ACTIVIDAD 2

Juan David Ríos Escudero, Erick Santiago Bustos guzmán

Universidad Iberoamericana

Bases de datos avanzadas

Profesor Jorge Castañeda

01/12/2024

1. Introducción

En el entorno actual, la disponibilidad continua y la redundancia de los sistemas de información son elementos críticos para garantizar la eficiencia y la confiabilidad de las aplicaciones empresariales. En este trabajo se aborda la implementación de una estrategia de replicación en bases de datos, específicamente utilizando MongoDB, para un evento deportivo. Este sistema debe asegurar una disponibilidad 24x7 y proporcionar tolerancia a fallos mediante un Replica Set distribuido en tres nodos.

El objetivo principal de esta actividad es garantizar la integridad y la disponibilidad de los datos, incluso en situaciones de fallo de uno o más nodos. Para ello, se ha diseñado y configurado un entorno de replicación en MongoDB que cumple con los requerimientos no funcionales de redundancia y alta disponibilidad. A lo largo del trabajo, se documentan los criterios de calidad relacionados con estos aspectos, se detallan los comandos utilizados para configurar el entorno, y se describen los casos de prueba que validan la correcta implementación del sistema.

Finalmente, se analizan los resultados de los casos de prueba para asegurar que el sistema cumple con los niveles de servicio esperados, garantizando la continuidad operativa y la consistencia de los datos en todo momento. Este informe se acompaña de un video explicativo que detalla el proceso de configuración y los resultados obtenidos.

2. Requerimientos no funcionales

Requerimientos Relacionados con Redundancia

- **Mantenimiento de replicas:**

El sistema debe mantener una copia exacta de los datos por lo menos en dos nodos además del nodo primario, asegurando que cualquier movimiento en el nodo primario sea replicado inmediatamente en los nodos secundarios, esto garantiza que, si el primer nodo falla, los nodos secundarios contengan una copia completa y actualizada de los datos, en base a lo anterior cada movimiento o actualización está siendo respaldada.

- **Failover automático:**

En caso de que el nodo primario falle, uno de los nodos secundarios debe asumir automáticamente el rol de primario en menos de 10 segundos. Esto asegura la continuidad del servicio sin intervención manual. Después de desconectar el nodo primario, el sistema debe promover automáticamente un nuevo nodo primario y registrar el cambio en el log de replicación.

Requerimientos Relacionados con Disponibilidad 24x7

- **Disponibilidad Continua de la Base de Datos:**

La base de datos debe estar disponible para consultas y operaciones las 24 horas del día, los 7 días de la semana, con un tiempo de inactividad mensual máximo de 5 minutos. La alta disponibilidad es crucial para mantener un servicio continuo y evitar interrupciones, de

acuerdo con esto la base de datos en pruebas de mantenimiento debe seguir siendo accesible al menos uno de los tres nodos.

- **Mantenimiento con Mínima Interrupción**

El sistema debe permitir la realización de tareas de mantenimiento en un nodo sin afectar la disponibilidad general garantizando que los usuarios no experimenten interrupciones mientras se realizan actualizaciones o reparaciones y puedan seguir realizando consultas y operaciones.

- **Latencia de Replicación**

El retraso máximo permitido entre el nodo primario y los secundarios no debe exceder los 2 segundos asegurando la consistencia eventual de los datos, especialmente para aplicaciones que requieren datos actualizados en tiempo casi real.

3. Estrategia de replicación

La replicación en MongoDB es un proceso que adopta la duplicación de datos entre servidores. Esta estrategia garantiza alta disponibilidad, redundancia y la posibilidad de recuperación en caso de fallos. El modelo de Replica Set de MongoDB es el más adecuado para cumplir con los requerimientos de redundancia y disponibilidad 24x7.

Tipo de Replicación:

En MongoDB, se utiliza un Replica Set que consiste en un conjunto de instancias de MongoDB que mantienen la misma copia de los datos. Cada conjunto de réplicas consta de:

Un nodo primario (Primary): donde se realizan todas las operaciones de escritura.

Múltiples nodos secundarios (Secondary): copias de solo lectura del nodo primario. Los nodos secundarios replican los datos del primario.

