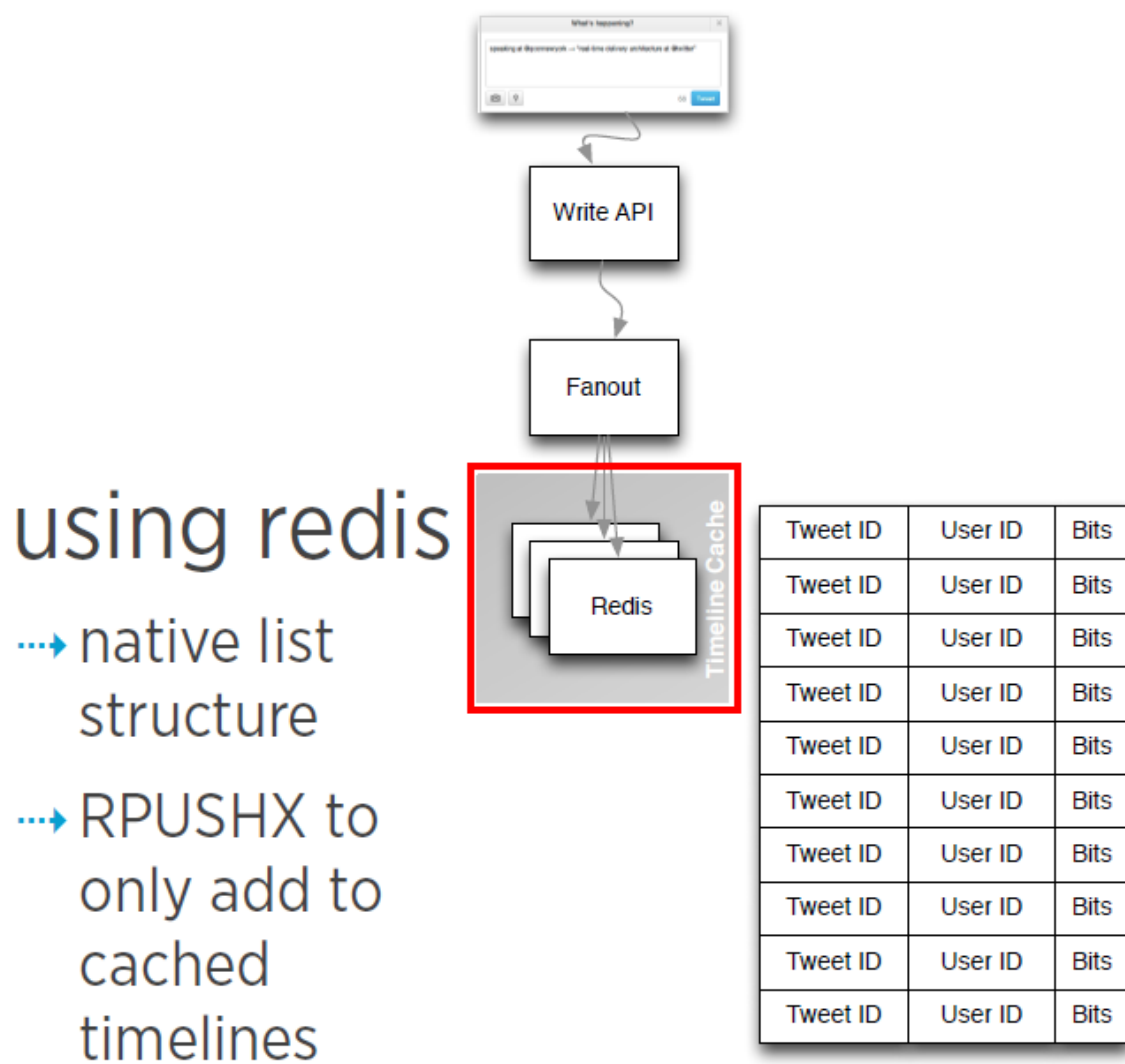
- 레디스가 가장 광범위하게 사용되는 곳은 데이터 저장 용도

# Redis 기업적용 사례

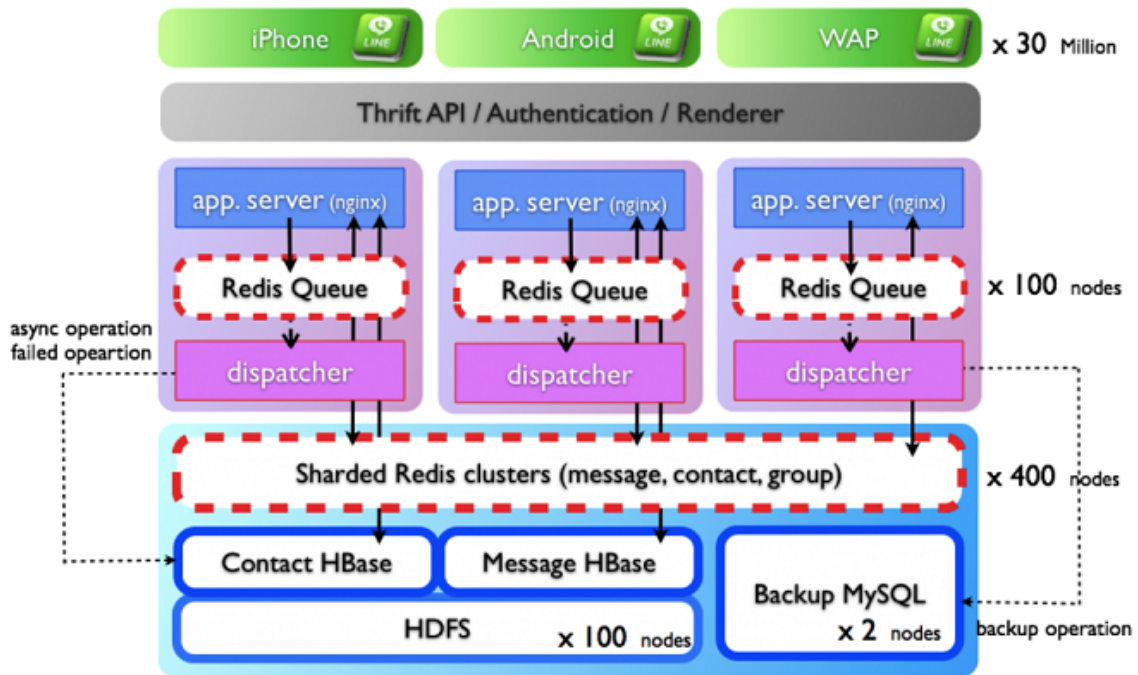
