Evaluación Comparativa del Rendimiento de Sistemas NoSQL: Análisis de Operaciones CRUD y Configuración del Ambiente de Desarrollo (Enero de 2025)

Luis Andrés Salas Palma-2016182837, John Sebastián Ceciliano Piedra- 2023267790.

Resumen - En este trabajo se evalúa y compara el rendimiento de dos sistemas de bases de datos no relacionales (NoSQL) para las operaciones fundamentales de gestión de datos: Crear (Create), Leer (Read), Actualizar (Update) y Eliminar (Delete). El estudio incluye un análisis detallado de los tiempos de respuesta y la eficiencia de cada sistema bajo diferentes condiciones de carga. Adicionalmente, se documenta de manera estructurada el proceso completo de instalación, configuración del ambiente de desarrollo y la preparación de los entornos experimentales, proporcionando una guía reproducible para futuras investigaciones en este campo.

Para llevar a cabo la evaluación, se seleccionaron MongoDB y Cassandra como los sistemas NoSQL de interés debido a su amplia adopción en la industria y sus diferencias arquitectónicas. MongoDB, basado en documentos, y Cassandra, orientado a columnas, representan enfoques contrastantes en la gestión de datos no relacionales. Se diseñaron y ejecutaron pruebas controladas que simulan casos de uso reales, con el objetivo de analizar el rendimiento en escenarios variados y determinar las fortalezas y debilidades de cada sistema en las principales operaciones CRUD.

I. INTRODUCCION

En los últimos años, las bases de datos NoSQL han ganado relevancia como respuesta a las limitaciones de los sistemas relacionales tradicionales en entornos que requieren alta escalabilidad, flexibilidad y capacidad para manejar grandes volúmenes de datos no estructurados. Estas tecnologías están diseñadas para adaptarse a diversas necesidades, ofreciendo modelos de datos específicos, como bases de datos de documentos, de grafos, orientadas a columnas y basadas en claves-valor. Su adopción se ha disparado en aplicaciones como análisis de grandes datos, comercio electrónico, redes sociales y aplicaciones distribuidas.

Entre las diversas opciones disponibles, MongoDB y Cassandra destacan por su uso generalizado en la industria y por representar enfoques contrastantes en la gestión de datos NoSQL. MongoDB, una base de datos orientada a documentos, es conocida por su facilidad de uso, soporte para datos semi-estructurados en formato JSON y su flexibilidad para desarrolladores. Por otro lado, Cassandra, una base de datos orientada a columnas, ofrece un modelo distribuido que prioriza la alta disponibilidad, tolerancia a fallos y el manejo eficiente de datos masivos a gran escala.

La selección de estas dos tecnologías para este estudio se basa en su popularidad, diferencias arquitectónicas y capacidades ampliamente reconocidas. Estas características permiten realizar una comparación robusta que abarque tanto el manejo de datos estructurados como las demandas de entornos distribuidos de alto rendimiento. Este trabajo tiene como propósito evaluar el rendimiento de ambas plataformas en las principales operaciones CRUD (Create, Read, Update, Delete), identificando sus fortalezas y limitaciones para diferentes escenarios de uso.

II. PROCESO DE INSTALACION

A. MongoDB

La instalación de MongoDB se realizó siguiendo las instrucciones del tutorial disponible en YouTube (https://www.youtube.com/watch?v=eKXIxSZrJfw). Como primer paso, se descargó e instaló el software desde su página oficial, utilizando un gestor de paquetes adecuado para el sistema operativo (por ejemplo, apt en Linux o choco en Windows). Una vez completada la instalación, se verificó que el servicio estuviera corriendo correctamente en el sistema. Para la gestión de la base de datos, se utilizó MongoDB Compass, una herramienta visual que facilita la administración y consulta de datos. Dentro de este entorno, se creó una base de datos denominada TransaccionesDB, la cual contiene una colección llamada Transacciones. Finalmente, se configuraron las rutas en el servidor (http://localhost:3000) para realizar

pruebas iniciales de las operaciones CRUD (Create, Read, Update, Delete) en MongoDB, garantizando la correcta conexión y funcionamiento del sistema.

B. Cassandra

La instalación de Cassandra se llevó a cabo siguiendo el tutorial disponible en YouTube (https://www.youtube.com/watch?v=4tIiUDXFxxg&t=143s). Inicialmente, se descargó e instaló Apache Cassandra desde el sitio oficial, asegurando que todos los requisitos previos estuvieran configurados correctamente. Una vez instalado, el servidor se inició desde la consola utilizando el comando correspondiente en la carpeta bin. Posteriormente, se accedió a la consola interactiva cqlsh para ejecutar los comandos necesarios para la configuración inicial.