Características del Replica Set:

- **Alta disponibilidad:**
  Si el nodo primario falla, uno de los nodos secundarios puede asumir el lugar automáticamente a primario, asegurando que el sistema siga disponible.
- **Redundancia:**
  Todos los datos del nodo primario están replicados en los nodos secundarios, lo que protege contra la pérdida de datos.

Comandos de replicación

- **Configuración del nodo primario:**
  mongod --replSet RS --dbpath="C:\DATA\db\mongodb\repset\rs-1" --port 27018

- **Inicialización del conjunto de réplicas:**
  rs.initiate()

- **Agregar nodos al conjunto de réplicas:**
  rs.add("localhost:27019")
  rs.add("localhost:27020")

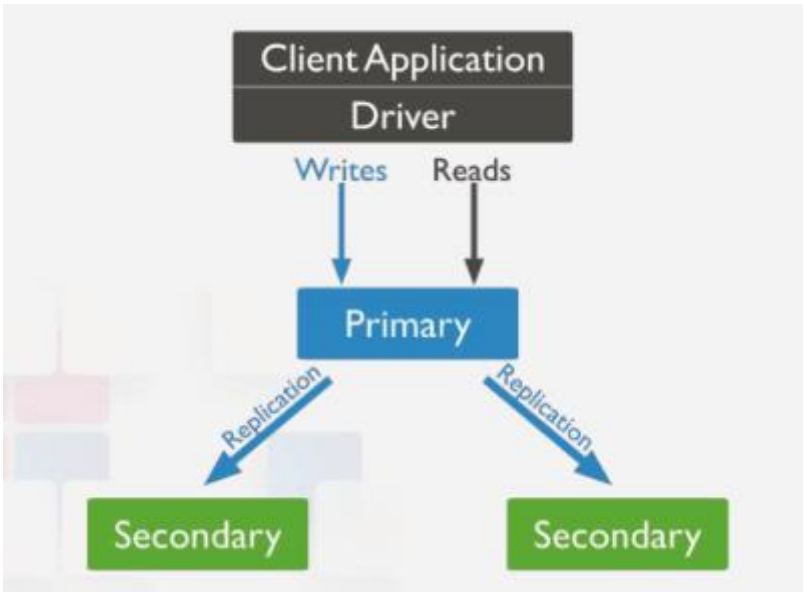
- Verificación del estado de la replicación:
  rs.status()

Estrategia propuesta:

La estrategia de replicación de este proyecto sigue una topología de Replica Set con 3 nodos, distribuidos de la siguiente manera:

- **1 nodo primario:**
  Acepta operaciones de escritura y lectura.
- **2 nodos secundarios:**
  Replican los datos del nodo primario. Estos nodos sirven para operaciones de lectura, asegurando que las consultas no sobrecarguen al nodo primario.

  Imagen1
  Imagen de muestra estructural de réplica SET

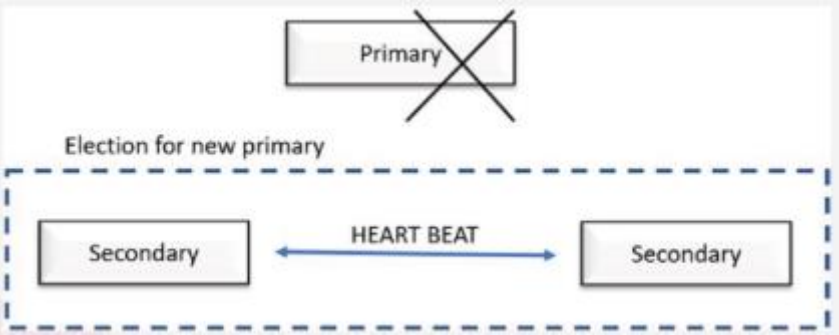


Estrategia de Failover Automático

- **Failover:**
  En caso de fallo del nodo primario, MongoDB utilizará un mecanismo de elección automática para seleccionar uno de los nodos secundarios como nuevo nodo primario. Este proceso se realiza en menos de 10 segundos.

Imagen2

Imagen de muestra de elección de nodo secundario por fallo del nodo primario



En caso de que falle el nodo primario se puede verificar la elección de otro nodo primarios con el comando **rs.status()**

Consistencia y Retraso en la Replicación

- **Consistencia eventual:**
  Es un modelo que garantiza que todos los nodos eventualmente tendrán los mismos datos, pero con un pequeño retraso entre el primario y los secundarios.
- **Latencia:**
  MongoDB proporciona herramientas para medir la latencia de replicación, y se establece un objetivo de retraso máximo de 2 segundos para las operaciones de replicación.

