REVISTA INDIA DE CIENCIA Y TECNOLOGÍA Un estudio de rendimiento y Comparación de bases de datos NoSQL: MongoDB, Cassandra y Redis usando YCSB

Artículo en Indian Journal of Science and Technology · Agosto de 2022

DOI: 10.17485/IJST/v15i31.1352

3 autores, entre ellos:

Mohammad Abu Kausar

Universidad de Nizwa

21 PUBLICACIONES 258 CITAS

VER EL PERFIL

Mohamed Nasr

Colegio Mazoon

25 PUBLICACIONES 181 CITAS

VER EL PERFIL

REVISTA INDIA DE CIENCIA Y TECNOLOGÍA

ARTÍCULO DE INVESTIGACIÓN

Un estudio de rendimiento y Comparación de bases de datos NoSQL: Uso de MongoDB, Cassandra y Redis YCSB

Mohammad Abu Kausar1 , Mohammad Nasar2 , Arockiasamy Soosaimanickam1

1 Departamento de Sistemas de Información, Universidad de Nizwa, Sultanato de Omán
2 Departamento de Computación e Informática, Mazoon College, Sultanato de Omán

ACCESO ABIERTO

Recibido: 28-06-2022

Aceptado: 23-07-2022

Publicado: 13.08.2022

Cita: Abu Kausar M, Nasar M, Soosaimanickam A (2022) Un estudio de rendimiento y comparación de bases de datos NoSQL: MongoDB, Cassandra y Redis utilizando YCSB. Revista India de Ciencia y Tecnología 15(31): 1532-1540. https://doi.org/10.17485/IJST/v15i31.1352 _

Autor correspondiente.

kausar@unizwa.edu.om

Financiamiento: Ninguno

Intereses en competencia: Ninguno

Publicado por la Sociedad India para Educación y Medio Ambiente (iSee)

ISSN
Imprimir: 0974-6846
Electrónica: 0974-5645

Abstracto

Antecedentes/Objetivos: Las bases de datos relacionales son una tecnología comúnmente utilizada que permite el almacenamiento, administración y recuperación de varios esquemas de datos. Sin embargo, para ciertas bases de datos grandes, la ejecución de consultas puede convertirse en un procedimiento ineficiente y que requiere mucho tiempo. Además, almacenar enormes volúmenes de datos requiere servidores con mayor capacidad y escalabilidad. Las bases de datos relacionales tienen límites cuando se trata de escalabilidad para grandes cantidades de datos. Por otro lado, los sistemas de bases de datos no relacionales, a menudo conocidos como NoSQL, se crearon para satisfacer mejor las demandas de almacenamiento de valores clave de enormes volúmenes de registros. Sin embargo, existen varias opciones NoSQL y la mayoría aún no se han comparado exhaustivamente. El objetivo de esta investigación es examinar diferentes bases de datos NoSQL y evaluar su rendimiento en términos de almacenamiento y recuperación de datos típicos. Métodos: En este estudio, utilizamos la herramienta YCSB para medir el rendimiento de tres bases de datos NoSQL: MongoDB, Cassandra y Redis. Probamos seis cargas de trabajo diferentes con 100.000, 250.000, 500.000, 750.000 y 1.000.000 operaciones. Nuestra prueba se diseñó con cinco operaciones diferentes, es decir, 100000, 250000, 500000, 750000 y 1000000, con seis cargas de trabajo diferentes para ver qué base de datos es más adecuada para aplicaciones que utilizan una gran cantidad de datos para procesar. Hallazgos: MongoDB es una base de datos NoSQL de rendimiento superior entre Cassandra y Redis. Las numerosas optimizaciones utilizadas por los diseñadores de soluciones NoSQL para mejorar el rendimiento, como un buen funcionamiento de la memoria caché, tienen un impacto directo en el tiempo de ejecución. En todas las cargas de trabajo, excepto en la carga de trabajo D, MongoDB ha reducido significativamente la latencia en todos los recuentos de operaciones. Novedad: también medimos la latencia promedio de diferentes escenarios de carga de trabajo que incluyen una combinación de actividades de lectura, escritura y actualización.

Palabras clave: NoSQL; YCSB; grandes datos; computación en la nube; MongoDB; Casandra; Redis

1. Introducción

Las siguientes son las razones principales por las que el "mecanismo de almacenamiento de datos" se considera el núcleo de los sistemas de software corporativo: (1) la parte más importante del software es la que controla la rapidez con la que una aplicación responde a una solicitud, y (2) la pérdida de datos es Con frecuencia se considera indeseable porque interrumpe operaciones comerciales críticas. Los sistemas de gestión de bases de datos relacionales (RDMS) eran la única opción hasta la llegada de las bases de datos NoSQL (Not-Only SQL). Sin embargo, a medida que crece la cantidad de datos conservados, las limitaciones del sistema de gestión de bases de datos relacionales, como la escalabilidad y el almacenamiento, así como la pérdida de eficiencia de las consultas debido a grandes volúmenes de datos, se vuelven más complejas, lo que hace que el almacenamiento y el mantenimiento de bases de datos más grandes sean más desafiantes. 1) .

