

10장 . 클러스터

레디스를 확장하는 클러스터에서 지원하는 기능과 사용법에 대해 알아보시다.

레디스를 어떻게 확장할까 ?

스케일 업 vs 스케일 아웃

- 스케일 업: 서버의 하드웨어를 높은 사양으로 업그레이드
- 스케일 아웃: 장비를 추가해 시스템을 확장

스케일 업

- 레디스 운영 중 키의 `eviction` 이 자주 발생할 때
 - **eviction**: 레디스 인스턴스의 `maxmemory` 만큼 데이터가 차 있을 때 발생
 - `maxmemory` 값을 증가시키자 !

스케일 아웃

레디스의 처리량은 스케일 업만으로는 한계가 있다.

- 단일 스레드로 동작하기 때문에 서버에 CPU를 추가해도 여러 CPU 코어를 동시에 활용할 수 없다.

레디스 클러스터

기능

- 아키텍처 변경 없이 레디스 인스턴스 간 수평 확장 가능
- 데이터 분산 처리, 복제, 자동 페일오버

데이터 샤딩

저장소를 수평 확장하여 여러 서버 간 데이터를 분할하는 아키텍처 패턴

- 샤딩 관련 기능은 레디스 내부에서 자체적으로 관리
- 클러스터에서 데이터는 키를 이용해 샤딩됨
- 하나의 키는 항상 하나의 마스터 노드에 매핑
- 클라이언트에서 클러스터 내 특정 키가 어떤 마스터에 저장돼있는지 정보를 캐싱하여 키 조회 시간을 단축시킬 수 있음

고가용성

클러스터 버스 : 추가 TCP 포트로 클러스터 노드 간 독립적인 통신

- `cluster_bus_port` (기본값: 일반 포트 + 10000) 6379, 16379
- 클러스터는 모든 노드가 풀 메쉬 토폴로지 형태

😓 오버헤드가 걱정되는걸 . . .

- 가십 프로토콜과 구성 업데이트 메커니즘으로 클러스터가 정상적인 상태에서는 노드 간 너무 많은 메시지를 교환하지는 않는다.

동작 방법

해시슬롯

- 모든 데이터는 16,384개의 해시슬롯 중 하나에 저장된다. `CRC16(key) mod 16384`
- 마스터 노드는 해시슬롯을 나눠 가진다.

다중 키 커맨드

- 한 번에 여러 키에 접근해 데이터를 가져오는 커맨드
- 서로 다른 해시슬롯에 속한 키에서 사용할 수 없음
 - 클러스터는 키를 이용해 커맨드를 **처리할 마스터로 클라이언트의 연결을 리디렉션하기 때문에** 한 번에 2개 이상의 키에 접근해야 하는 커맨드는 처리할 수 없다.

해시태그

- 키에 대괄호를 사용하여 대괄호 사이에 있는 값을 이용해 해시될 수 있게 함
- 랜덤한 해시슬롯에 지정되는 것을 막아서 다중 키 커맨드를 사용할 수 있게 한다.

자동 재구성

- 복제와 자동 페일오버를 이용해 고가용성을 확보한다.
- 데이터를 저장하는 일반 레디스 노드가 서로 감시한다 일반 구조에서 센티넬의 역할
- 모든 노드는 클러스터 버스를 통해 통신한다.

종류

1. 자동 페일오버 : 복제본 노드를 마스터로 승격
2. 복제본 마이그레이션 : 잉여 복제본을 다른 마스터에 연결

😞 왜 하는가?

- 클러스터 내 마스터가 하나라도 정상 상태가 아닐 경우 전체 클러스터를 사용할 수 없다.
 - 데이터의 정합성을 위해 클러스터 전체 상태가 fail이 된다.
 - 다운타임을 줄이고 싶다면 자동 복제본 마이그레이션이 가능하도록 아무 마스터 노드에 복제본을 하나 더 추가하는 것을 고려하자.

자동 복제본 마이그레이션

- 레디스 클러스터가 마스터에 연결된 복제본 노드의 불균형을 파악해 남은 복제본을 다른 마스터의 복제본이 되도록 이동시키는 것
- 복제본 마이그레이션은 모든 마스터가 적어도 1개 이상의 복제본에 의해 복제되는 것을 보장한다.
- 클러스터 전체의 안정성을 향상시킨다.