Estrategia de Lectura

- **Read preference:**
  Se utiliza para equilibrar la carga de trabajo. Por defecto, las lecturas se realizan desde el nodo primario, pero también se pueden realizar lecturas desde los nodos secundarios para distribuir la carga.

Estrategia de Backup

- **Backups periódicos:**
  Asegurarse de que se realicen copias de seguridad de los datos en intervalos regulares, preferentemente utilizando herramientas como Mongodump o soluciones en la nube como MongoDB Atlas.
- Los backups no deben afectar la disponibilidad del sistema, y se deben hacer cuando el sistema esté bajo mínima carga.

4. Casos de Prueba

Replicación de Datos:

En este caso de prueba buscamos insertar un nuevo dato a una colección en el nodo primario y deberá verse reflejado en los demás nodos.

Imagen3

Inserción de un nuevo deportista.

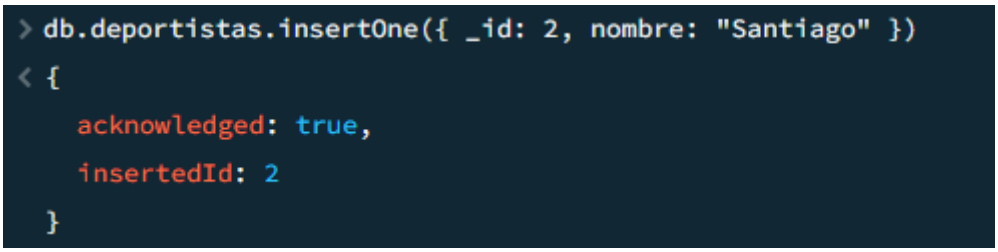


Imagen4

Integración de datos en los nodos secundarios

De acuerdo con los procesos anteriores, podemos decir que si se replican los datos insertados en el nodo primario a los nodos secundarios**.**

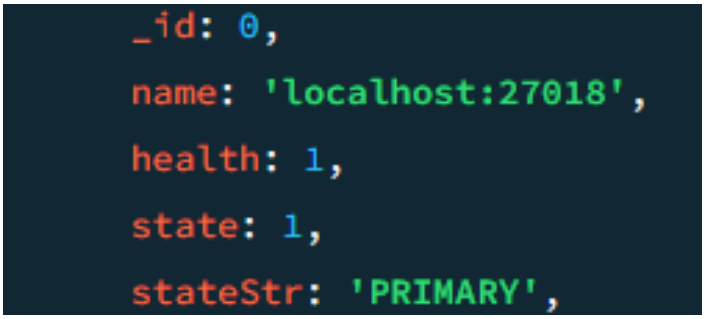
Failover Automático:

En caso de que el nodo primario falle, automáticamente se busca el reemplazo.

En el siguiente caso se simulará un fallo del nodo primario apagándolo temporalmente

Imagen5

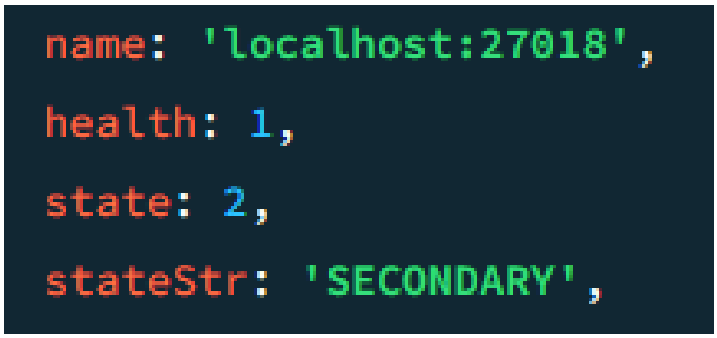
Muestra de nodo primario en el puerto 27018



Después de ejecutar el comando rs.stepDown() veremos como el nodo pasa de ser primario a ser secundario.

Imagen6

Muestra del cambio del nodo primario del puerto 27018 a nodo secundario.



De acuerdo con lo anterior veremos que el nodo del puerto 27019 tomo el lugar primario

Imagen7

Nombramiento de nodo primario al nodo del puerto 27019

De acuerdo con los anteriores procesos podemos confiar en el Failover automático de la replicación de nodos.

