Redis CLUSTER Introduction

Redis CLUSTER Background

레디스 클러스터(cluster)에 들어가기 전에 Scale up, Scale out, Data Partitioning, Data Sharding, Topology에 대해 살펴봅니다.

성능 향상 방법 - How to improve performance

- **스케일 업(Scale Up)**: 단일 머신(machine)에 CPU, 디스크 등을 추가해서 성능을 향상하는 방법.

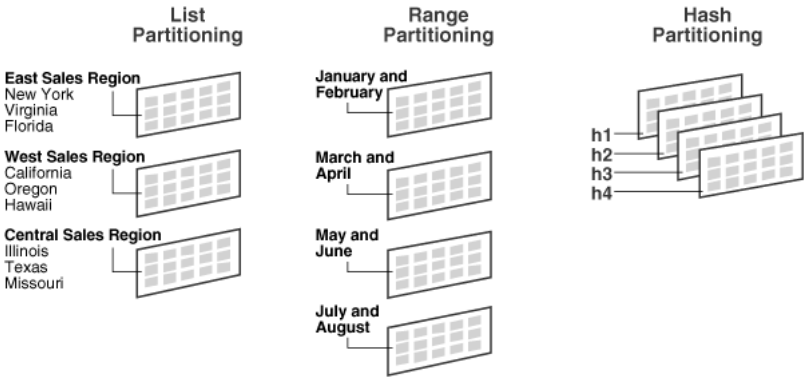
Scale Up

- **스케일 아웃(Scale Out)**: 적절한 성능의 머신(machine)을 추가해서 전체적인 성능을 향상하는 방법. ---» 소프트웨어가 scale out을 지원해야 합니다.

Scale Out

대량 데이터 처리, 저장 방법 - How to Process and Store Big Data

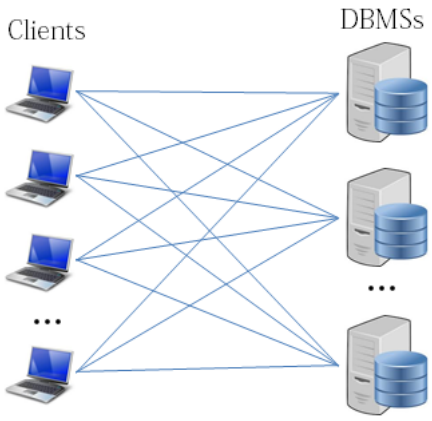
- **데이터 파티셔닝(Data Partitioning)**: 대량의 데이터를 처리하기 위해 DBMS 안에서 분할하는 방식입니다. 한 대의 DBMS만 설치하면 됩니다. 아래 그림은 테이블 하나를 여러가지 방법으로 분할한 것을 표시한 것입니다.



Data Partitioning Wiki Microsoft

  - Scale up ---» 1 Machine ---» 1 DBMS ---» Data Partitioning

- **데이터 샤딩(Data Sharding)**: 대량의 데이터를 처리하기 위해 여러 개의 DBMS에 분할하는 기술이다. DBMS안에서 데이터를 나누는 것이 아니고 DBMS 밖에서 데이터를 나누는 방식이다. 그러므로 샤드 수에 따라 여러 대의 DBMS를 설치해야 한다.