다양한 데이터 Collection

트위터, 웨이보, 라인, 카카오톡 등...

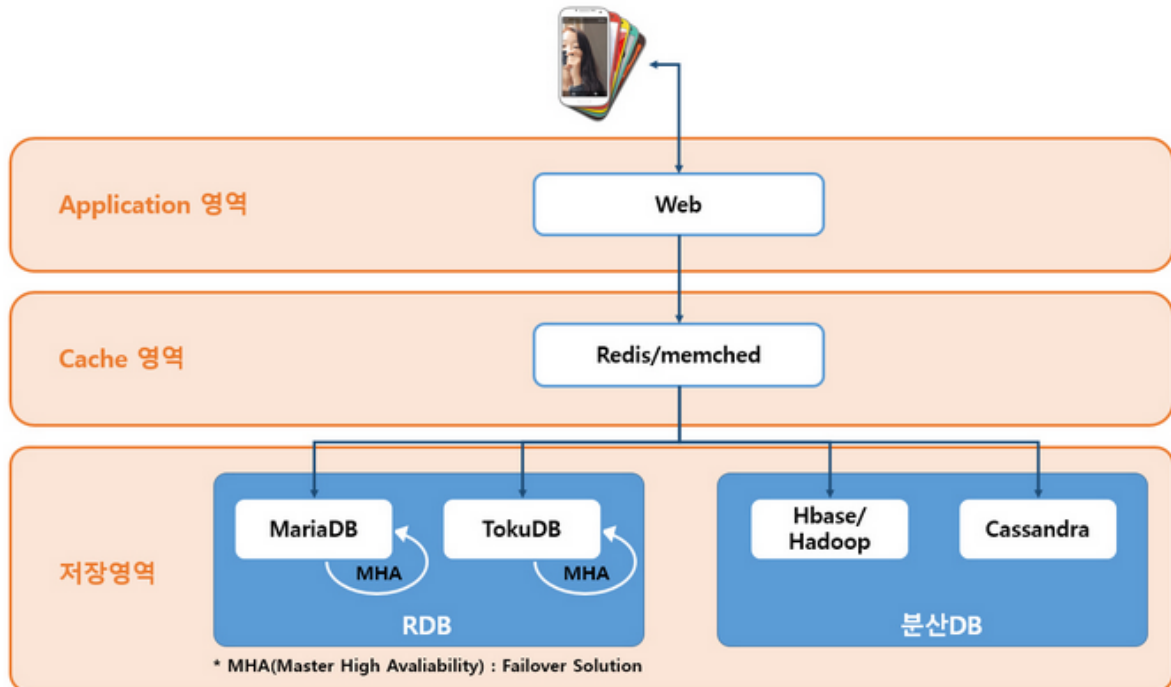
트위터 : 초당 30만 트윗을 처리할 수 있도록 구축, Timeline Cache, List를 수정/개선해서 사용