Comprobación de la Disponibilidad Continua 24x7

Con la inserción de datos anterior, nos pudimos dar cuenta que la base de datos se mantiene disponible

Imagen4

Integración de datos en los nodos secundarios

```
_id: 2
nombre : "Santiago"
```

También podemos hacer consultas aun estando un nodo inhabilitado, demostrando que se mantiene funcional aun habiendo nodos deshabilitados o con fallas.

Imagen8

Verificación de Consistencia de Datos

Para verificar que si hay consistencia de datos vamos a realizar una inserción de datos en el nodo primario, de acuerdo con eso vamos a realizar las consultas en los nodos secundarios

Imagen9

Inserción de un nuevo deportista

```
> db.deportistas.insertOne({ _id: 19, nombre: "Juan Rios" })
< {
    acknowledged: true,
    insertedId: 19
  }
```

Imagen10

Consulta del nuevo dato insertado a la base de datos

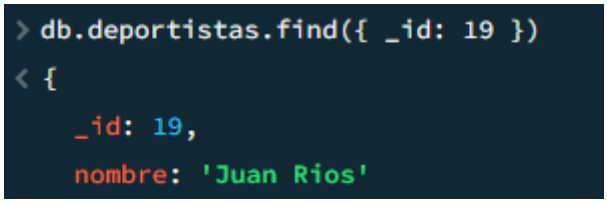
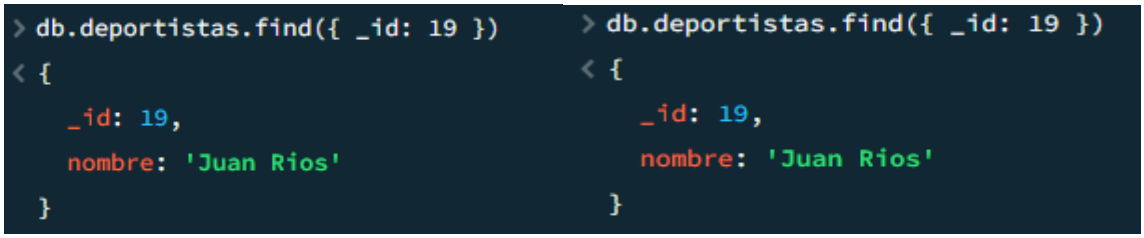


Imagen 11

Consultas en los dos nodos secundarios



De acuerdo con los procedimientos realizados, nos damos cuenta de que la consistencia de datos es correcta, ya que podemos hacer consultas en cualquiera de los nodos ya sea el primario o los secundarios.

Estos casos de prueba garantizan que el sistema:

- Replica correctamente los datos entre nodos.
- Puede realizar Failover automático sin pérdida de datos.
- Mantiene una alta disponibilidad 24x7.
- Mantiene la consistencia de los datos en todos los nodos.

5. Enlaces dirigidos al repositorio de GitHub y video explicativo de YouTube.
    - Enlace a GitHub: https://github.com/erick2215/Actividad-2.git
    -

Conclusión

La implementación de un Replica Set en MongoDB para el evento deportivo demostró ser una solución eficaz para cumplir con los requisitos de redundancia y disponibilidad 24x7. A través de la estrategia de replicación configurada, se garantiza que el sistema sea capaz de tolerar fallos, manteniendo la consistencia y la integridad de los datos en todo momento. Esto es crucial en escenarios donde la continuidad del servicio es indispensable.

Los casos de prueba diseñados y documentados validan que el sistema responde adecuadamente ante fallos en los nodos primarios, promoviendo automáticamente un nodo secundario a primario. Asimismo, se verificó que las operaciones de escritura y lectura se replican de manera eficiente, con una latencia mínima, asegurando que los datos estén siempre disponibles en todas las réplicas.

Este proyecto pone en evidencia la importancia de planificar y ejecutar estrategias de replicación en bases de datos distribuidas, especialmente en sistemas críticos que requieren alta disponibilidad. Los resultados obtenidos confirman que MongoDB, con su robusto mecanismo de replicación, es una herramienta confiable para soportar aplicaciones de misión crítica. Finalmente, el trabajo realizado cumple con los requerimientos planteados y se ajusta a los argumentos que demuestran el uso de una base de datos escalable.