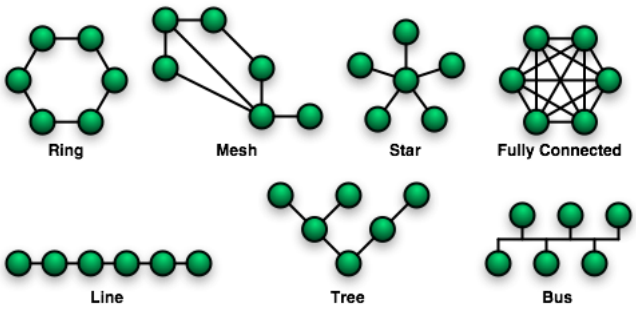


Data Sharding Wiki NAVER agildata

  - Scale out ---» n Machines/VMs ---» n DBMSs ---» Data Sharding

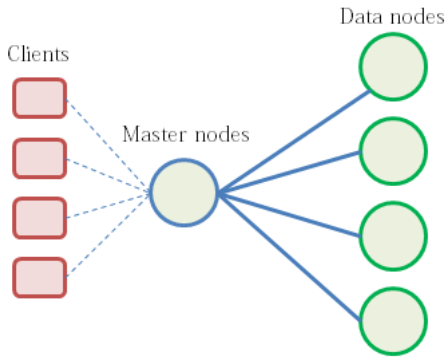
토폴로지 Topology

- 여러 가지 토폴리지

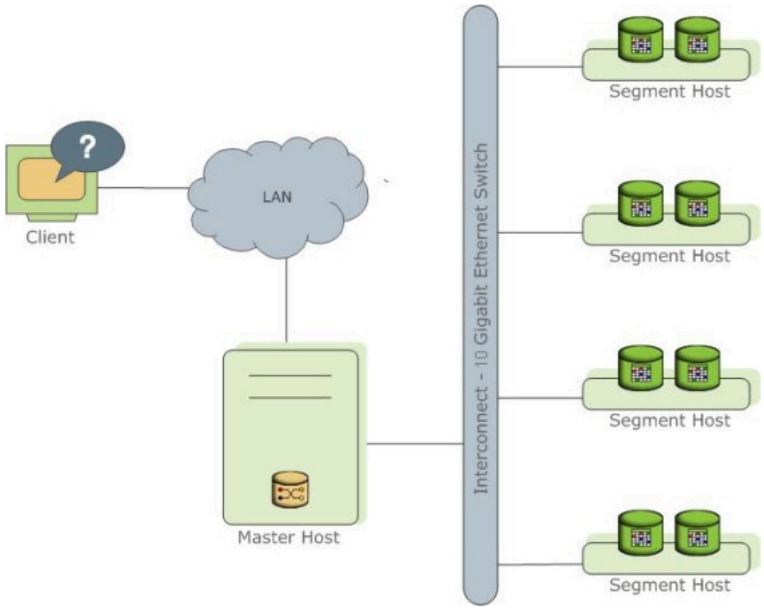


Network Toplolgies

- **단일 장애점(Single point of failure, SPOF)이 있는 토폴로지**: 스타(Star), 트리(Tree) ---» Greenplum, HBase

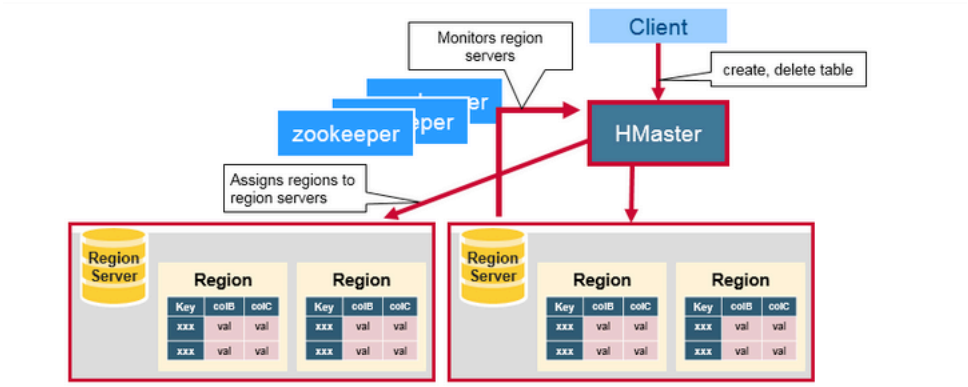


  - **Greenplum Database Architecture**: Active 마스터는 1개만 필요하고, 백업용 Standby 마스터를 두고 Active 마스터 다운 시 Standby 마스터가 마스터의 역할을 이어 받습니다. 마스터는 데이터가 어느 세그먼트 노드에 있는지와 세션(connection) 정보를 관리합니다. 클라이언트는 항상 마스터를 통해서만 쿼리를 수행할 수 있습니다.