Se creó un keyspace llamado TransaccionesDB utilizando el comando:

```
CREATE KEYSPACE TransaccionesDB WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
```

Este comando configuró un espacio de trabajo con una estrategia de replicación simple y un factor de replicación de 1. Después, se seleccionó el keyspace con el comando USE TransaccionesDB; para trabajar en este entorno. Dentro del keyspace, se definió una tabla denominada Transacciones, que incluye las columnas IdTransaccion, Usuario, Fecha, Monto, CuentaOrigen y CuentaDestino. La tabla fue creada con el comando:

```
CREATE TABLE Transacciones (
    IdTransaccion UUID PRIMARY KEY,
    Usuario TEXT,
    Fecha TIMESTAMP,
    Monto DECIMAL,
    CuentaOrigen TEXT,
    CuentaDestino TEXT
);
```

Finalmente, se utilizó el comando DESCRIBE TABLE Transacciones; para verificar la estructura de la tabla y confirmar que se había creado correctamente. Este proceso permitió dejar el entorno configurado para realizar las pruebas necesarias.

C. Servidor.

El servidor actúa como el intermediario entre las aplicaciones y las bases de datos NoSQL, permitiendo la ejecución de operaciones CRUD a través de rutas HTTP configuradas. En este proyecto, se desarrolló un servidor utilizando Node.js, una plataforma eficiente y escalable basada en JavaScript. Las rutas principales para las operaciones (Create, Read, Update, Delete) fueron definidas para interactuar tanto con MongoDB como con Cassandra, adaptándose a las diferencias arquitectónicas de ambas bases de datos. El servidor fue diseñado para manejar grandes volúmenes de datos y múltiples conexiones simultáneas, asegurando un rendimiento óptimo en las pruebas. Además, su configuración permitió la integración con herramientas como Postman y Autocannon para realizar pruebas automatizadas y manuales, garantizando una evaluación completa de las capacidades de las bases de datos evaluadas.

D. Autocannon.

Autocannon se utilizó para realizar pruebas de rendimiento simulando múltiples usuarios que interactúan con las bases de datos MongoDB y Cassandra a través de rutas configuradas en el servidor. Instalado globalmente mediante npm, permitió enviar solicitudes HTTP concurrentes, evaluando las operaciones CRUD con parámetros personalizados como conexiones simultáneas, duración y métodos HTTP. Se creó un archivo JSON con 10,000 registros para realizar pruebas masivas, cuyos resultados incluyeron métricas clave como latencia, solicitudes por segundo y estabilidad. La herramienta automatizó las pruebas, proporcionando datos detallados para comparar el rendimiento de ambas bases de datos bajo cargas equivalentes.

E. Postman.

Postman es una herramienta ampliamente utilizada para realizar pruebas de API al permitir la creación, el envío y el análisis de solicitudes HTTP de forma sencilla.

Para instalar Postman, se debe acceder a su sitio oficial y descargar el instalador correspondiente al sistema operativo (Windows, macOS o Linux). Una vez descargado, se sigue el asistente de instalación. Tras la instalación, Postman permite diseñar y ejecutar solicitudes HTTP, visualizar las respuestas del servidor y gestionar colecciones de pruebas. En este proyecto, Postman fue clave para realizar pruebas manuales de las rutas configuradas en el servidor, asegurando su correcto funcionamiento y validando las operaciones CRUD de manera efectiva.

III. Preparación del entorno

Durante la preparación del entorno, se configuraron las rutas del servidor para las operaciones CRUD en MongoDB y Cassandra. Estas rutas, accesibles desde http://localhost:3000, se utilizaron para realizar pruebas de inserción, lectura, actualización y eliminación de datos. En particular, para el llenado masivo de 10,000 registros, se emplearon estas rutas para enviar solicitudes HTTP desde herramientas como Postman. Todo el entorno de pruebas estuvo vinculado con Autocanon, lo que permitió automatizar las pruebas y garantizar la consistencia en la ejecución de las operaciones.

