샤딩을 통한 시스템 확장

샤딩 개관

샤딩이란 무엇인가?

일반적으로 scale-up 에 의한 처리 성능 향상 정도 및 비용보다는 scale-out 에 의한 처리 성능 향상도가 더 높다. 이처럼 비교적 저렴한 서버를 활용하여 샤드 클러스터의 처리 성능을 선형적으로 늘리는 것을 샤딩이라 한다.

언제 샤딩을 할 것인가?

- 최초의 구성에서 많은 사용자와 데이터 증가시 scale-out 이 필요한 경우
    - application 에서 scale-out 은 한계와 구현이 어려움이 있다.
    - 서비스의 패턴과 부합해야 하지만 데이터베이스에서 지원하는 경우 많은 부담을 덜을 수 있다.

샤드 클러스터 구성 요소

샤드 애플리케이션 데이터 저장소

- 샤드 서버는 샤드 클러스터의 전부 혹은 일부 샤드 데이터를 가지고 있는 데이터 서버이다.
- 모든 샤드는 primary shard 를 가지며 shard 되지 않은 collection 등을 관리한다.

mongos 라우터

mongos 라우터의 역할

- 사용자 쿼리를 전달해야 할 샤드 서버를 결정하고 해당 샤드로 쿼리 전송
- 샤드 서버로부터 반환된 결과를 조합하여 사용자에게 결과 반환
- 샤드간 청크 밸런싱 및 청크 스플릿 수행
- 반환할 데이터 중 올바르지 않은 데이터(orphaned document)의 삭제
    - 리발란싱 진행중, 오류 등으로 잘못되거나 중복된 데이터를 validation 을 통해 제거 한다.
    - 사용자가 샤드 서버에서 직접 조회하면 안되는 이유

Config 서버

샤드 클러스터에 필요한 모든 정보를 보관한다. 다음은 config 서버에서 저장하는 collection 이다. | collection | 설명 | | databases | 샤드 클러스터가 가지고 있는 데이터베이스의 목록을 관리 | | collections | 샤딩되고 있는 collection 의 목록을 관리 | | chunks | 샤딩되고 있는 collection 의 chunk 정보를 관리 | | shards | 등록된 샤드 서버의 정보를 replica set 단위로 관리 | | mongos | config 서버와 연결된 모든 mongos 의 목록을 관리 | | setttings | chunk 의 리발란싱과 관련된 설정을 저장 | | version | 샤드 클러스터 모든 멤버의 전체 version 을 관리 | | lockpings | 샤드 클러스터 멤버와 config 서버와의 연결 기록 관리 | | locks | 샤드 서버와 mongos 는 작업시 발생하는 충돌을 방지하기 위해 config 서버의 locks collection 을 참조한다. | | changelog | config 서버의 메타 정보 변경 이벤트 이력 관리 |

SCCC (Sync Cluster Connection Config)

config 서버를 사용하는 mongos 혹은 샤드 서버가 구성된 모든 config 서버를 일일이 업데이트하는 2phase-commit 방식

- 구현이 어렵고 config 서버의 동기화 문제가 자주 발생된다.
- mongoDB 3.4 부터 제공되지 않는다.

CSRS (Config Servers as Replica Sets)

일반 샤드 서버 처럼 replica set 으로 구성하는 방식이다.

- 방식이 통일되면서 구축과 운용성이 개선되었다.

replica set 으로 config 서버를 구축하기 위해서는 다음 조건이 필요하다.

- config 서버는 반드시 wiredTiger 스토리지 엔진을 사용해야 한다.
- replica set 은 아비터를 가질 수 없다.
- replica set 은 지연된 멤버를 가질 수 없다.
- buildIndexes 가 true 인 멤버만 포함될 수 있다.
- 최소 3개 이상의 멤버로 구성해야 한다.

Read and Write

mongoDB 가 config 서버에 대해 read 및 write 동작을 할때는

- read concern 의 majority 기준으로 읽는다.
- write concern 의 majority 기준으로 기록 한다.

샤드 클러스터에서의 데이터 분산

샤드된 클러스터에 데이터를 분산시킬 수 있는 법

chunk

각 collection 이 샤드 키에 의해 조게지는 조각의 단위를 chunk 라고 한다.

- 한 collection 의 복수의 chunk 는 하나의 collection 파일에 기록된다.
    - 논리적인 단위로 구분하는 정보는 config server가 가지고 있다.
- 샤드 키의 원본 혹은 해쉬값의 일정 범위 를 가진다. (minkey, maxkey)
- 기본적으로 64mb 까지 저장되며 더 커지면 발란서에 의해 split 된다.

hashed sharding

샤드 키 값을 hash 하여 (md5) chunk 에 할당하는 알고리즘이다.