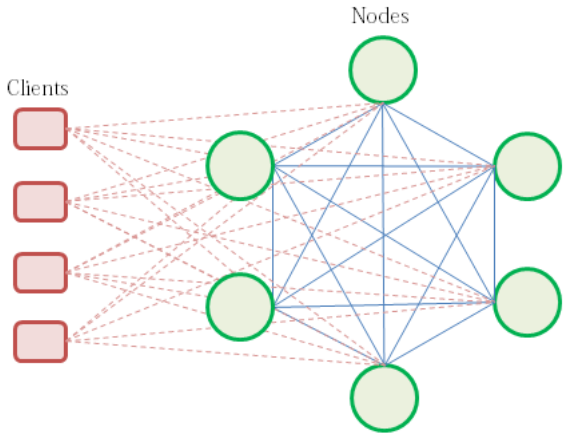
Greenplum Database Architecture

  - **HBase Architecture**: Hbase는 HDFS(Hadoop Distributed File System)에 기반합니다. 마스터는 Region 정보를 가지고 있고, HDFS의 name node에 위치합니다. 클라이언트는 마스터로 부터 Region 정보를 가져와서, 연결(connetion)은 마스터를 거치지 않고 Region 서버에 직접합니다. 따라서 마스터는 세션 정보를 가지고 있지 않습니다. 그러므로 Region 정보가 복제된 여러 개의 마스터를 둘 수 있습니다.
    HBase는 단일 장애점(SPOF)을 가지는 구조는 아니지만, 소수의 마스터가 다운될 경우 HBase를 사용할 수 없는 구조입니다. 굳이 말아자면 Multiple Points of Failure(MPOF) 아키텍처입니다.



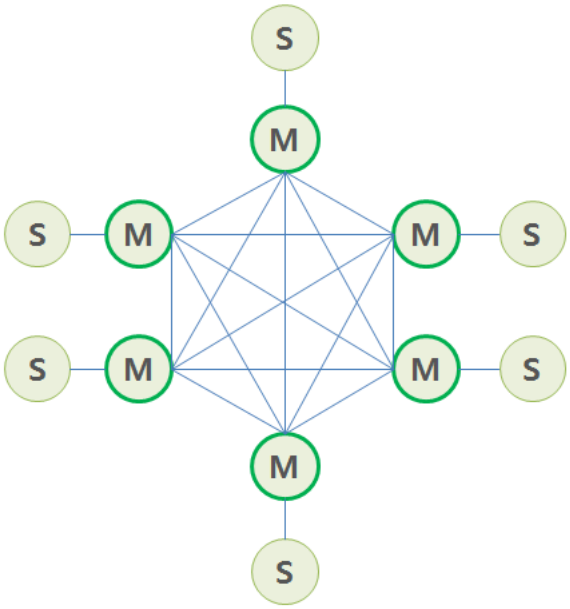
HBase Architecture

- **단일 장애점(Single point of failure)이 없는 토폴로지**: 메쉬(Mesh) ---» Redis Cluster



Fully Connected Mesh Topology

  - **Redis Cluster Architecture**: Clone 노드를 포함한 모든 노드가 서로를 확인하고 정보를 주고 받습니다. ---» Fully Connected Mesh Topology.
    Greenplum이나 HBase 같이 별도의 마스터 노드를 두지 않는 구조입니다. 모든 노드가 클러스터 구성 정보(슬롯 할당 정보)를 가지고 있습니다. 클라이언트는 어느 노드든지 접속해서 클러스터 구성 정보(슬롯 할당 정보)를 가져와서 보유하며, 입력되는 키(key)에 따라 해당 노드에 접속해서 처리합니다.
    일부 노드가 다운되어도 다른 노드에 영향을 주지 않습니다. 단, 과반수 이상의 노드가 다운되면 레디스 클러스터는 멈춥니다. 데이터를 처리하는 마스터 노드는 1개 이상의 복제 노드를 가질 수 있습니다.
    아래 그림에서는 복잡함을 피하기 위해 복제 노드와의 연결은 그리지 않았습니다.