기준

- 가장 많은 수의 복제본이 연결된 마스터의 복제본 중 하나가 옮겨진다.
- FAIL 상태가 아닌 복제본 중 노드 ID가 가장 작은 복제본이 이동될 노드로 선택된다.

```
cluster-allow-replica-migration yes
cluster-migration-barrier 1
```

- `cluster-allow-replica-migration` 옵션이 yes일 때 마이그레이션이 동작
- `cluster-migration-barrier` 옵션은 마이그레이션하기 전 마스터가 가지고 있어야 할 최소 복제본의 수 (초과하여야 동작함)

실행하기

클러스터 모드로 사용하기 위해 레디스 마스터 노드는 최소 3개가 있어야 한다.

실제 운영 목적으로는 3개의 마스터 노드에 복제본을 추가해 총 6개 노드로 클러스터를 구성한다.


```
# 클러스터 초기화
cluster-enabled yes

# 클러스터 생성
redis-cli -cluster create [host:port] --cluster-replicas 1
# --cluster-replicas 1 : 마스터마다 1개의 복제본을 추가할 것
```

알 수 있는 내용

- 복제본 노드는 마스터와 동일한 데이터를 저장하기 때문에 해시슬롯을 할당받지 않는다.

클러스터 접근하기

- 레디스 클러스터에 접속하기 위해 클러스터 모드를 지원하는 레디스 클라이언트가 필요
 - 클러스터 모드를 지원하는 클라이언트만이 **리디렉션** 등의 기능을 제공하고 그래야 클러스터를 제대로 사용할 수 있다.
- redis-cli를 이용해 레디스 클러스터에 접근해서 커맨드를 수행하면
 - 해당 키가 들어가야 할 해시슬롯과 해시슬롯을 가진 레디스 마스터 노드의 ip를 응답한다. (**불편**)
- Jedis, Redisson 등의 레디스 클라이언트들은 클러스터 모드 기능을 제공한다. (redis-cli -c도 클러스터 제공)
 Lettuce도 제공!
- 리디렉션 기능이 있으면 **커넥션을 옮겨가는 과정 없이 캐싱된 노드로 바로 커맨드 보내 클러스터 성능을 향상**시킨다.

페일오버 테스트

실제 레디스를 운영 서비스에 투입하기 전 테스트를 해보자

커맨드를 이용한 수동 페일오버 발생

페일오버를 발생시킬 복제본 노드에서 커맨드 실행

```
192.168.0.55:6379 > INFO REPLICATION
# Replication
role:slave
master_host:192.168.0.11
master_port:6379
...
```

```
192.168.0.55:6379 > CLUSTER FAILOVER
OK
```

```
192.168.0.55:6379 > INFO REPLICATION
# Replication
role:master
connected_slaves:1
...
```

- 수동 페일오버가 진행되는 동안 기존 마스터에 연결된 클라이언트는 잠시 블락된다.
- 페일오버를 시작하기 전 복제 딜레이를 기다린 뒤, 마스터의 복제 오프셋을 복제본이 따라잡는 작업이 완료되면 페일오버를 시작한다.
- 페일오버가 완료되면 클러스터의 정보를 변경하고 클라이언트는 새로운 마스터로 리디렉션된다.

마스터 동작을 중지시켜 자동 페일오버 발생

직접 마스터 노드에 장애를 발생시켜 페일오버가 발생하는지 확인

```
# redis-cli -h (마스터 주소) -p (마스터 포트) shutdown
$ redis-cli -h 192.168.0.33 -p 6379 shutdown
```

복제본이 주기적으로 마스터 노드에 ping을 보내 응답 확인

- `redis.conf` 에 설정한 `cluster-node-timeout` (기본값:15000, 15초) 동안 응답이 오지 않으면 페일오버 트리거

레디스 클러스터 운영하기

클러스터 리샤딩

해시슬롯 중 일부를 다른 마스터로 이동하는 것

```
$ redis-cli --cluster reshard 192.168.0.66 6379
...
# 1. 이동시킬 슬롯의 개수 정하기
How many slots do you want to move (from 1 to 16384)? 100
# 2. 해시슬롯을 받을 노드의 ID 입력
What is the receiving node ID? ab23kg0a9sdf7ad6f89b6adf23a
...
# 3. 해시슬롯을 이동시킬 노드의 ID 입력
# all:모든 마스터 노드에서 조금씩 이동 / done:마스터 ID 하나씩 입력한 뒤
Source node #1: all
...
```

3번까지의 과정이 끝나면 리샤딩이 진행될 소스와 목적지 마스터 노드 정보를 확인하고 리샤딩 플랜을 보여준다. 리샤딩 작업은 중단 없이 진행될 수 있다.

```
# 조금 더 간단하게 리샤딩하기
redis-cli --cluster reshard <host>:<port> --cluster-from <node-id>
--cluster-to <node-id> --cluster-slots <number of slots> --cl
```