- hash 도 결국 range 와 같다.
    - 단 hash function 을 거쳐 보다 균등하게 분포되게 할뿐.
- 범위 검색시에는 broadcast query 를 하게 되는것으 단점
- 샤드 키 필드에 대해서 해시 인덱스를 생성해야 한다.
    - 기본적으로 단일 키에 대해 인덱스 생성이 가능
    - 4.4 부터 단일 키를

ranged sharding

샤드키 값을 기준으로범위를 나누고 사용자의 데이터가 어느 chunk 에 포함될지 결정하는 샤딩 알고리즘이다.

- targeted query 를 수행할 수 있다.
- 샤드 키가 데이터의 분포가 극단적인 경우 jumbo chunk 가 발생할 수 있다.
- 샤드 키가 데이터의 분포가 고른 날짜 혹은 단순 증가(auto increment) 패턴에 유리하다.

zone sharding

샤드키를 기반으로 복수의 샤드를 특정 zone 으로 묶어서 데이터를 분포를 관리할 수 있다.

샤드에 데이터베이스 분산 처리

컬렉션 내에서 샤딩

- 하나의 collection 을 여러 샤드 서버로 분산하기 위한 최소한의 단위를 chunk 라고 한다.

샤드 클러스터 질의 및 인덱싱

쿼리 라우팅

사용자의 질의를 mongos 가 받게 되면 다음과 같이 동작 한다.

1. 질의내 collection 의 chunk 메타정로를 검색 한다.
    1. mongos 에 있다면 참조한다.
    2. 없거나 오래되었다면, config 서버에서 참조한다.
2. 질의의 샤딩키 조건을 확인한다.
    1. 샤딩키가 있다면 해당하는 모든 shard 서버로 질의를 전달한다.
    2. 샤딩키가 없다면 모든 shard 서버로 질의를 전달한다.
3. shard 서버로 부터 결과를 병합하여 사용자에게 전달 한다.

Targeted Query

샤드키를 이용하여 전체 혹은 일부의 샤드 서버로 쿼리를 제한하는 쿼리

Broadcast Query

사용자 쿼리에 샤드키가 없어 모든 샤드 서버로 쿼리를 요청하는 쿼리

샤드 클러스터에서 인덱싱하기

- 샤드 클러스터에서 샤드 키 인덱스는 샤드키 혹은 샤드키가 포함된 compound 인덱스로 생성할 수 있다.
    - 샤드 키가 인덱스의 prefix 여야 한다.
- 데이터에 따른 샤드 키 인덱스 생성
    - 있는 경우 "sh.shardCollection()" 호출하기 전에 샤드 키 인덱스를 생성해야 한다.
    - 없는 경우 "sh.shardCollection()" 이 샤드 키로 인덱스를 생성한다.
- 즉 collection 에 샤드 키를 포함한 인덱스가 반드시 있어야 한다.

Unique Index

- 기본 샤드 키와 생성 조건은 같다.
- _id 인덱스는 샤드 키에 포함하지 않을 경우에만 샤드 단위로 유일성을 보장 한다.
- sh.shardCollection() 으로 생성하는 경우 unique 파라미터를 true 한다.

Unique Index 제약사항

- 샤딩 할 collection 이 이미 unique index 를 가지고 있으면 샤딩을 할 수 없다.
- 이미 샤딩된 collection 에 다른 필드로 unique index 를 만들 수 없다.
- index 로 지정되는 필드가 null 값일 경우 주의
    - duplicate 처리된다.

샤드 클러스터에서의 explain()

https://docs.mongodb.com/manual/reference/method/db.collection.explain/index.html
https://docs.mongodb.com/manual/reference/explain-results/index.html

사드 클러스터에서의 집계

샤드키 선택

Shard key carinality

샤드 키가 유니크한 값을 가지는 정도를 의미 한다. 유니크한 값이 많다면 그 만큼 balancer 가 많은 chunk 를 가질 수 있다. 또한 이 성질을 이용하여 수평적 확장 정도를 조절 할 수 있다.

Shard Key Frequency

샤드 키를 중복 정도를 의미한다. 중복된 데이터가 많다면 측정 chunk 에 데이터가 몰리게 된다.

Shard Key Monotonicity

지속적으로 증가하거나 감소하는 단조로운 패턴으로 특정 chunk 에 데이터가 몰릴 수 있다.(manual) 하지만 전체적으로 데이터량이 늘어나로 rebalancing 된다면 해결될 이슈라고 생각함.

실제 서비스에서의 샤딩

프로비저닝, 배포, 유지보수 등

Reference

- https://docs.mongodb.com/manual/core/sharded-cluster-config-servers/
- https://docs.mongodb.com/manual/sharding/#sharding-strategy
- https://docs.mongodb.com/manual/core/index-unique/#unique-index-and-missing-field
- https://docs.mongodb.com/manual/core/sharding-shard-key/#shard-key-indexes