샤딩이란?

Sharding 은 대용량 데이터를 여러 서버에 분산 저장하여 성능과 확장성을 높일 수 있는 수평적 확장 방법이다.

1. **확장성**: 데이터를 여러 서버에 분산 저장하여 저장 용량과 처리 성능을 확장할 수 있다
2. **고가용성**: 샤드 중 일부에 장애가 발생해도 전체 시스템이 중단되지 않으며, 복제를 통해 가용성을 유지할 수 있다
3. **성능 향상**: 데이터를 여러 샤드에 분산하여 읽기 및 쓰기 성능을 향상시킬 수 있다

샤딩의 주요 용어

용어	설명
샤드(Shard)	전체 데이터를 나누어 저장하는 MongoDB 서버 또는 복제 세트. 각 샤드는 전체 데이터의 일부분을 보유함.
Config 서버	샤딩 클러스터의 메타데이터(샤드 키 범위, 청크 위치 등)를 관리하는 중앙 서버. 일반적으로 3 개 구성.
Mongos	클라이언트의 요청을 적절한 샤드로 라우팅해주는 라우터 역할을 수행하는 프로세스.
샤드 키(Shard Key)	데이터를 어떤 샤드에 저장할지 결정하는 기준 필드. 효율적인 분산을 위해 적절히 설계되어야 함.
청크(Chunk)	샤드 키 값을 기준으로 분할된 데이터 블록 단위. 샤드 간 데이터 균형을 위해 이동될 수 있음.

샤딩 vs 복제

- 복제(Replication): 동일한 데이터를 여러 서버에 복사하여 데이터 가용성 확보

- 샤딩(Sharding): 데이터를 나누어 여러 서버에 저장하여 확장성과 성능 확보

MongoDB 샤딩 관리 및 진단 명령 메소드

명령어	설명
sh.addShard("shardReplSet/host:port")	새로운 샤드를 클러스터에 추가
sh.enableSharding("database")	특정 데이터베이스에 샤딩을 활성화
sh.shardCollection("db.collection", { shardKey: 1 })	컬렉션에 샤드 키를 설정하고 샤딩 시작
sh.status()	샤딩 클러스터의 현재 상태를 요약해서 확인
db.printShardingStatus(true)	샤딩 상태를 더 자세하게 출력 (청크 분포 포함)
sh.splitAt("db.collection", { shardKey: value })	특정 위치에서 청크를 분할
sh.moveChunk("db.collection", { shardKey: value }, "shardName")	청크를 특정 샤드로 이동
sh.removeShard("shardName")	클러스터에서 특정 샤드를 제거

샤딩 설정 단계

[1 단계] 데이터 경로 준비

각 샤드와 Config 서버의 데이터를 저장할 경로설정

```
mkdir c:₩db₩mydata01
mkdir c:₩db₩mydata02
mkdir c:₩db₩mydata03
mkdir c:₩db₩mongoc1
mkdir c:₩db₩mongoc2
mkdir c:₩db₩mongoc3
mkdir c:₩db₩log
```

[2 단계] 샤드 서버 설정

각 샤드 서버를 실행하고 레플리카 세트를 구성, 위해 세 개의 터미널(쉘)을 열고 각각의 샤드 서버를 실행

배치 파일 (.bat) 작성 _window : 샤드 서버를 실행하는 배치 파일 작성

- start_shard.bat 라는 이름으로 배치 파일을 작성
- 각 샤드 서버를 자동으로 실행하는 명령어를 추가

다음과 같이 한 줄로 작성

```
@echo off
echo Starting MongoDB Shard Servers...
REM mydata01 샤드 서버 시작
start mongod --shardsvr --dbpath c:₩db₩mydata01 --logpath c:₩db₩log₩shard1.log --port 27030 --nojournal --replSet RS

REM mydata02 샤드 서버 시작
start mongod --shardsvr --dbpath c:₩db₩mydata02 --logpath c:₩db₩log₩shard2.log --port 27031 --nojournal --replSet RS

REM mydata03 샤드 서버 시작
start mongod --shardsvr --dbpath c:₩db₩mydata03 --logpath c:₩db₩log₩shard3.log --port 27032 --nojournal --replSet RS

echo MongoDB Shard Servers started.
pause
```

@echo off
echo Starting MongoDB Shard Servers...
REM mydata01 샤드 서버 시작
start mongod --shardsvr --dbpath c:₩db₩mydata01 --logpath
c:₩db₩log₩shard1.log --port 27030 --nojournal --replSet RS

REM mydata02 샤드 서버 시작
start mongod --shardsvr --dbpath c:₩db₩mydata02 --logpath
c:₩db₩log₩shard2.log --port 27031 --nojournal --replSet RS

