Bases de Datos Avanzadas

Documento de Scripts: Configuración de Particionamiento en la Base de Datos NoSQL

Materia: Bases de Datos Avanzadas (28102024_C2_202434)

Estudiante: Lizeth Valentina Benitez ID de Estudiante: 100173953

Profesor: Jorge Isaac Castañeda Valbuena

Fecha: Diciembre de 2024

Introducción

En la actualidad, el manejo eficiente y escalable de grandes volúmenes de datos es una necesidad crítica para las aplicaciones modernas. Las bases de datos NoSQL, como MongoDB, se han convertido en una solución popular para satisfacer esta demanda, especialmente en entornos donde la flexibilidad, el rendimiento y la disponibilidad son factores claves. El particionamiento o *sharding* juega un papel fundamental en este contexto, ya que permite distribuir datos entre múltiples servidores, mejorando el rendimiento y la escalabilidad horizontal del sistema.

Este documento tiene como objetivo detallar el proceso de implementación de particionamiento horizontal (*sharding*) en la base de datos `torneo_tenis`, desarrollada como parte del caso práctico planteado. Se presentarán los comandos necesarios para configurar cada componente del clúster de MongoDB, incluyendo el servidor de configuración, los nodos de shard y el router o `mongos`. Además, se incluirán capturas de pantalla que evidencian los resultados obtenidos en cada etapa, lo cual proporciona transparencia y verificabilidad del proceso realizado.

El escenario que motiva esta configuración es el crecimiento esperado de los datos relacionados con un torneo de tenis, donde la base de datos debe soportar la carga generada por la gestión de jugadores, encuentros, posiciones y demás componentes del evento. La implementación del *sharding* no solo garantiza una distribución uniforme de los datos, sino también la capacidad de manejar de manera eficiente un aumento significativo en el número de consultas y transacciones concurrentes.

Este informe está estructurado en secciones que explican paso a paso la configuración de cada componente del clúster, la habilitación del particionamiento en la base de datos y la definición de estrategias de particionamiento específicas para las colecciones `jugadores` y `encuentros`. Al final del documento, se podrá constatar cómo la base de datos ha sido preparada para enfrentar los desafíos de escalabilidad y rendimiento en el evento deportivo.

1. Configuración del Servidor de Configuración

El servidor de configuración almacena metadatos del clúster y debe estar configurado como un replicaset.

Comando ejecutado:

mongod --configsvr --replSet configReplSet --dbpath C:\data\configdb --port 27019

Descripción:

- `--configsvr`: Indica que el nodo actúa como servidor de configuración.
- `--replSet configReplSet`: Configura el replicaset con el nombre `configReplSet`.
- `--dbpath`: Ruta al directorio donde se almacenan los datos.
- `--port 27019`: Especifica el puerto en el que escucha el servidor.