클러스터 확장

추가하고자 하는 레디스에는 데이터가 저장되지 않은 상태여야 한다. (비어있지 않으면 에러)

```
# 마스터로 추가하기
# 기존 노드의 상태를 확인한 뒤 새로운 노드를 추가한다.
$ redis-cli --cluster add-node <추가할 노드 IP:PORT> <기존 노드 ID>

# 노드 구성 확인
$ redis-cli cluster nodes

# 복제본으로 추가하기
```

```
# --cluster-master-id 옵션이 없으면 복제본이 적게 연결돼있는 마스터를 선택
$ redis-cli --cluster add-node <추가할 노드 IP:PORT> <기존 노드 IP:PORT>
--cluster-slave [--cluster-master-id <기존 마스터 ID>]
```

노드 제거하기

마스터 노드를 삭제하는 경우 노드에 저장된 데이터가 없는 상태여야 한다.

```
$ redis-cli --cluster del-node <기존 노드 IP:PORT> <삭제할 노드 IP:PORT>
...
>>> Sending CLUSTER FORGET messages to the cluster...
>>> Sending CLUSTER RESET SOFT to the deleted node.
```

CLUSTER FORGET

클러스터를 제거하기 위해 할 일

1. 제거될 노드에서 클러스터 구성 데이터를 지운다.
2. 클러스터 내 다른 노드들에게 해당 노드를 지우라는 커맨드를 보낸다.

CLUSTER FORGET <node-id> 커맨드를 수신한 노드는 노드 테이블에서 노드의 정보를 지운 뒤 60초 동안은 이 노드 ID를 가지고 있는 노드와 신규로 연결되지 않도록 설정한다.

CLUSTER RESET

제거될 노드에서 수행되는 커맨드

과정

1. 클러스터 구성에서 복제본 역할을 했었다면 노드는 마스터로 전환되고, 노드가 가지고 있던 모든 데이터셋은 삭제된다. 노드가 마스터이고 저장된 키가 있다면 리셋 작업이 중단된다.
2. 노드가 해시슬롯을 가지고 있었다면 모든 슬롯이 해제된다.
3. 페일오버가 진행되고 있었다면 페일오버에 대한 진행 상태가 초기화된다.
4. 클러스터 구성 내 다른 노드 데이터가 초기화된다. 기존에 클러스터 버스를 통해 연결됐던 노드를 더 이상 인식할 수 없다.
5. currentEpoch, configEpoch, lastVoteEpoch 값이 0으로 초기화된다.

6. 노드의 ID가 새로운 임의 ID로 변경된다.

옵션

- **HARD**: 1~6번까지 과정 수행
- (default) **SOFT**: 1~4번까지 과정 수행

클러스터로의 데이터 마이그레이션

싱글/센티널 구성의 레디스 인스턴스 → 클러스터 구성의 레디스로 마이그레이션
운영 중인 경우 소스 레디스에 연결된 클라이언트를 모두 중단시키는 것이 좋다.

- 마이그레이션 도중 원본 레디스 노드에서 변경된 데이터는 마이그레이션되는 클라이언트에 반영되지 않음

복제본을 이용한 읽기 성능 향상

레디스 클라이언트는 같은 데이터를 갖고 있는 복제본 노드여도 우선 마스터로 연결을 리디렉션한다.

읽기 성능 향상을 위해 복제본 노드를 **읽기 전용**으로 사용할 수 있다.

마스터에 데이터를 읽어가는 부하가 집중되는 경우 데이터 쓰기는 마스터에, 읽기는 복제본에서 수행할 수 있도록 커넥션을 분배할 수 있다.

```
$ redis-cli -h 192.168.0.55 -c
192.168.0.55:6379> readonly
OK
```

레디스 클러스터 동작 방법

하트비트 패킷

클러스터 노드들이 지속적으로 상태 확인을 위해 주고받는 패킷
일반적으로 주고받는 패킷에 가십 섹션이 추가된 형태

가십 섹션

패킷을 발신하는 노드가 알고 있는 클러스터 내 다른 노드 정보
수신 노드는 가십 섹션을 통해 다른 노드에 대한 정보를 얻을 수 있다.

