Corporación Universitaria Iberoamericana

Facultad de Ingeniería de Software

Bases de datos avanzadas (28102024_C2_202434)

Actividad 3 - Conceptos y Comandos básicos del particionamiento en bases de datos NoSQL

Profesor:
Jorge Castañeda

Estudiantes:
Saulo Andres Franco Torres

15 de diciembre de 2024

Introducción.

La implementación de Sharding en MongoDB es una técnica fundamental para gestionar grandes volúmenes de datos distribuidos en múltiples servidores. En este documento, se describe detalladamente el proceso de configuración del entorno de Sharding utilizando Docker, una herramienta que facilita la creación y gestión de contenedores para ejecutar servicios de manera eficiente. A través de este enfoque, se establecerán los servicios de MongoDB necesarios para habilitar el sharding, incluyendo los servidores de configuración, shards y el router mongos. Además, se explicará la configuración de réplicas para asegurar la alta disponibilidad y la correcta distribución de los datos. Finalmente, se proporcionan los pasos para habilitar el particionamiento de bases de datos y verificar su correcta implementación, lo que resulta crucial para mejorar el rendimiento y escalabilidad en entornos de bases de datos distribuidas.

Estrategia de Particionamiento Horizontal (Sharding) para la Base de Datos torneo_deportes

El particionamiento horizontal, o **sharding**, es una técnica utilizada en bases de datos distribuidas para dividir los datos en fragmentos más pequeños (llamados **shards**), los cuales se almacenan en diferentes servidores o nodos. Esta estrategia es crucial para manejar grandes volúmenes de datos y garantizar un rendimiento óptimo a medida que la base de datos crece. En el contexto de un evento deportivo como el torneo de fútbol, es necesario elegir una estrategia de particionamiento que permita distribuir de manera eficiente los datos asociados a partidos, jugadores, equipos, etc., entre múltiples servidores.

Análisis de los Requerimientos

En el caso del **torneo_deportes**, la base de datos debe ser capaz de almacenar información sobre partidos, equipos, jugadores, resultados, estadísticas y otras entidades relacionadas. Como se trata de un evento deportivo, cada partido involucra equipos que juegan en diferentes fechas, lo que genera un volumen significativo de datos. Para manejar este volumen y permitir una rápida consulta y escalabilidad, la **estrategia de particionamiento** debe distribuir los datos de manera que las consultas más frecuentes, como las que buscan partidos de un equipo específico, se realicen de forma eficiente.

Estrategia de Particionamiento Propuesta

Se plantea una estrategia de particionamiento basada en el campo **equipo_id**, que será la clave de partición. Esto permite que los datos de los partidos se distribuyan entre los shards según el equipo que participe en cada partido. La elección de **equipo_id** se debe a que las consultas de la base de datos probablemente buscarán partidos, estadísticas y detalles relacionados con un equipo específico, lo que hace que este campo sea ideal para distribuir los datos y minimizar la sobrecarga de las consultas.

La estrategia propuesta sería la siguiente:

1. **Campo de Partición: equipo_id**
   Utilizamos el campo equipo_id, que identifica a cada equipo en un partido, como la clave de partición. Esto permitirá que los partidos en los que participe un mismo equipo se almacenen en el mismo shard, reduciendo así la cantidad de shards involucrados en las consultas relacionadas con ese equipo.

2. **Distribución de Datos:**
   - Cada shard se encargará de almacenar los partidos de un conjunto de equipos, dependiendo de cómo MongoDB divida los valores de equipo_id.

- De este modo, las consultas que busquen información sobre un equipo específico podrán ser manejadas por un solo shard, mejorando la eficiencia y reduciendo la latencia.