라인 : 앞 단 Queue 용도로 레디스를 사용



카카오톡 : cache 영역에서 사용

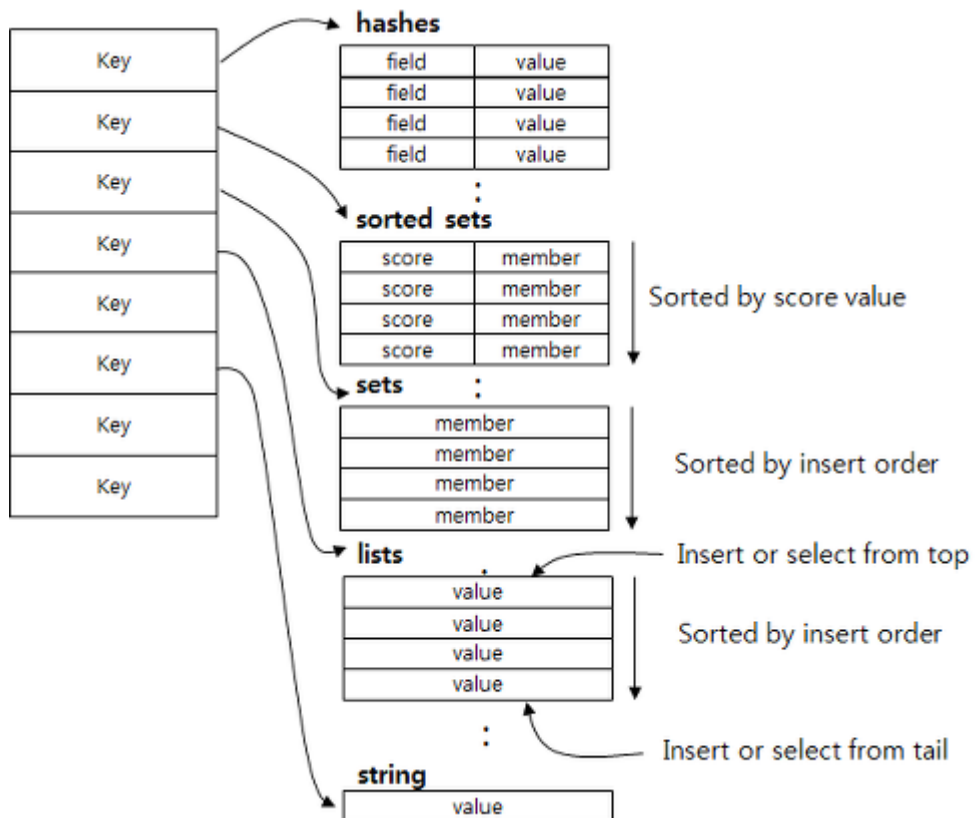


## 다양한 데이터 Collection

String : 일대일 관계

Lists, Sets, Sorted Sets, Hashes : 일대다 관계

Collection을 잘 이용함으로 여러가지 개발 시간을 단축시키고, 문제를 줄여줄 수 있음

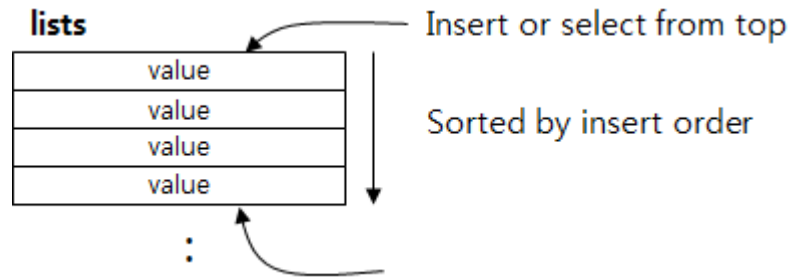


## 1. String : 문자열 <key, value>

- key와 value는 일대일 관계, 모두 최대 길이는 512MB

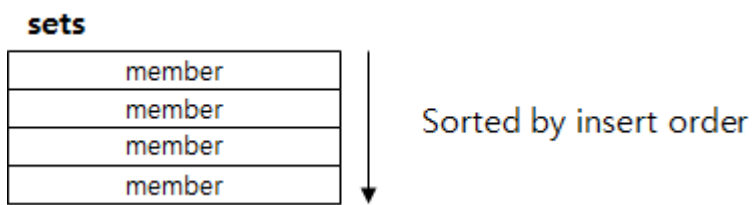
## 2. Lists : 리스트 <key, value[]>

- 데이터를 순차적으로 저장/처리하는데 사용
- 데이터값의 중복을 허용
- 큐(들어오는 데이터를 순서대로 처리할때), 스택(주로 되돌아갈때 사용)으로 사용



