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

Actividad 4 : Pruebas de particionamiento de bases de datos NoSQL

Cristian Leandro Pérez Peláez

Natalia Sierra Salamando

Facultad de Ingeniería, Corporación Universitaria Iberoamericana

Ingeniería De Software

Profesor: Jorge Castañeda

15 de diciembre de 2024

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

Actividad 4 : Pruebas de particionamiento de bases de datos NoSQL

Cristian Leandro Pérez Peláez y Natalia Sierra Salamando

Profesor: Jorge Castañeda

Corporación Universitaria Iberoamericana

Facultad de Ingeniería

Ingeniería De Software

Bogotá, Colombia

2024

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

Actividad 4 : Pruebas de particionamiento de bases de datos NoSQL

Introducción:

En este documento se definen los criterios de calidad relacionados con la redundancia y la disponibilidad 24x7 del sistema de bases de datos para gestionar un torneo de fútbol (el cual se realizó su estudio y creación de BD en la actividad 1), implementado con base de datos NO SQL MongoDB. Este sistema nos garantizar la continuidad operativa y la integridad de los datos mediante una configuración de replicación adecuada.

- Objetivos
    - Asegurar la disponibilidad continua de los datos mediante alta disponibilidad.
    - Reducir el riesgo de pérdida de información con redundancia.
    - Garantizar la eficiencia en la recuperación y sincronización de los datos tras un fallo.

1. **Requerimientos No Funcionales**

- Distribución de Datos:
    - Los datos de las colecciones principales deben distribuirse entre los nodos shard según criterios específicos como la ciudad del equipo, para garantizar una distribución equilibrada.
- Escalabilidad Horizontal:
    - El sistema debe permitir la adición de nuevos nodos shard sin interrumpir las operaciones de lectura/escritura.

- Alta Disponibilidad:
    - Cada shard debe tener un conjunto de réplicas para garantizar disponibilidad en caso de fallos.

- Minimización de la Latencia:
    - Las consultas deben dirigirse al shard correspondiente para minimizar la latencia en la recuperación de datos.

- Integridad de los Datos:
    - Los datos deben mantenerse consistentes a través de los shards y sus réplicas durante operaciones de lectura y escritura.

2. Comandos para la configuración del particionamiento horizontal
    - Configurar el clúster de sharding:

      Inicia los procesos para el servidor de configuración:

      mongod --configsvr --replSet cfg --port 27019 --dbpath /data/config --bind_ip localhost

    - Inicializa el conjunto de réplicas del servidor de configuración:

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

3. Inicia los nodos shard
    - Configura los procesos de MongoDB para cada shard:

mongod --shardsvr --replSet shard1 --port 27020 --dbpath /data/shard1 --bind_ip localhost

mongod --shardsvr --replSet shard2 --port 27021 --dbpath /data/shard2 --bind_ip localhost

- Inicializa los conjuntos de réplicas para los shards:

  rs.initiate({ _id: "shard1", members: [{ _id: 0, host: "localhost:27020" }] });

  rs.initiate({ _id: "shard2", members: [{ _id: 0, host: "localhost:27021" }] });

4. Configurar el router de sharding:
   - Inicia el proceso del mongos:

     mongos --configdb cfg/localhost:27019 --bind_ip localhost --port 27017

   - Añade los shards al clúster:

     sh.addShard("shard1/localhost:27020");

     sh.addShard("shard2/localhost:27021");

5. Habilitar el particionamiento y configurar la clave shard:
   - Habilita el sharding para la base de datos:

     sh.enableSharding("torneoFutbol");

   - Configura la clave shard para una colección:

     sh.shardCollection("torneoFutbol.equipos", { ciudad: 1 });

6. Pruebas para validar el particionamiento o sharding:

   - Prueba de distribución de datos:

     Paso: Inserta datos en la colección equipos con diferentes ciudades.

     Comando:

     db.equipos.insertMany([

       { _id: "equipo_1", nombre: "Águilas Doradas", ciudad: "Medellín" },

       { _id: "equipo_2", nombre: "Guerreros FC", ciudad: "Cali" },

       { _id: "equipo_3", nombre: "Tiburones", ciudad: "Barranquilla" }

     ]);

     Resultado Esperado: Los datos se distribuyen automáticamente entre los shards.

   - Prueba de consultas dirigidas:

     Paso: Realiza una consulta para un equipo de una ciudad específica.

     Comando:

     db.equipos.find({ ciudad: "Medellín" });

     Resultado Esperado: La consulta se dirige al shard correspondiente.

   - Prueba de escalabilidad:

     Paso: Añade un nuevo shard al clúster y verifica su integración.

     Comando:

     sh.addShard("shard3/localhost:27022");

     Resultado Esperado: Confirma la integración del cluster.

- Prueba de alta disponibilidad:

  Paso: Apaga un nodo en un shard y verifica la disponibilidad de los datos en sus réplicas.

  Comandos:

  Apagar nodo:

  ```
  sudo systemctl stop mongod --port 27020
  ```

  Verificar la disponibilidad de los datos:

  ```
  db.equipos.find();
  ```

  Reinicia el nodo apagado y verifica que se sincroniza correctamente con el resto del clúster:

  sudo systemctl start mongod --port 27020

  Resultado Esperado:

  - Durante la caída del nodo, los datos deben seguir siendo accesibles a través de las réplicas del shard.
  - Al reiniciar el nodo, debe sincronizarse con los otros nodos y recuperar el estado actualizado.

7. Resultados de las pruebas

   - Distribución de datos:

     Los documentos se distribuyen uniformemente entre los shards según la clave shard configurada.

   - Consultas dirigidas:

     Las consultas específicas se dirigen al shard correspondiente, mejorando la latencia.

   - Escalabilidad:

     La adición de un nuevo shard no interrumpió las operaciones existentes.

   - Alta disponibilidad:

     Durante la caída de un nodo, las réplicas asumieron el control sin pérdida de datos.

Repositorio GitHub:

https://github.com/clpp-dev/ACTIVIDAD3y4-FragmentacionBD