3. **Consideraciones Adicionales:**
   - **Balanceo de Carga:** MongoDB, por defecto, maneja el balanceo de carga entre los shards, asegurando que los datos se distribuyan equitativamente. Sin embargo, si se observa que algunos equipos tienen una cantidad desproporcionada de partidos (por ejemplo, equipos más populares o con más partidos jugados), podría ser necesario ajustar el shard key o utilizar una estrategia de particionamiento más avanzada, como el **rangos**.
   - **Escalabilidad:** Al dividir los datos de los partidos por equipo_id, la base de datos puede escalar horizontalmente de manera eficiente, añadiendo más shards a medida que el número de equipos y partidos aumenta.

Comandos para la Implementación del Particionamiento Horizontal (Sharding)

A continuación se describen los comandos necesarios para configurar el particionamiento de la base de datos torneo_deportes en MongoDB, utilizando Docker y los contenedores previamente definidos.

1. Conectar al Router MongoDB (mongos):

Primero, debemos acceder al contenedor de MongoDB Router (mongos), que es responsable de dirigir las solicitudes a los shards correctos. Para ello, utilizamos el siguiente comando:

```
docker exec -it mongos mongosh --port 20006
```

2. Habilitar el Sharding en la Base de Datos torneo_deportes:

El primer paso es habilitar el sharding para la base de datos torneo_deportes, lo que permitirá que MongoDB distribuya los datos de esta base de datos entre los shards.

```
sh.enableSharding("torneo_deportes");
```

3. Definir la Clave de Particionamiento:

A continuación, se define la clave de partición utilizando el campo equipo_id. Esto indicará a MongoDB cómo distribuir los datos de la colección partidos entre los shards. El comando será el siguiente:

```
sh.shardCollection("torneo_deportes.partidos", { "equipo_id": 1 });
```

- El comando sh.shardCollection toma dos parámetros:
- El primero es el nombre completo de la colección a particionar: "torneo_deportes.partidos".
- El segundo es el campo que se utilizará para el particionamiento: { "equipo_id": 1 }. El valor 1 indica que la distribución será ascendente según el valor del campo equipo_id.

4. Verificar el Estado del Sharding:

Después de habilitar el sharding, es importante verificar que el proceso se ha realizado correctamente y que la base de datos está utilizando los shards de manera adecuada. Para ello, se puede ejecutar el siguiente comando:

```
sh.status();
```

Este comando proporcionará un resumen detallado del estado del sharding, incluyendo información sobre los shards, las colecciones particionadas y el balance de los datos entre los shards.

5. Verificar la Configuración de las Réplicas:

Asegúrese de que las réplicas de los shards están correctamente configuradas para garantizar la alta disponibilidad de los datos. Ejecute el siguiente comando para verificar la configuración de las réplicas en los shards:

```
rs.conf();
```

Este comando devolverá la configuración de las réplicas para cada uno de los shards, asegurando que los nodos estén correctamente configurados para soportar fallos y garantizar la continuidad del servicio.

6. Verificación de la Distribución de Datos:

Finalmente, para asegurarse de que el particionamiento se ha realizado correctamente, se puede revisar cómo se distribuyen los datos entre los shards. Esto se puede hacer consultando la distribución de los datos con el siguiente comando:

```
sh.status();
```

Este comando también puede proporcionar información sobre cuántos documentos están en cada shard, lo que permite comprobar si la distribución está equilibrada y si el particionamiento está funcionando correctamente.

Resumen de Comandos

1. Conectarse al contenedor mongos:

```
docker exec -it mongos mongosh --port 20006
```

2. Habilitar el sharding para torneo_deportes:

```
sh.enableSharding("torneo_deportes");
```

3. Definir la clave de particionamiento (equipo_id):

```
sh.shardCollection("torneo_deportes.partidos", { "equipo_id": 1 });
```

4. Verificar el estado del sharding:

```
sh.status();
```

5. Verificar la configuración de las réplicas:

```
rs.conf();
```

6. Verificar la distribución de los datos:

```
sh.status();
```

Con estos comandos, se implementa un particionamiento horizontal efectivo para la base de datos torneo_deportes, lo que garantiza la escalabilidad y el rendimiento óptimo para el evento deportivo.