Las bases de datos NoSQL, que significa "No sólo SQL", se conocen desde 2009 para satisfacer las nuevas necesidades de rendimiento al procesar grandes volúmenes de datos. NoSQL no reemplaza las bases de datos relacionales; más bien, complementa o reemplaza la funcionalidad de las bases de datos relacionales para brindar soluciones más interesantes en situaciones específicas. El término "NoSQL" se compone de dos palabras: "no" y "SQL"(2) . El nombre puede parecer opuesto a las bases de datos SQL, lo que lleva a algunos a creer que indica el fin del lenguaje SQL y, como resultado, debe evitarse. En verdad, "No" es un acrónimo de "No sólo", lo que enfatiza que las bases de datos relacionales y SQL son mucho más que eso. El objetivo principal del movimiento no es reemplazar las bases de datos relacionales como SQL sino proporcionar alternativas o ampliar las capacidades de los modelos actuales para gestionar grandes cantidades de datos dispersos. El movimiento NoSQL reúne numerosas soluciones de gestión de datos que ya no se basan en la arquitectura clásica de las bases de datos relacionales y se distinguen del modelo SQL por una lógica de representación de datos no relacional(3) . La unidad lógica ya no es la tabla y los datos generalmente no se manipulan con SQL. Originalmente, se utilizaba para modificar bases de datos masivas para sitios web con grandes audiencias, como Amazon.com, Google, Facebook o eBay. Estas nuevas estructuras de almacenamiento de datos, a diferencia de los DBMS y almacenes de datos tradicionales, están dispersas en muchos servidores y creadas para acomodar a los cientos o millones de usuarios que realizan cambios y lecturas(4) . Dependen de sistemas de archivos paralelos para aumentar la eficiencia y hacer frente a las limitaciones de recursos multiplicando el hardware para permitir la paralelización del almacenamiento y el acceso a la información. Las bases de datos NoSQL tienden a utilizar servidores de gama baja a costos más bajos para equipar los "clústeres". Los servidores para bases de datos NoSQL son generalmente económicos y de calidad media, a diferencia de los que utilizan las bases de datos relacionales. Además, la gran mayoría de soluciones NoSQL son de código abierto, lo que refleja, por un lado, un importante ahorro en el precio de las licencias. El volumen de datos y su diversidad son tan importantes hoy en día que han surgido los sistemas de almacenamiento y gestión de datos. Esto abarca desde sistemas de gestión de archivos hasta sistemas altamente estructurados(5) (como los sistemas relacionales), pasando por una variedad de sistemas de almacenamiento de datos semiestructurados.

Esta diversidad también ha llevado al amplio uso de formatos de intercambio de datos. En cuanto a los datos, algunos están semánticamente desestructurados, como el sonido. Los datos semiestructurados tienen una cierta forma de estructuración que generalmente no es explícita y que no impone una tipificación estricta.

El teorema CAP es un acrónimo de "coherencia", "disponibilidad" y "tolerancia de partición", también conocido como teorema de Brewer.
Este teorema, formulado por Eric Brewer en 2000 y demostrado por Seth Gilbert y Nancy Lych en 2002, es una conjetura que establece que es imposible, en un sistema informático con computación acanalada, garantizar las siguientes tres restricciones al mismo tiempo(6) : Consistencia: en el mismo momento, todos los nodos (servidores)

del sistema ven los mismos datos.

Disponibilidad: asegúrese de que cualquier solicitud obtenga una respuesta, incluso si no se ha modificado.

Tolerancia para particiones: excepto en el caso de una interrupción generalizada de la red, el sistema debe poder responder correctamente a todas solicitudes en todas las condiciones. Al dividir una red en subredes, cada subred debe poder funcionar de forma independiente.

MongoDB es una base de datos orientada a documentos, de código abierto y proporcionada bajo la licencia AGPL. (Licencia gratuita), asegurando excelente rendimiento, disponibilidad y escalabilidad bajo demanda. La base de datos MongoDB ha sido creada en C++ por la empresa 10gen desde 2007, cuando trabajaba en un sistema de computación en nube de datos ampliamente distribuido similar al App Engine de Google. La versión inicial se lanzó en 2009, pero la versión 1.4 no fue declarada comercialmente aceptable hasta 2010(7) .

Cassandra es un sistema de gestión de datos a gran escala diseñado originalmente en 2007 por ingenieros de Facebook para abordar cuestiones relacionadas con el almacenamiento y uso de grandes volúmenes de datos. En 2008, intentaron democratizarlo proporcionando una versión estable y documentada, disponible en GoogleCode. Sin embargo, Cassandra no recibió una acogida especialmente entusiasta(8) . Por eso los ingenieros de Facebook decidieron en 2009 llevar a Cassandra a la Incubadora Apache. En 2010, Cassandra fue ascendida al Proyecto Apache de nivel superior.