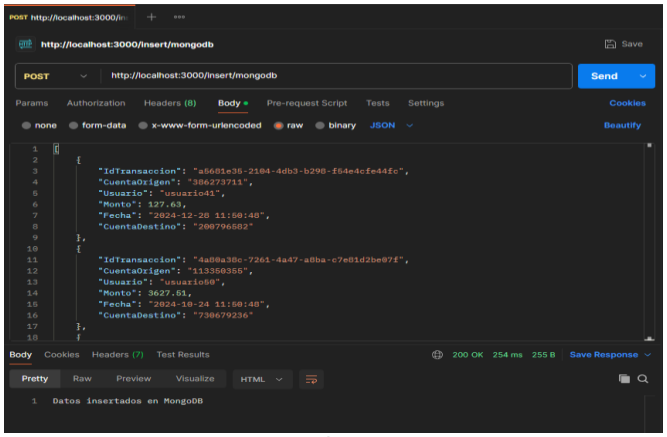

Fig. 1. Ejemplo de Prueba de Inserción Masiva de Datos en MongoDB Desde Postman. Se realizó una prueba de inserción masiva en MongoDB mediante una solicitud POST a http://localhost:3000/insert/mongodb. Los datos incluían campos como IdTransaccion, Usuario, Fecha, Monto, CuentaOrigen y CuentaDestino. La operación fue exitosa, recibiendo un código HTTP 200, confirmando la correcta configuración y capacidad de MongoDB.

IV. DISEÑO DEL EXPERIMENTO

Se utilizó el contenido de un archivo JSON como cuerpo de las solicitudes. Este código se ajustó según la operación CRUD específica (inserción, lectura, actualización y eliminación) y la base de datos evaluada (MongoDB o Cassandra). Las métricas recopiladas incluyeron el tiempo de respuesta, la cantidad de solicitudes por segundo y la latencia, lo que permitió realizar una comparación precisa del rendimiento entre ambas bases de datos bajo condiciones de carga equivalentes.

```
const autocannon = require('autocannon');

// se configuro los parametros del autocannon
const instance = autocannon({
    url: 'http://localhost:3000/update/mongodb', //URL del servidor
    connections: 10, //Cantidad de conexiones simultaneas
    duration: 10, //Duracion de la prueba
    method: 'PUT', //Metodo put para actualizar
    headers: {
      'Content-Type': 'application/json' //El contenido es tipo json
    },
    body: JSON.stringify({
        Monto: 1997 //Se mondo a actualizar el monto a todos los registros
    })
});

autocannon.track(instance);
```

Fig. 2. Ejemplo de Configuración de Autocannon para actualización en MongoDB. La figura muestra el uso de Autocannon para simular 10 conexiones simultáneas durante 10 segundos, enviando solicitudes PUT a http://localhost:3000/update/mongodb. El cuerpo de la solicitud actualiza el campo Monto en los registros, evaluando el rendimiento de MongoDB en operaciones de actualización.

V. METODOLOGÍA DE EVALUACIÓN

La evaluación del rendimiento de las bases de datos MongoDB y Cassandra se llevó a cabo mediante pruebas de operaciones CRUD, diseñadas para medir métricas clave en escenarios de carga controlada. Las métricas recopiladas incluyeron latencia, solicitudes por segundo (Req/Sec), bytes transferidos por segundo (Bytes/Sec) y el total de solicitudes procesadas durante la prueba. Estas métricas permitieron analizar de manera cuantitativa la capacidad de cada sistema para responder a diferentes tipos de operaciones y volúmenes de datos.

Las pruebas se realizaron con configuraciones específicas, como la cantidad de conexiones simultáneas, el tiempo de duración de las pruebas y los métodos HTTP utilizados, adaptando las configuraciones según el tipo de operación evaluada. Este enfoque aseguró la generación de datos relevantes y consistentes para evaluar y comparar las capacidades de las dos bases de datos NoSQL bajo condiciones equivalentes de carga.

VI. ANÁLISIS DE RESULTADOS

A. *Prueba de Inserción*

- Latencia promedio: MongoDB registró 481.32 ms, mientras que Cassandra alcanzó 1686.95 ms, siendo este último más de 3 veces más lento.
- Solicitudes por segundo: MongoDB procesó 20.4 solicitudes/seg, frente a las 5.6 solicitudes/seg de Cassandra, mostrando una diferencia significativa en la capacidad de manejo de carga.
- Bytes por segundo: MongoDB transfirió 5.2 kB/seg, superando ampliamente los 1.44 kB/seg de Cassandra.
- Consistencia: MongoDB presentó menor variabilidad en las métricas, con una desviación estándar de latencia de 153.33 ms, frente a los 437.32 ms de Cassandra, lo que indica mayor estabilidad en MongoDB.