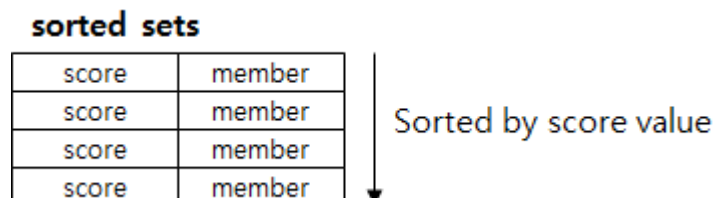
### 3. Sets : 집합 <key, Set<value>>

- 멤버(데이터) 중복을 허용하지 않음
- 집합의 성격을 갖는 데이터에 사용
- 집합연산(합집합, 교집합, 차집합)을 제공
- 친구리스트



### 4. Sorted Sets : 정렬이 되는 집합 <key, Set<value(with score)>>

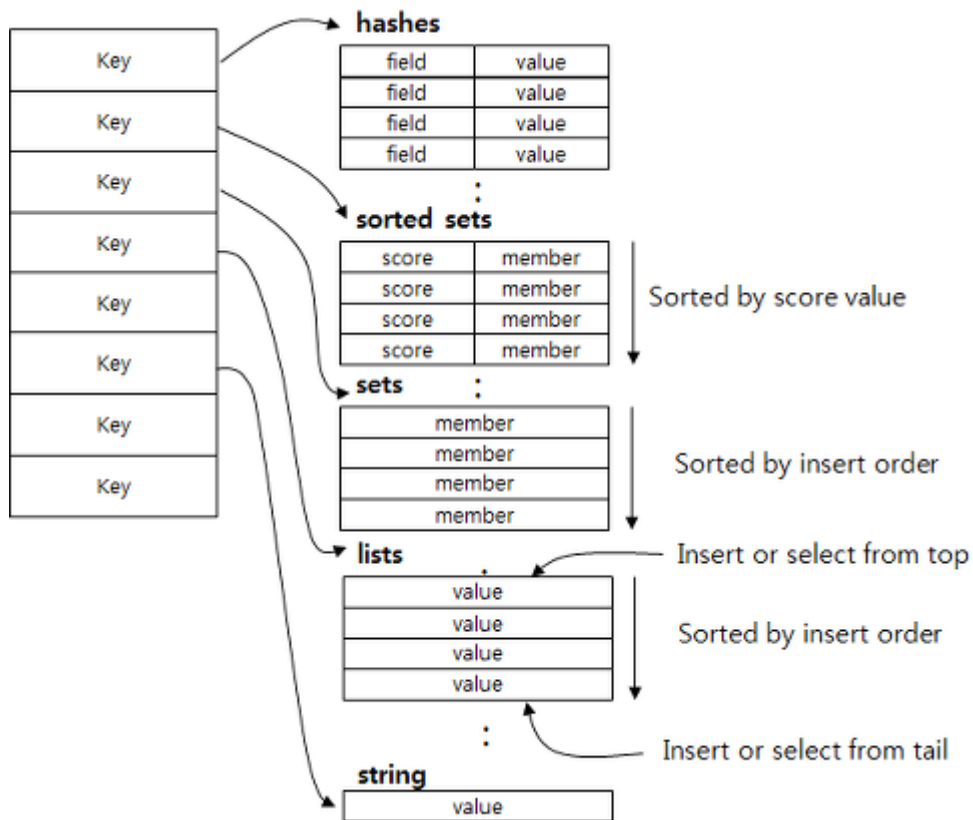
- key 하나에 여러개의 score와 value로 구성
- score로 member의 순서를 정렬, 정렬된 데이터가 필요한 경우 사용
- member의 중복을 허용하지 않음
- 합집합, 교집합 연산을 할 수 있고, score를 이용한 연산이 제공
- 랭킹시스템



### 5. Hash : 해시 <key, <field, value>>

- key 하나에 여러개의 field와 value로 구성

- value의 이름으로 구분할 수 있도록 field name이 제공
- RDB테이블과 유사함. Hash Key는 table 의 PK, field는 column
- Hash의 field수는 40억개로 무제한