REM mydata03 샤드 서버 시작
start mongod --shardsvr --dbpath c:₩db₩mydata03 --logpath
c:₩db₩log₩shard3.log --port 27032 --nojournal --replSet RS

echo MongoDB Shard Servers started.
pause

start_shard.sh _mac

```
#!/bin/bash
echo "Starting MongoDB Shard Servers..."

# mydata01 샤드 서버 시작
mongod --shardsvr --dbpath /db/mydata01 --logpath /db/log/shard1.log --port 27030 --nojournal --replSet RS &

# mydata02 샤드 서버 시작
mongod --shardsvr --dbpath /db/mydata02 --logpath /db/log/shard2.log --port 27031 --nojournal --replSet RS &

# mydata03 샤드 서버 시작
mongod --shardsvr --dbpath /db/mydata03 --logpath /db/log/shard3.log --port 27032 --nojournal --replSet RS &

echo "MongoDB Shard Servers started."
```

셸 스크립트 실행 권한 부여: chmod +x start_shard.sh

[3 단계] 샤드 레플리카 세트 설정

1) MongoDB 클라이언트로 접속

```
mongo localhost:27030
```

2) **레플리카 세트 구성 및 초기화**:

```
use admin
var config = {
   _id: 'RS',
   members: [
      { _id: 0, host: 'localhost:27030' },
      { _id: 1, host: 'localhost:27031' },
      { _id: 2, host: 'localhost:27032' }
   ]
};
rs.initiate(config);
rs.status();
```

3) **샤드 삭제 및 추가**

rs.remove("localhost:27032")

4) 샤드 추가

rs.add("localhost:27032")

[4 단계] Config 서버 설정

Window : start_config.bat 이라는 배치 파일을 작성하고, 세 개의 Config 서버를 자동으로 실행

```
@echo off
echo Starting MongoDB Config Servers...

REM mongoc1 Config 서버 시작
start mongod --configsvr --dbpath c:₩db₩mongoc1 --logpath c:₩db₩log₩mongoc1.log --port 27011 --replSet CRS

REM mongoc2 Config 서버 시작
start mongod --configsvr --dbpath c:₩db₩mongoc2 --logpath c:₩db₩log₩mongoc2.log --port 27012 --replSet CRS

REM mongoc3 Config 서버 시작
start mongod --configsvr --dbpath c:₩db₩mongoc3 --logpath c:₩db₩log₩mongoc3.log --port 27013 --replSet CRS

echo MongoDB Config Servers started.
pause
```

Mac: Config 서버 자동 실행 (Shell Script)_ start_config.sh

```
#!/bin/bash

echo "Starting MongoDB Config Servers..."



# mongoc1 Config 서버 시작

mongod --configsvr --dbpath /db/mongoc1 --logpath /db/log/mongoc1.log --port
27011 --replSet CRS &



# mongoc2 Config 서버 시작

mongod --configsvr --dbpath /db/mongoc2 --logpath /db/log/mongoc2.log --port
27012 --replSet CRS &



# mongoc3 Config 서버 시작

mongod --configsvr --dbpath /db/mongoc3 --logpath /db/log/mongoc3.log --port
27013 --replSet CRS &



echo "MongoDB Config Servers started."
```

권한 부여 chmod +x start_config.sh

[5 단계] Mongos 라우터 설정

Mongos 는 클라이언트와 샤드를 연결하는 라우터 역할로 **Mongos 라우터**를 설정하여 샤드 간의 트래픽을 관리하고 데이터 분산 처리를 자동으로 수행할 수 있다.

-Windows Mongos 라우터 자동 실행 (Batch 파일) : start_mongos.bat

```
@echo off
echo Starting MongoDB Mongos Router...

REM Mongos 라우터 시작
start mongos --configdb CRS/localhost:27011,localhost:27012,localhost:27013 --port 27021

echo MongoDB Mongos Router started.
pause
```

Mac: Mongos 라우터 자동 실행 (Shell Script): start_mongos.sh

```
#!/bin/bash
echo "Starting MongoDB Mongos Router..."

# Mongos 라우터 시작
mongos --configdb CRS/localhost:27011,localhost:27012,localhost:27013 --port 27021 &

echo "MongoDB Mongos Router started."
```

권한 부여 chmod +x start_mongos.sh

[6 단계] 샤드 추가 및 샤딩 활성화

1. Mongos 에 샤드 추가

```
mongos> sh.addShard('RS/localhost:27030,localhost:27031,localhost:27032')
```

2. 데이터베이스에 샤딩 활성화

```
mongos> sh.enableSharding('blog')
```

3. 컬렉션에 샤드 키 설정

```
mongos> sh.shardCollection('blog.user', { 'userId': 1 })
```

[7 단계] 샤딩 상태 확인

1. 기본 샤딩 상태 확인

```
mongos> db.printShardingStatus()
```

2. 상세 샤딩 상태 확인

```
mongos> db.printShardingStatus(true)
```