Configuración del Entorno del Sharding en MongoDB

Contenedor de Docker

Los comandos de ejecución se inicializarán por Docker que es una herramienta ágil para ejecutar múltiples comandos.

Entramos al CMD en modo administrador y nos ubicamos en la carpeta donde se encuentra el contenedor de Docker en mi caso guarde la carpeta en el índice de C:

```
cd C:\Docker
```

En esta ubicación iniciamos los servicios del contenedor con el comando:

```
docker-compose up -d
```

el -d hará que se ejecute en segundo plano.

Y para cuando se requiera detener los servicios se usa el comando:

```
docker-compose down
```

para verificar que los servicios de Docker este en ejecución podemos ejecutar el siguiente comando desde el CMD:

```
docker ps
```

o podemos ir a la aplicación Docker Desktop donde podemos ver los servicios en ejecución.

Contenido del contenedor de servicios.

```
services:
  # Config Servers

  config1:
    image: mongo:latest
    container_name: config1
    command: mongod --configsvr --replSet configReplSet --port 20001
    ports:
      - "20001:20001"

  config2:
    image: mongo:latest
    container_name: config2
    command: mongod --configsvr --replSet configReplSet --port 20002
    ports:
      - "20002:20002"

  config3:
    image: mongo:latest
    container_name: config3
    command: mongod --configsvr --replSet configReplSet --port 20003
    ports:
      - "20003:20003"

  # Shard 1
  shard1:
    image: mongo:latest
    container_name: shard1
    command: mongod --shardsvr --replSet shard1ReplSet --port 20004
    ports:
      - "20004:20004"

  # Shard 2
  shard2:
    image: mongo:latest
    container_name: shard2
    command: mongod --shardsvr --replSet shard2ReplSet --port 20005
    ports:
      - "20005:20005"

  # Mongos Router
  mongos:
    image: mongo:latest
    container_name: mongos
```

```
    command: >
      mongos --configdb configReplSet/config1:20001,config2:20002,config3:20003 --port
20006
    ports:
      - "20006:20006"
```

Despues de mantener ejecutado los servicios del contenedor de Docker procedemos a configurar el replicaSet

Entramos al a la replica 1 (principal)

```
mongosh localhost:20001
```

Inicializamos las Replicas

```
rs.initiate({
  _id: "configReplSet",
  members: [
    { _id: 0, host: "config1:20001" },
    { _id: 1, host: "config2:20002" },
    { _id: 2, host: "config3:20003" }
  ]
});
```

Entramos al shard1

```
rs.initiate({
  _id: "shard1ReplSet",
  members: [{ _id: 0, host: "shard1:20004" }]
});
```

luego entramos al shard2

```
rs.initiate({
  _id: "shard2ReplSet",
  members: [{ _id: 0, host: "shard2:20005" }]
});
```

Por ultimo entramos al orquestador

```
docker exec -it mongos mongosh --port 20006
```

y añadimos los shards

```
sh.addShard("shard1ReplSet/shard1:20004");
sh.addShard("shard2ReplSet/shard2:20005");
```

habilitamos el particionamiento en la base de datos.

```
sh.enableSharding("torneo_deportes");
```

verificamos las particiones

```
sh.status()
```

y las replicas

```
rs.conf()
```

VIDEO EXPLICATIVO DE LA ACTIVIDAD

https://github.com/ZauloFranco/Actividad---Sharding---Particionamiento

Referencias.

- libro Aramburu Cabo, M. J. y Sanz Blasco, I. (2012). Bases de datos avanzadas. D - Universitat Jaume I. Servei de Comunicació i Publicacions, https://elibro.net/es/lc/biblioibero/titulos/51741

- Capítulo 7 (Replicación) del libro de Sarasa, A. (2016) Introducción a las bases de datos NoSQL usando MongoDB. Editorial UOC, disponible en: https://elibro.net/es/lc/biblioibero/titulos/58524