Redis Cluster Architecture

키-노드 할당 방식 - How to Assign keys to Nodes

- **범위 할당 - Range Assignment**: Key의 범위로 노드를 할당합니다.
  예를 들어 key의 알파벳 첫 글자로 노드를 할당할 경우, 노드가 26개이면 A, B, C, D 순으로 노드를 할당하면 됩니다. 하지만 아래 영어 단어 비율표를 보면 P=11.53%이고 Y=0.48%로 각 노드의 데이터 크기가 매우 차이 납니다.
  노드가 10대 있는 경우, 한 노드에 2.6개의 알파벳이 할당됩니다. 그러므로 일률적으로 할당할 수 없고, 노드 당 2개 또는 3개의 알파벳을 할당해야 합니다. 아래 표와 같이 할당했다면

alphabets	node	alphabets	node	alphabets	node	alphabets	node	alphabets	node
A,B	1	C,D,E	2	F,G	3	H,I,J	4	K,L	5
M,N,O	6	P,Q	7	R,S,T	8	U,V,W	9	X,Y,Z	10

  - 이 표는 각 알파벳으로 시작하는 영어 단수 비율입니다.

A=7.94%	B=3.86%	C=8.50%	D=4.13%	E=4.09%
F=2.81%	G=2.95%	H=3.75%	I=3.38%	J=0.98%
K=1.32%	L=3.30%	M=5.90%	N=2.28%	O=3.38%
P=11.53%	Q=1.56%	R=3.43%	S=10.45%	T=6.14%
U=1.21%	V=2.66%	W=1.93%	X=0.61%	Y=0.48%
Z=1.43%				

  - 위 표를 기준으로 각 노드당 할당 비율을 계산해 보면, 가장 많이 할당된 노드는 20.02%, 가장 적게 할당된 노드는 2.52%로 약 8배 차이가 납니다.

alphas	node	alphas	node	alphas	node	alphas	node	alphabets	node
A,B	11.79%	C,D,E	16.72%	F,G	5.76%	H,I,J	8.11%	K,L	4.62%
M,N,O	11.56%	P,Q	13.09%	R,S,T	20.02%	U,V,W	5.80%	X,Y,Z	2.52%

  - 위 단어 비율로 가능한 균등하게 다시 할당해 보면, 가장 많이 할당된 노드는 12.63%, 가장 적게 할당된 노드는 7.11%로 약 0.5배 차이가 납니다.

alphas	node	alphas	node	alphas	node	alphas	node	alphabets	node
A,B	11.79%	C,D	12.63%	E,F,G	9.85%	H,I,J,K	9.43%	L,M,N	11.48%
O,Q,R	8.37%	P	11.53%	S	10.45%	T,U	7.35%	V,W,X,Y,Z	7.11%

  - 각 시도별 인구 수 분포 (2015년 기준): Range나 List로는 균등하게 할당하기 어려움.

시도	인구수	시도	인구수	시도	인구수	시도	인구수	시도	인구수

명		명		명		명		명	
서울	990만	부산	345만	대구	247만	인천	289만	광주	150만
대전	154만	울산	117만	경기	1천248만	강원	159만	충북	159만
충남	211만	전북	183만	전남	180만	경북	268만	경남	333만
제주	60만								

    - List 할당 방식은 Range 할당 방식의 한 종류입니다.
    - HBase: Range Sharding by Rowkey
      데이터를 Region 서버에 할당하는 것은 key의 값에 의존적인 Range 방식을 사용합니다. 따라서 데이터가 몰리는 Hot Region이 발생할 수 있기 때문에 Key 설계가 어렵습니다.
- **해시 할당 - Hash Assignment**: Key에 Hash 함수를 적용해서 노드를 할당합니다.
  노드 개수와 무관하고 모든 노드에 일정하게 데이터가 할당됩니다.
    - Redis Cluster: Hash Sharding by Key