1. 알지 못하던 다른 노드 받아들이기
2. 장애 감지

에포크

분산 환경에서 노드 간 구성의 **정합성**을 유지하기 위한 개념

클러스터에서 여러 **이벤트의 순서**를 나타내는 값

각 노드는 에포크 값을 가지고 이 값이 클수록 최신 구성을 갖고 있는 노드임을 의미

1. 클러스터를 생성할 때 모든 노드의 에포크 값은 0으로 시작
2. 다른 노드들과 통신하며 패킷에 들어있는 값을 확인
 - 수신받은 패킷의 에포크가 로컬 노드의 값보다 크면 수신한 에포크 값으로 업데이트
3. 모든 노드들이 끊임없이 패킷을 주고받으며 클러스터에서 가장 큰 에포크 값으로 통일됨

해시슬롯 구성 전파

하트비트 패킷

마스터 노드가 패킷을 보낼 때 항상 **자기가 갖고 있는 해시슬롯을 패킷에 추가한다.**

업데이트 메시지

수신한 패킷의 에포크 값이 오래됐다면 신규 에포크의 구성 정보를 포함한 업데이트 메시지를 노드에 보내서 **오래된 노드의 해시슬롯 구성을 업데이트** 시킨다.

노드 핸드셰이크

한 노드가 클러스터에 합류하기 위해 . . .

1. **CLUSTER MEET** 커맨드를 다른 노드에 보낸다.
2. 수신한 노드가 자신이 알고 있는 다른 노드들에게 전파한다.
3. 전파받은 노드가 신규 합류한 노드를 모르면 해당 노드와 **CLUSTER MEET** 을 통해 신규 연결을 맺는다.

CLUSTER MEET은 방향성이 없다 !

클러스터 라이브 재구성

클러스터 해시슬롯 마이그레이션 중에 . . .

1. 키를 읽는 쿼리가 들어오면 source 노드에서 수행한다.
2. 키를 쓰는 쿼리가 들어오면 target 노드에서 수행한다.

마이그레이션하는 동안 각 노드는 락이 걸려 경쟁 상황은 발생하지 않는다.

리디렉션

요청하는 해시슬롯이 다른 노드에 있다면 ?

MOVED

앞으로의 요청은 그 노드에 보내라

- 해시슬롯을 가지고 있는 마스터 정보를 반환하면 클라이언트는 해당 정보로 다시 데이터를 조회한다.
- 이 때 클라이언트는 해시슬롯 A가 B 노드에 존재한다는 것을 기억하고 이후 다시 조회할 때 리디렉션 과정을 생략하여 시간을 단축시킨다.
- 모든 클라이언트는 해시슬롯, 노드의 맵을 가지고 있어 올바른 노드를 직접 찾아갈 수 있다.

ASK

이번에만 그 노드에 보내고 다음에는 나한테 와

- 해시슬롯이 이동되는 과정에서만 발생한다.
 - A → B로 해시슬롯이 이동 중일 때 B에게 요청이 들어온다면, A로 데이터를 조회하지만 결과적으로 B로 해시슬롯이 이동할 것이기 때문에 다음 번 요청은 B로 보내는 게 맞다.
- 해시슬롯 맵을 업데이트 하지 않는다.

장애 감지와 페일오버

PFAIL

특정 노드에 `NODE_TIMEOUT` 시간 이상 동안 도달할 수 없는 경우

클러스터 내 모든 노드들은 다른 노드를 `PFAIL` 로 플래그할 수 있다.

FAIL

특정 노드가 다른 노드를 PFAIL 상태로 플래깅한 이후 일정 시간 내 또 다른 노드에서 플래깅한 노드에 대한 PFAIL, FAIL 알림을 받는 경우

페일오버 조건

- 마스터가 FAIL 상태
- 마스터는 1개 이상의 해시슬롯을 가짐
- 마스터와 복제가 끊어진지 오래 (파라미터로 조절 가능)

페일오버 과정

1. 복제본은 마스터로 선출되기 위해 현재 에포크 값을 1 증가시킨다.
2. 클러스터의 모든 마스터 노드에 FAILOVER_AUTH_REQUEST 패킷으로 투표를 요청한다.
3. 요청받은 마스터는 FAILOVER_AUTH_ACK 패킷으로 투표에 동의할 수 있다.
 - 다른 복제본 승격을 방지하기 위해 NODE_TIMEOUT*2 동안 다른 복제본에게 투표할 수 없다.
4. NODE_TIMEOUT*2 동안 과반수 이상의 ACK를 받지 못하면 페일오버는 중단된다.
 - 페일오버 중단 시 NODE_TIMEOUT*4 지연 후 새로운 투표를 시도할 수 있다.
5. 랜덤한 지연시간을 이용해 같은 마스터에 연결된 여러 복제본이 동시에 투표하는걸 방지한다.

$DELAY = 500ms + \text{랜덤 지연시간 (0~500ms)} + SLAVE_RANK * 1000 ms$

- 복제본끼리 메시지를 교환하여 가장 최근의 오프셋을 가진 복제본이 우선순위가 높다.