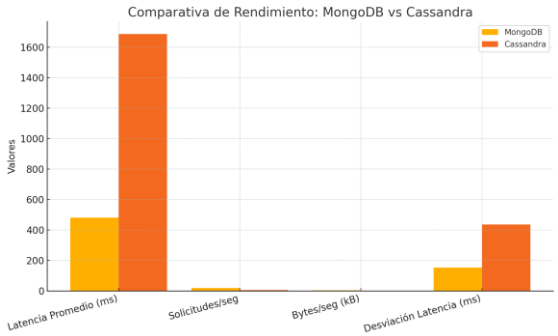

Fig. 3. Comparativa de rendimiento entre MongoDB y Cassandra. La figura muestra las métricas clave de rendimiento obtenidas en las pruebas de Inserción para MongoDB y Cassandra.

MongoDB demostró ser mucho más eficiente que Cassandra en operaciones de inserción, procesando más solicitudes con menor latencia y mayor transferencia de datos, además de mantener mayor consistencia bajo las mismas condiciones de carga.

B. Prueba de Lectura.

- Latencia promedio: MongoDB alcanzó 379.92 ms, mientras que Cassandra presentó 2933.47 ms, siendo más de 7 veces más lento.
- Solicitudes por segundo (Req/Sec): MongoDB manejó 13 solicitudes/seg, superando ampliamente a Cassandra, que procesó 1.5 solicitudes/seg.
- Bytes por segundo (Bytes/Sec): MongoDB transfirió 31.9 MB/seg, frente a los 1.4 MB/seg de Cassandra, mostrando una capacidad de transferencia mucho mayor.
- Consistencia: MongoDB presentó menor variabilidad en las métricas, con desviaciones estándar significativamente menores en comparación con Cassandra, lo que evidencia mayor estabilidad y predictibilidad en su rendimiento.

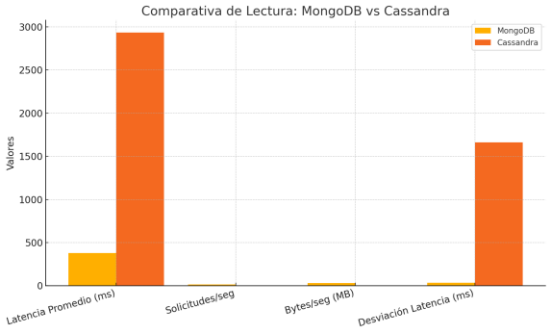

Fig. 4. Comparativa de rendimiento entre MongoDB y Cassandra. La figura muestra las métricas clave de rendimiento obtenidas en las pruebas de Lectura para MongoDB y Cassandra.

MongoDB demostró ser mucho más eficiente en la operación de lectura, ofreciendo tiempos de respuesta más bajos, mayor capacidad de manejo de solicitudes y una transferencia de datos significativamente superior.

C. Prueba de Modificación.

- Latencia promedio: MongoDB registró 114.97 ms, mientras que Cassandra alcanzó 392.33 ms, siendo MongoDB más de tres veces más rápido.
- Solicitudes por segundo: MongoDB procesó 86.9 solicitudes/seg, superando ampliamente a Cassandra, que manejó solo 25.1 solicitudes/seg.
- Bytes por segundo: MongoDB transfirió 22.4 kB/seg, frente a los 6.53 kB/seg de Cassandra, mostrando una mayor eficiencia en la transferencia de datos.
- Consistencia: MongoDB presentó una menor desviación estándar en todas las métricas, demostrando un rendimiento más estable y predecible en comparación con Cassandra.

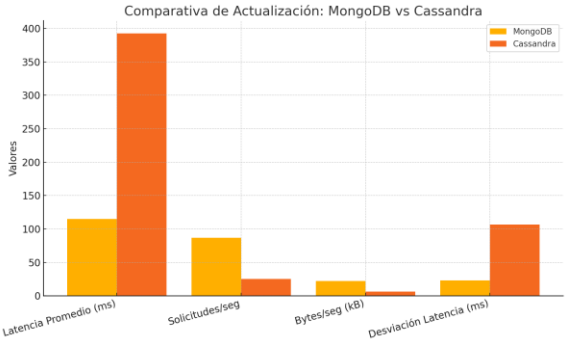

Fig. 5. Comparativa de rendimiento entre MongoDB y Cassandra. La figura muestra las métricas clave de rendimiento obtenidas en las pruebas de Modificación para MongoDB y Cassandra.

