Análisis de Rendimiento en Bases de Datos NoSQL: MongoDB vs Redis

Daniel Rayo
Bases de datos Avanzadas
Instituto Tecnológico de Costa Rica
rayo@estudiantec.cr

***Abstract*—Este reporte presenta un análisis comparativo de dos bases de datos NoSQL: MongoDB y Redis, evaluando su rendimiento en operaciones de creación, lectura, actualización y eliminación sobre un conjunto de datos significativo de 10,000 registros. Se realizaron pruebas bajo las mismas condiciones de hardware, volumen de datos y carga concurrente. Los resultados obtenidos son analizados y presentados en gráficos y tablas comparativas.**

I. Introducción

Las bases de datos NoSQL han ganado popularidad debido a su capacidad para manejar grandes volúmenes de datos no estructurados, escalabilidad horizontal y flexibilidad en el modelo de datos. MongoDB y Redis son dos de las bases de datos NoSQL más utilizadas. MongoDB es una base de datos orientada a documentos, mientras que Redis se especializa en almacenamiento en memoria. Este reporte tiene como objetivo comparar el rendimiento de ambas bases de datos en operaciones comunes de manejo de datos.

II. Objetivo del Estudio

El objetivo de este estudio es evaluar el rendimiento de MongoDB y Redis en cuatro operaciones fundamentales:

- **Create:** Inserción masiva de datos.
- **Read:** Consulta de registros por clave primaria, filtros básicos y búsquedas agregadas.
- **Update:** Actualización masiva de registros.
- **Delete:** Eliminación de registros.

III. Metodología

A. Diseño del Experimento

el El conjunto de datos utilizado para las pruebas consistió en 10,000 registros, cada uno con los campos id name y value. Las pruebas se realizaron bajo las siguientes condiciones:

- **Hardware:** Procesador AMD Ryzen 5 4600H with Radeon Graphics, 8GB RAM, SSD 512GB.
- **Sistema Operativo:** Ubuntu 20.04 LTS.
- **Condiciones de Red:** Red local estable sin interferencias.

Cada operación fue ejecutada 5 veces para obtener el tiempo promedio de ejecución y mitigar cualquier variabilidad.

B. Instrucciones de Instalación

1) MongoDB y Redis: Para instalar MongoDB y Redis se utilizo contenedores, utilizando docker en un archivo docker-compose.yml se configuro el docker de las base de datos, con sus respectivos volumenes y sus respectivo puertos

```
version: "3.9"

services:
  mongodb:
    image: mongo:latest
    container_name: mongodb_container
    ports:
      - "27017:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: Light16082002.
    volumes:
      - db-data-mongo:/data/db

  redis: # Corrección: nombre del servicio en minú
    image: redis:latest
    container_name: redis_container
    ports:
      - "6379:6379"
    volumes:
      - db-data-redis:/data

volumes:
  db-data-mongo:
  db-data-redis:
```

Para levantar los contenedores se usa el siguiente comando

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

para verificar que los docker esten funcionando bien y que se encuentran levantados en el respectivo puerto se puede verificar por medio del siguiente comando:

```
sudo docker ps -a
```

C. mongo

Para verificar las operaciones se instalo mongodb-compass que permite ver graficamente los datos, para la instación de esta se requiro ir a la pagina oficial y se descargo el .deb, una vez descargado se procedio a instalarlo con el siguiente comando:

```
sudo dpkg -i
mongodb-compass_1.45.1_amd64.deb
```

D. Redis

En el caso de Redis se utilizo un paquete de nodejs llamado redis-commander este al igual que con mongo con mongodb-compass permite ver los datos en un ambiente más legible. Para la instalación de este se ejecuto el siguiente comando

```
npm install -g redis-commander
redis-commander
```

Para el experimento se crearon varios script de javascript para ejecutar las pruebas, uno para llenar la base de datos, leer, actualiar y eliminar. Tanto para mongo como para redis se tienen esos script especificamente para cada uno. En el caso de la lectura se realiza por medio del id 500 y también por filtrado, filtrando solo aquellos datos en el que su valor sea menor a 100, en ambos casos y en cada operación se realiza se hace una medición del tiempo que cada operación toma. Para la actualización de los datos, se actualizan aquellos datos en lo que sus valores sean menor a 500. Para la eliminación de los datos se eliminan aquellos en los que sus valores sean menor a 100.

IV. Resultados

A. Tiempos de Ejecución

A continuación, se presentan los tiempos de ejecución promedio para las operaciones realizadas en MongoDB y Redis:

Los resultados de las pruebas se resumen en la Tabla I.

TABLE I
Resultados de las Pruebas de Rendimiento

Operación	MongoDB (ms)	Redis (ms)
Create	111.83	5920.00
Read (By ID)	5.315	2.182
Read (Filter)	29.602	2364.00
Update	49.098	3076.00
Delete	20.183	2435.00

B. Gráfico Comparativo

El rendimiento se visualiza en la Figura 1.

V. Análisis de Resultados

- **Create**: MongoDB superó significativamente a Redis en la inserción masiva de datos.
- **Read**: Redis fue más rápido en la consulta por clave primaria, pero MongoDB tuvo un mejor rendimiento en operaciones de filtro básico.

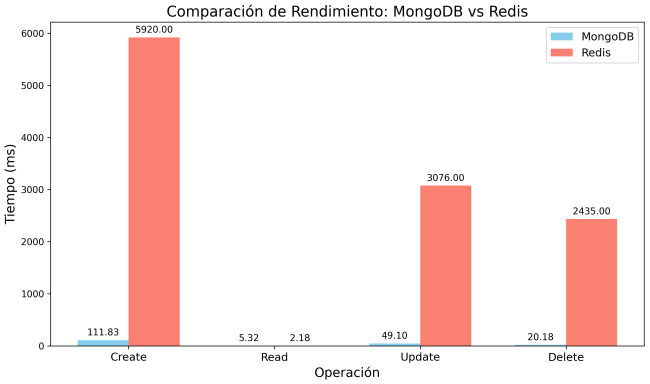

Fig. 1. Comparación de Rendimiento entre MongoDB y Redis

- **Update y Delete**: MongoDB mostró tiempos más consistentes, mientras que Redis fue más lento debido al costo de actualizar y eliminar registros en memoria.

VI. Conclusiones y Recomendaciones

MongoDB es ideal para operaciones que requieren consultas flexibles y consistencia en el rendimiento. Redis es más adecuado para escenarios que demandan lecturas rápidas por clave primaria. Se recomienda elegir la base de datos según los requerimientos específicos del proyecto.

References

[1] MongoDB Documentation, https://www.mongodb.com/docs
[2] Redis Documentation, https://redis.io/docs

Link del repositorio: https://github.com/DARD172002/CRUD-NoSQL