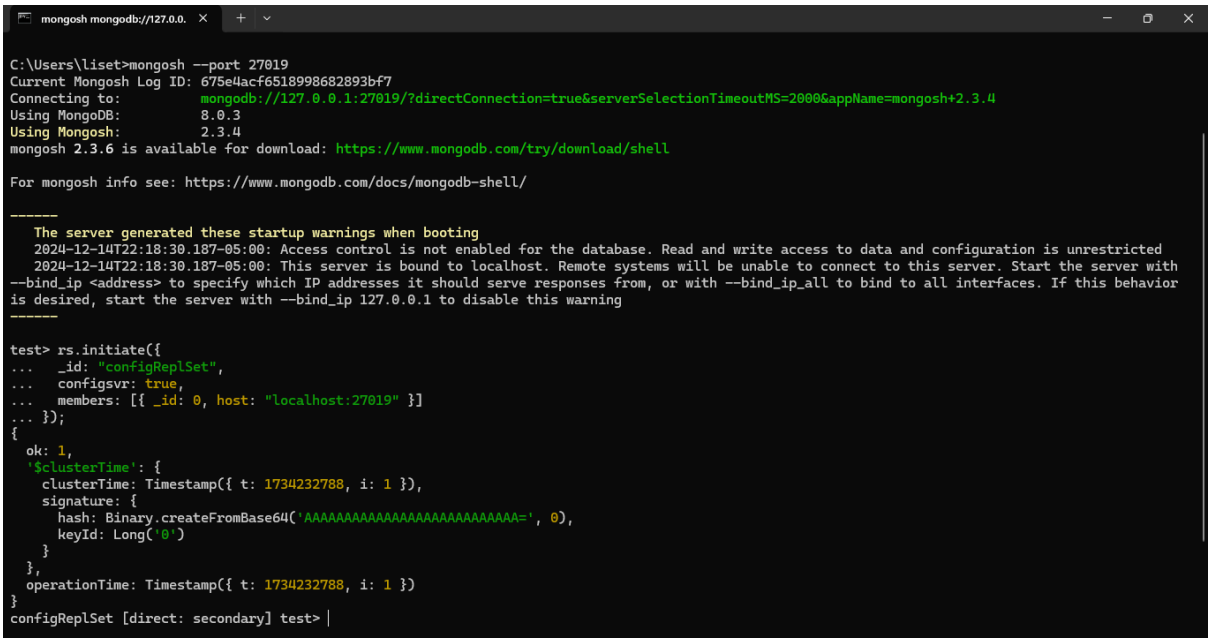
2. Inicialización del Replicaset del Servidor de Configuración

Comando ejecutado en `mongosh`:

```
rs.initiate({
  _id: "configReplSet",
  configsvr: true,
  members: [{ _id: 0, host: "localhost:27019" }]
});
```

Descripción:

- Inicializa el replicaset para el servidor de configuración.
- Define un miembro en el replicaset que escucha en `localhost:27019`.



3. Configuración de los Nodos de Shard

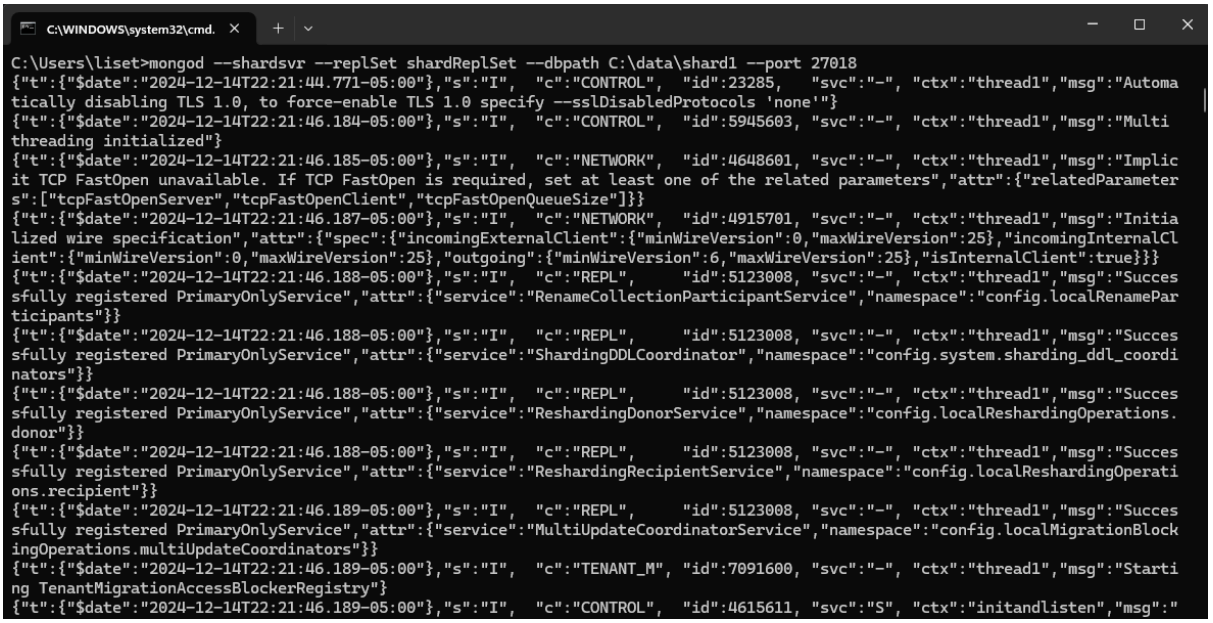
Cada nodo de shard se configura también como un replicaset para garantizar la disponibilidad de los datos.