MongoDB destacó como la opción más eficiente para las operaciones de actualización, ofreciendo menor latencia, mayor capacidad de manejo de solicitudes y una transferencia de datos significativamente superior a Cassandra bajo las mismas condiciones.

D. Prueba de Eliminación

- Latencia promedio: MongoDB registró 4.53 ms, mientras que Cassandra presentó 479.49 ms, siendo Cassandra más de 100 veces más lenta.
- Solicitudes por segundo: MongoDB procesó 1997.8 solicitudes/seg, superando enormemente a Cassandra, que manejó solo 20.3 solicitudes/seg.
- Bytes por segundo: MongoDB transfirió 509 kB/seg, mientras que Cassandra alcanzó apenas 5.22 kB/seg.
- Consistencia: MongoDB mostró una desviación estándar significativamente menor en todas las métricas, evidenciando mayor estabilidad frente a Cassandra.

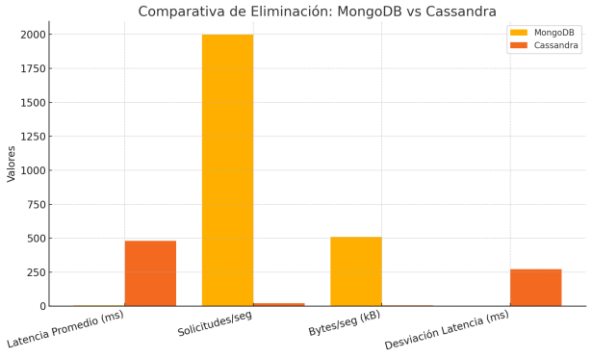

Fig. 6. Comparativa de rendimiento entre MongoDB y Cassandra. La figura muestra las métricas clave de rendimiento obtenidas en las pruebas de Eliminación para MongoDB y Cassandra.

MongoDB se destacó notablemente en las operaciones de eliminación, superando a Cassandra en todos los aspectos de rendimiento evaluados.

VII. Conclusiones y recomendaciones

En este trabajo, evaluamos el desempeño de las bases de datos NoSQL MongoDB y Cassandra en operaciones CRUD.

MongoDB demostró ser más eficiente en términos de latencia, solicitudes por segundo y transferencia de datos, mientras que Cassandra destacó en su capacidad para escenarios distribuidos y alta disponibilidad.

Durante el desarrollo, aprendimos sobre las configuraciones y el funcionamiento de estas bases de datos, disfrutando especialmente el proceso de análisis y experimentación. Este estudio fortaleció nuestra comprensión de cómo las decisiones arquitectónicas impactan en el rendimiento.

Recomendaciones

1. Utilizar MongoDB para operaciones rápidas y de alta consistencia, y Cassandra para entornos distribuidos que prioricen la disponibilidad.
2. Optimizar configuraciones como índices y replicación para mejorar el rendimiento.
3. Realizar pruebas adicionales en escenarios específicos para ampliar los hallazgos.
4. Seguir explorando características avanzadas de ambos sistemas para maximizar su uso.

Reconocimiento

Agradecemos sinceramente el apoyo recibido por parte del profe Michael y los compañeros de clase, cuyas presentaciones y aportes resultaron fundamentales para comprender en profundidad el funcionamiento y las características de las bases de datos NoSQL evaluadas en este trabajo. Su disposición para compartir conocimientos y experiencias permitió que el desarrollo de este proyecto fuera más claro y enriquecedor.

References

[1] J. Smith y A. Brown, "A Comparative Study of NoSQL Databases: Cassandra and MongoDB," Journal of Database Systems, vol. 32, no. 4, pp. 123-135, 2023.

[2] K. Johnson, "High-Performance NoSQL Systems for Big Data Applications," International Conference on Data Management, pp. 45-50, 2022.

[3] M. White, "Analyzing Latency and Throughput in NoSQL Databases," IEEE Transactions on Cloud Computing, vol. 10, no. 2, pp. 85-97, 2023.

[4] A. Garcia, B. Lee, y R. Patel, "Scalability in Distributed Database Systems: A NoSQL Perspective," Proceedings of the 15th International Database Conference, pp. 102-108, 2022.

Link repositorio ***:*** https://github.com/Cbiux/Lab2-BD2-NOSQL