#### ※ 데이터베이스와 비교

데이터베이스는 로우수가 증가하면 인덱스 크기도 커지고 검색속도도 느려짐. 큰 테이블을 검색하는데 인덱스를 사용하지 않는 쿼리를 실행하면 DB서버의 많은 리소스에 락을 걸어야 할 수도 있음

레디스는 모든 키검색은 O(1) 수행시간. 모든 작업의 수행시간이 일정해 1개의 키이든 100만개의 키이든 상관없이 GET을 수행하는데 각각 같은 시간이 소요. 그리고 이 시간이 매우 적게 듬.

## 데이터를 디스크에 저장하는 방식

redis는 데이터를 메모리에 저장하기 때문에 서버가 비정상적으로 종료되면 데이터가 유실되는 문제

이 문제를 해결하기 위해서 Disk에 데이터를 저장하는 RDB, AOF 두가지 방식을 지원

## 1. RDB(snapshot, 백업)

- 순간적으로 메모리에 있는 내용을 DISK 전체에 옮겨 담는 방식
- SAVE와 BGSAVE 두 가지 방식
  - SAVE는 blocking 방식으로 순간적으로 redis의 모든 동작을 정지시키고, 그때의 snapshot을 disk에 저장
  - BGSAVE는 non-blocking 방식으로 별도의 process를 띄운 후, 명령어 수행 당시의 메모리 snapshot을 disk에 저장하며, 저장 순간에 redis는 동작을 멈추지 않고 정상적으로 동작
- 장점 : 메모리의 snapshot을 그대로 뜯는 것이기 때문에, 서버 restart시 snapshot만 load하면 되므로 restart 시간이 빠름
- 단점 : snapshot을 추출하는데 시간이 오래 걸리며, snapshot 추출된후 서버가 down 되면 snapshot 추출 이후 데이터는 유실

## 2. AOF(Append Only File, 보관)

- 모든 쓰기, 업데이트 연산 자체를 모두 로그 파일에 기록하는 형태
- 서버가 재시작 될 때 기록된 write, update 연산을 순차적으로 재실행하여 데이터를 복구
- 기본적으로 non-blocking call
- 장점 : 항상 현재 시점까지의 로그를 기록할 수 있으며, Log file에 기록만 하기 때문에 log write 속도가 빠름
- 단점 : 모든 쓰기, 업데이트 연산을 로그로 남기기 때문에 RDB방식보다 데이터량이 과대하게 크고, 복구시 모든 연산을 재실행하기 때문에 재기동 속도가 느림

## 3. RDB + AOF

RDB와 AOF 장단점을 서로 상쇄하기 위해 두가지 방식을 같이 쓸 것을 권장함

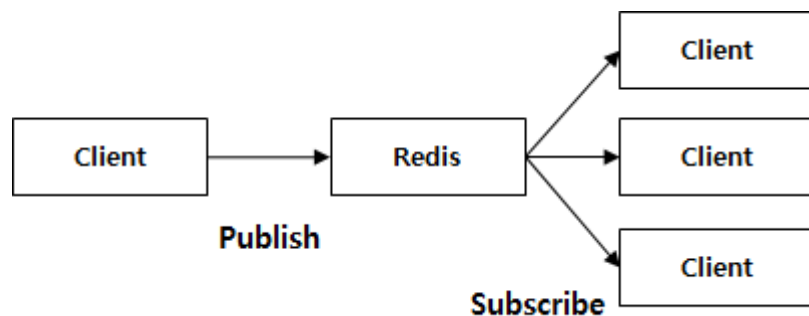
- 주기적으로 snapshot으로 백업 + 다음 snapshot까지 저장을 AOF로 수행
- 서버 재기동시 백업된 snapshot을 reload하고 소량의 로그만 재실행