CLUSTER Introduction

레디스 클러스터 목표

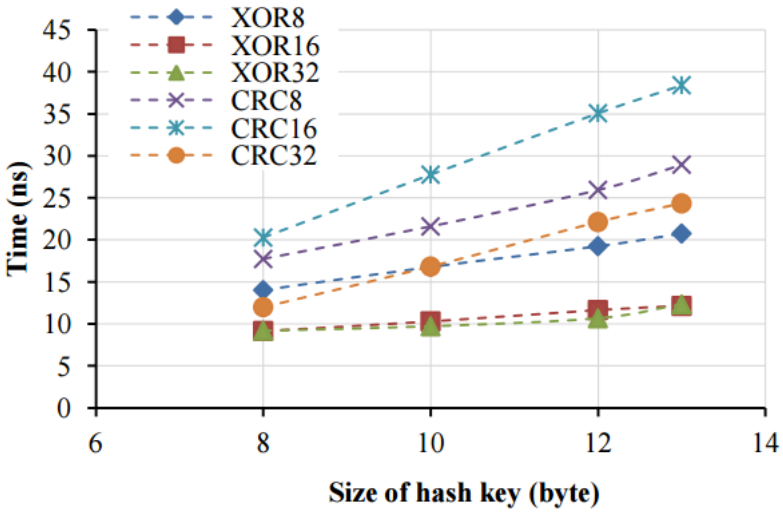
- 1000대의 노드까지 확장할 수 있도록 설계되었습니다.
- 노드 추가, 삭제 시 레디스 클러스터 전체를 중지할 필요 없고, 키 이동 시에만 해당 키에 대해서만 잠시 멈출 수 있습니다.

레디스 클러스터 키-노드 할당 방법

- 이 섹션에서는 레디스의 여러 노드에 데이터(key)를 어떻게 할당하는지에 대해서 알아봅니다.
- 좀 쉽게 설명하면, 레디스 서버가 10대 있을 경우 입력된 키를 그 중 한 서버로 들어가게 하기 위해서, 입력된 키에 해시 함수(Hash function)을 적용해 1 부터 100까지의 숫자로 만들어서, 1 부터 10까지는 1번 서버에 넣고, 11번 부터 20번까지는 2번 서버에 넣고, 계속 진행해서, 91번 부터 100번까지는 10번 서버에 할당합니다. 1 부터 100까지의 숫자를 슬롯(slot)이라고 합니다. 모든 서버는 자신이 보유하고 있는 슬롯 정보를 가지고 있으며, cluster nodes 명령으로 각 서버에 할당된 슬롯 정보를 얻을 수 있습니다.
- 레디스 클러스터는 16384개의 슬롯을 사용합니다. 슬롯 번호는 0~16383이고, 이 슬롯을 노드에 할당합니다. 레디스 노드가 3개 일 경우, 1번 노드는 0-5460, 2번 노드는 5461-10922, 3번 노드는 10923-16383 슬롯을 가지게 된다. 물론, 슬롯을 노드에 할당하는 것은 레디스 클러스터 관리자가 합니다.
- 해시 함수는 CRC16 function을 사용합니다.
  CRC16(Cyclic redundancy check) Wiki English
  CRC16(Cyclic redundancy check) Wiki 한글

    HASH_SLOT = CRC16(key) mod 16384

CRC16 Hash function time



CRC16 hash function 한글설명1 한글설명2 Vladut vs Salvatore 논쟁

레디스 서버 역할

- 레디스 서버는 마스터 또는 복제(슬레이브)입니다. 한 마스터 다운 시 다른 마스터들이 장애조치(failover)를 진행합니다. 따라서 레디스 클러스터에서는 별도의 센티널이 필요하지 않습니다.

레디스 클러스터 제한 사항

- 기본적으로 멀티 키 명령(operation)을 수행할 수 없습니다. 예를 들어, MSET key1 value1 key2 value2, SUNION key1 key2, SORT 이런 명령은 클러스터에서 사용할 수 없습니다. 하지만 hash tag를 사용하면 사용할 수 있습니다. Hash tag는 키의 일부를 {}로 감싸는 것입니다. 예를 들어, {user001}.following과 {user001}.followers는 같은 슬롯에 저장됩니다.
- 클러스터 모드에서는 DB 0번만 사용할 수 있습니다.
- 레디스 버전 3.0 이상에서 클러스터를 사용할 수 있습니다.
- 멀티 키 명령에 대해서: 엔터프라이즈 게이트 서버를 사용하면 멀티 키 명령을 사용할 수 있습니다. 자세한 내용은 Enterprise 게이트 서버를 보세요.

<< Redis Cluster Cluster Introduction Cluster Start >>

redisgate@gmail.com 02.503.2235
서울시 강남구 강남대로 342 역삼빌딩 5층 (역삼동) 우 06242