Comando ejecutado para el primer nodo de shard:

mongod --shardsvr --replSet shardReplSet --dbpath C:\data\shard1 --port 27018

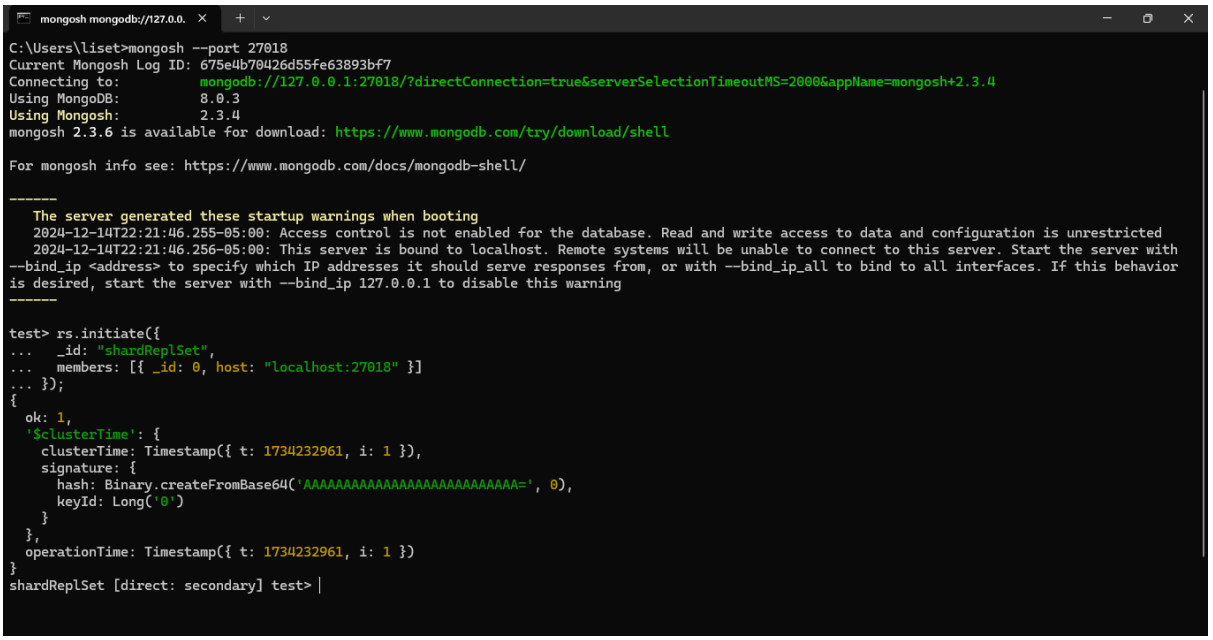
Descripción:

- `--shardsvr`: Indica que el nodo actúa como servidor de shard.
- `--replSet shardReplSet`: Configura el replicaset con el nombre `shardReplSet`.
- `--dbpath`: Ruta al directorio donde se almacenan los datos.
- `--port 27018`: Especifica el puerto del nodo.



Inicialización del replicaset del nodo de shard:

```
rs.initiate({
  _id: "shardReplSet",
  members: [{ _id: 0, host: "localhost:27018" }]
});
```