이 방식은 AOF 로그만 replay하면 되기 때문에, restart 시간을 절약하고 데이터의 유실을 방지함

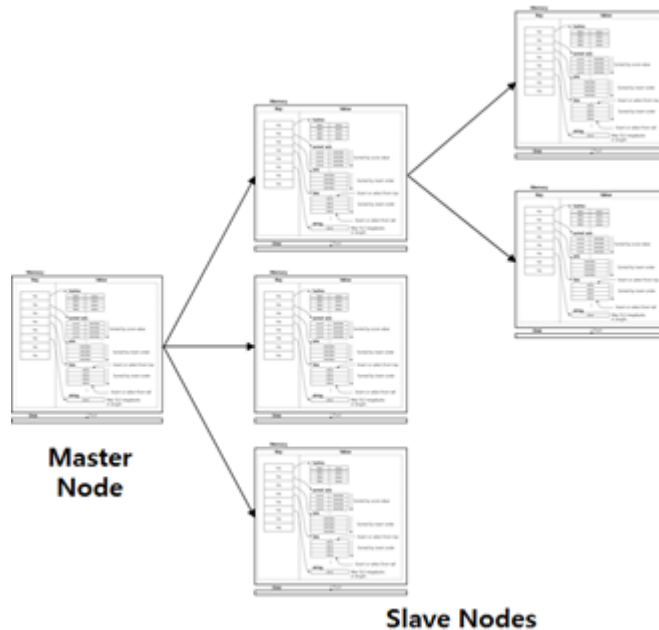
## PUB/SUB 모델

- 하나의 Client가 메시지를 Publish하면, 이 Topic에 연결되어 있는 다수의 클라이언트가 메시지를 받을 수 있는 구조
- 1:1 형태의 Queue 뿐만 아니라 1:N 형태의 Publish/Subscribe 메세징도 지원
- 일반적인 Pub/Sub 시스템의 경우 Subscribe 하는 하나의 Topic에서만 Subscribe하는데 반해서, Redis에서는 pattern matching을 통해서 다수의 Topic에서 message를 subscribe할 수 있음



## Replication

- Master/Slave 구조의 Replication(복제)를 지원. 즉 Master 노드에 write된 내용을 Slave 노드가 복제하는 Non-Blocking 구조
- 1개의 master node는 n개의 slave node를 가질 수 있으며, 각 slave node도 그에 대한 slave node를 또 가질 수 있음
- Master-Slave 구조에서 복제 연결이 되어있는 동안 Master 노드의 데이터는 실시간으로 Slave 노드에 복사
- 동시접속자수나 처리 속도를 증가시킬 수 있음



(Master 노드는 자식 프로세스를 만들어 백그라운드로 덤프파일을 생성하고 이를 Slave 노드에 보낸다.) 따라서 서비스를 제공하던 Master가 종료되더라도 Slave 노드에 애플리케이션을 재연결해주면 서비스를 계속 사용할 수 있다.)

Query Off Loading(master node는 write only, slave node는 read only 로 사용)을 통한 성능 향상. 특히 redis의 경우 value에 대한 여러가지 연산(합집합,교집합,Range Query)등을 수행하기 때문에, 단순 PUT/GET만 하는 NoSQL이나 memcached에 비해서 read에 사용되는 resource의 양이 상대적으로 높기 때문에 redis의 성능을 높이기 위해서 효과적인 방법