4. Configuración del Router (`mongos`)

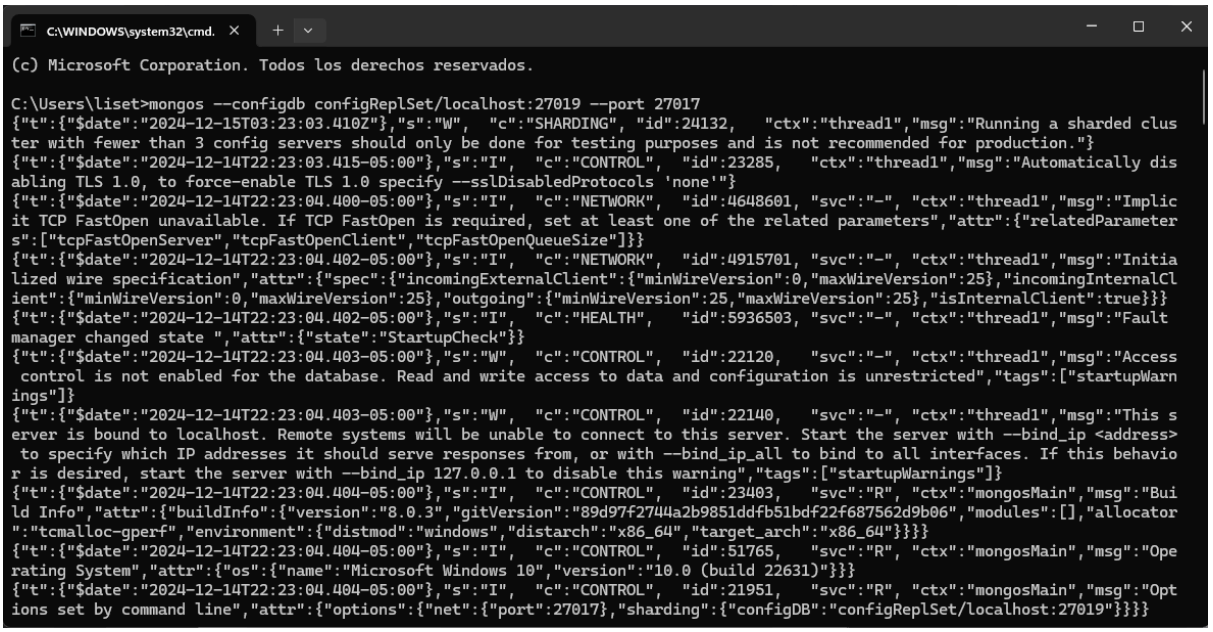
El router actúa como intermediario entre los clientes y los nodos del clúster.

Comando ejecutado:

mongos --configdb configReplSet/localhost:27019 --port 27017

Descripción:

- `--configdb configReplSet/localhost:27019:` Especifica la ubicación del servidor de configuración.
- `--port 27017:` Especifica el puerto del router.



5. Adición de Shards al Clúster

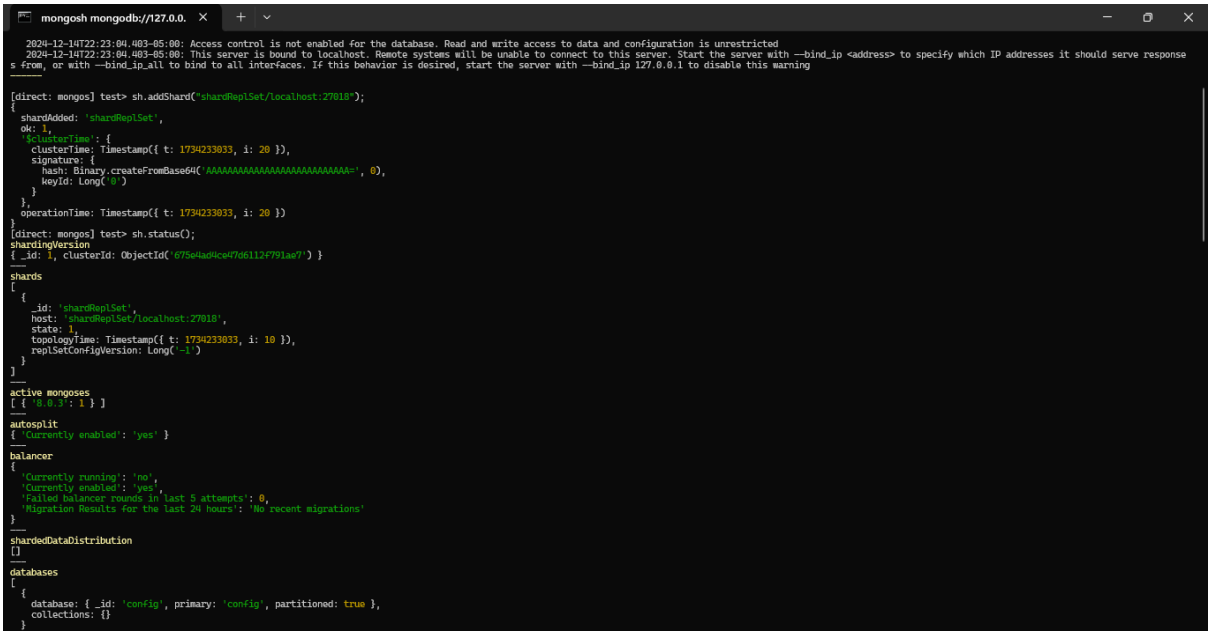
Una vez configurados los shards, se añaden al clúster utilizando el router.