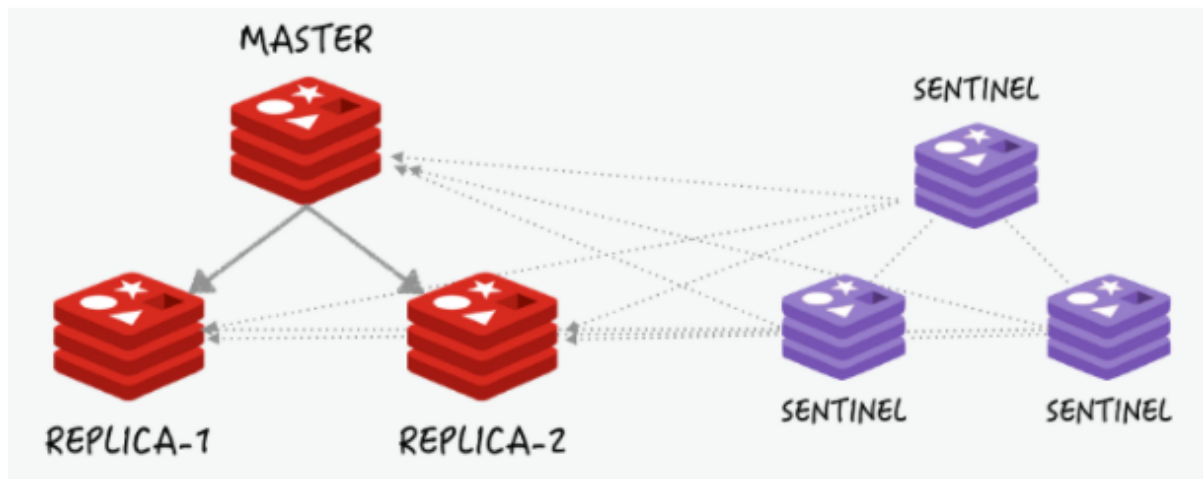
## Redis Mode

### 1. Standalone Mode

하나의 Redis 인스턴스로 서비스를 하는 방식

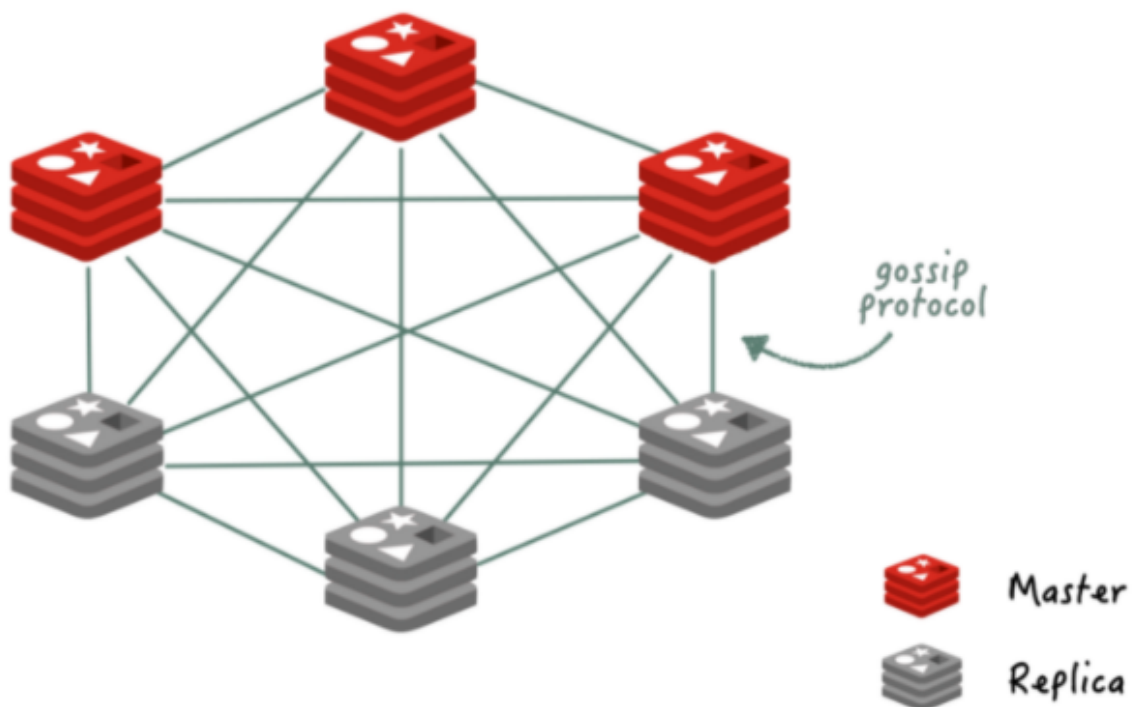
### 2. Sentinel Mode

- Redis의 Master/Slave를 모니터링하는 서버
- Sentinel은 Master와 Slave 노드를 계속 모니터링하며 장애상황에는 Slave 노드를 Master 노드로 승격시키기 위해 자동 Failover를 진행
- 정상적인 기능을 위해서는 적어도 세 개 이상의 홀 수의 Sentinel 인스턴스가 필요하고, 세 대의 Sentinel 노드 중 과반수 이상이 동의해야만 Failover를 시작



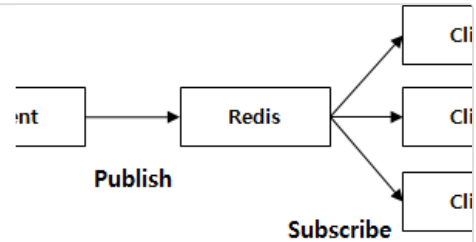
### 3. Cluster Mode

- 데이터셋을 여러 노드에 자동으로 분산하는 확장성 및 고성능의 특징
- 일부 노드가 종료되어도 계속 사용 가능한 고가용성의 특징
- 클러스터를 사용하기 위해서는 최소 세 개의 Master 노드가 필요, 모든 master와 replica 노드는 서로 연결되어 있음
- Replica 노드는 Master 노드의 정확한 복제본을 가지고 있음



### [Redis] Expire, Persistence, Pub/Sub Model, Replication

Redis는 데이터에 대해 생명주기를 정해서 일정 시간이 지나면 자동으로 삭제되게 할 수 있다. Redis가 expire된 데이터를 삭제하는 정책은 내부적으로 Active와 Passive 방식이 존재한다. Active 방식은 클라이언트가 데이터를 가져올 때, Redis가 expire된 데이터를 삭제하는 방식이다. <https://ozofweird.tistory.com/entry/Redis-Expire-Persistence?category=895904>