Comando ejecutado en `mongosh`:

sh.addShard("shardReplSet/localhost:27018");

Descripción:

- Agrega el replicaset `shardReplSet` al clúster.



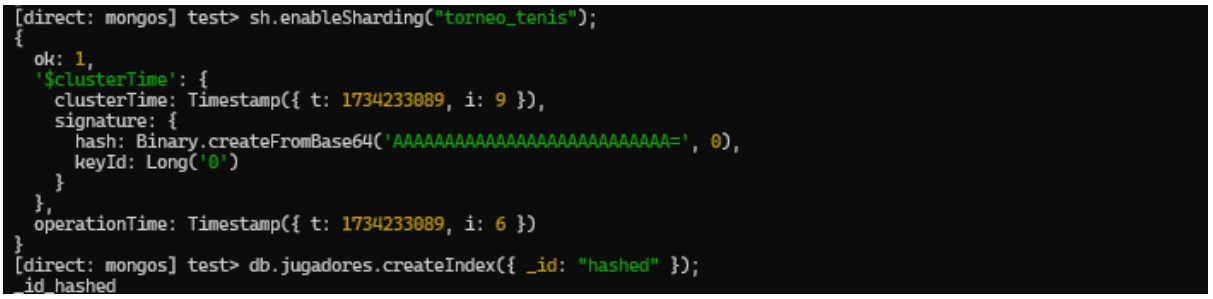
6. Habilitación del Sharding en la Base de Datos

Comando ejecutado en `mongosh`:

sh.enableSharding("torneo_tenis");

Descripción:

- Habilita el sharding en la base de datos `torneo_tenis`.



7. Configuración del Particionamiento de Colecciones

Particionamiento de la colección `jugadores`:

sh.shardCollection("torneo_tenis.jugadores", { _id: "hashed" });

Descripción:

- Particiona la colección `jugadores` utilizando la clave `_id` con un índice *hashed* para una distribución uniforme.

```
[direct: mongos] test> db.jugadores.createIndex({ _id: "hashed" });
_id_hashed
[direct: mongos] test> sh.shardCollection("torneo_tenis.jugadores", { _id: "hashed" });
{
  collectionsharded: 'torneo_tenis.jugadores',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1734233110, i: 51 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1734233110, i: 50 })
}
```

Particionamiento de la colección `encuentros`:

sh.shardCollection("torneo_tenis.encuentros", { fecha: 1 });

Descripción:

- Particiona la colección `encuentros` utilizando la clave `fecha` para consultas basadas en el calendario.

```
[direct: mongos] test> sh.shardCollection("torneo_tenis.encuentros", { fecha: 1 });
{
  collectionsharded: 'torneo_tenis.encuentros',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1734234433, i: 33 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1734234433, i: 33 })
}
```

Conclusión

Los comandos ejecutados y la configuración realizada permiten distribuir los datos de la base de datos `torneo_tenis` de manera eficiente en un clúster de MongoDB. Esto garantiza escalabilidad, balanceo de carga y tiempos de respuesta óptimos para las consultas y operaciones durante el evento deportivo.