## Expiration

데이터에 대해서 생명주기를 정해서 일정 시간이 지나면 자동으로 삭제

데이터 삭제 정책은 두 가지 방법을 사용

### 1. Active

Client가 expired된 데이터에 접근하려고 했을 때, 그때 체크해서 지우는 방법

### 2. Passive

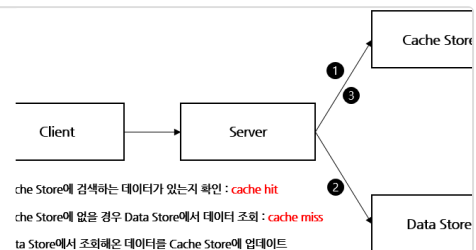
주기적으로 key들을 random으로 100개만 (전부가 아니라) 스캔해서 지우는 방식

참고 사이트 :

### Redis5 설계하기 총정리

대표적인 Cache Server 배치 전략으로는 Cache Aside 패턴과 Write Back 패턴이 있다. 특징 : 읽기에 적합. 캐시 장애 대비 구성. 정합성 문제 발생. 반복적인 호출에 적합 Cache Aside 패턴은 캐시로

👉 <https://waspro.tistory.com/697>



### [펌] [Redis] Redis란 ? 2번째


Redis 관련해서 기초 이해하기 위해 아래 블로그도 많은 도움이 되었다. 물론 해당 내용이 2012년도? 내용이기엔 변경된 부분도 조금 있다. 아래 내용의 경우 현재 Cluster 가 개발되어 있다. redis

👉 <https://hyunki1019.tistory.com/104?category=669533>




## [Redis] Redis 요약 및 정리

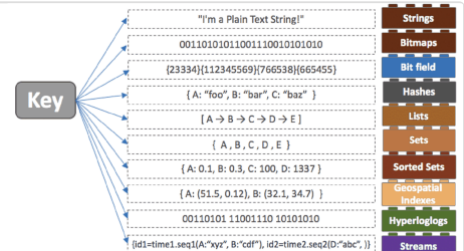
Redis 사용법에 대해 자세히 나열되어있는 사이트이다. (  
[redisgate.kr/redis/introduction/redis\\_intro.php](http://redisgate.kr/redis/introduction/redis_intro.php)) 1) Redis REmote  
Dictionary Server의 약자로, 메모리 기반의 키-값 구조 데이터 관리

 <https://ozofweird.tistory.com/entry/Redis-Redis-%EC%9A%94%EC%95%BD-%EB%B0%8F-%ED%95%84%EC%88%98%EB%A1%9C-%EC%95%8C%EC%95%84%EC%95%BC%ED%95%A0-%EB%82%B4%EC%9A%A9>

## [Redis] Redis 기초 정리하기


<https://www.youtube.com/watch?v=mPB2CZiAkKM&t> 본 포스트  
는 우아한테크 세미나의 우아한 Redis 영상을 정리한 내용입니다. 레  
디스는 In-Memroy 기반의 Data Structure Store이다. 인메모리 기반

 <https://icarus8050.tistory.com/124>



## Redis - 레디스 클러스터와 레디스 센티널 (집단지성)

레디스를 처음 설계할 때는 레디스를 매우 가볍고 빠르게 하는 것이 목  
적이었다. 지금까지 레디스를 사용할 수 있는 topology는 마스터/슬레  
이브의 topology였다. 이 topology는 마스터가 모든 쓰기 작업을 받

 <https://great-song2.tistory.com/8>



## Redis - Cluster & Sentinel 차이점 및 Redis에 대해

RDBMS만큼의 정합성과 영속성을 보장할 필요가 없는 데이터들을 빠  
르게 처리하거나 일정 기간동안만 보관하고 있기 위한 용도로 레디스  
(Redis), memcached 등의 in-memory 기반 저장소가 많이 사용된

 <https://coding-start.tistory.com/128?category=791662>



## 레디스 클러스터, 센티널 구성 및 동작 방식

RDBMS만큼의 정합성과 영속성을 보장할 필요가 없는 데이터들을 빠  
르게 처리하거나 일정 기간동안만 보관하고 있기 위한 용도로 레디스  
(Redis), memcached 등의 in-memory 기반 저장소가 많이 사용된

 <https://www.letmecompile.com/redis-cluster-sentinel-overview/>