Redis, que significa Remote Dictionary Server, es una base de datos NoSQL de tipo clave/valor con licencia BSD que se creó en C (9).
. Pertenece al movimiento NoSQL y se esfuerza por ofrecer el mayor rendimiento. Redis admite una variedad de tipos de datos básicos, incluidas listas, matrices asociativas, conjuntos y conjuntos ordenados. Con el lanzamiento de "sentinelfailover", que maneja el monitoreo, las alertas y el cambio automático de instancias en caso de problemas, Redis continúa silenciosamente su curso después de ver un crecimiento significativo en 2010.

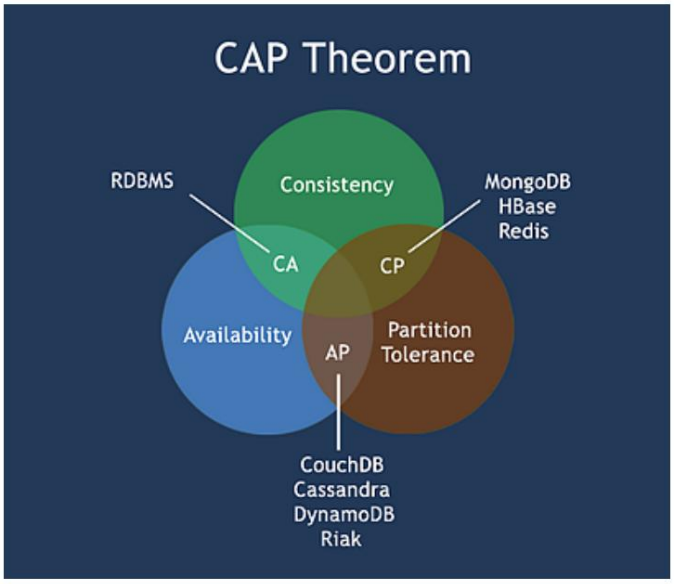

Figura 1. El teorema CAP (5)

Trabajo relacionado

El autor(10) contrasta las bases de datos SQL y NoSQL, detallando su historia y aplicaciones. El autor (11) ofrece una descripción detallada de las bases de datos NoSQL, así como una comparación basada en las siguientes características: (1) escalabilidad, (2) integridad y coherencia transaccional, (3) modelado de datos, (4) funcionalidad de consulta y (5) disponibilidad de acceso y interfaz(11) . Hecht y Jablonski dan una encuesta sobre bases de datos NoSQL basada en casos de uso(12) . Evalúan las bases de datos NoSQL en función de sus modelos de datos, capacidades de consulta, controles de concurrencia, particiones y capacidades de replicación.

Yahoo! Cloud Serving Benchmark es utilizado por(1) para evaluar y comparar el rendimiento de las bases de datos NoSQL. Al alterar los índices de operaciones de lectura, actualización e inserción, produjeron 600.000 registros al azar y los utilizaron con diferentes cargas de trabajo.
Las bases de datos utilizadas en el experimento fueron Redis, Cassandra, HBase, MongoDB y OrientDB8. Afirman que Redis, una base de datos en memoria, funciona al más alto nivel en general. Además, afirman que debido a que Cassandra y HBase están optimizados para operaciones de actualización, funcionan bien.

En esta investigación(13) se examinó el comportamiento de dos de las bases de datos NoSQL basadas en documentos más populares, MongoDB y MySQL basada en documentos, en términos de la complejidad y efectividad de las operaciones CRUD, particularmente en operaciones de consulta.
Se utilizaron las bases de datos MongoDB y MySQL para crear una aplicación de estudio de caso que tiene como objetivo modelar y simplificar las operaciones de los proveedores de servicios que requieren una gran cantidad de datos para poder realizar esta investigación. Además, estas pruebas se ejecutan en conjuntos de datos con entre 1000 y 100 000 registros. Los datos se produjeron al azar mediante un proceso iterativo y solo permitieron campos particulares. El autor destacó la herramienta YCSB como un posible trabajo futuro.

En el artículo (14), El autor comparó CouchDB y MongoDB, dos bases de datos NoSQL basadas en documentos. El autor realizó importantes comparaciones paramétricas entre estos dos conjuntos de datos. La técnica de replicación y el soporte de la plataforma son dos distinciones clave. Las comparaciones del autor hacen evidente que MongoDB es una opción superior a CouchDB si una aplicación necesita mayor eficiencia y velocidad. Si la base de datos se expande rápidamente, CouchDB es menos adecuado que MongoDB. El autor no aplicó ninguna compresión basada en datos.

En esta investigación, se comparan y contrastan dos sistemas de gestión de bases de datos NoSQL ampliamente utilizados, MongoDB y Apache Cassandra, en la investigación de evaluación comparativa de rendimiento del autor(15). La métrica de rendimiento que se ha analizado es el tiempo de ejecución total. Los resultados medidos muestran que Apache Cassandra supera a MongoDB cuando el número de operaciones y el nivel de paralelismo son altos.

Cornelia A. Gyorodi, Diana V. Dumse-Burescu, Doina R. Zmaranda y Robert S. Gyorodi(16) realizaron una comparación de dos bases de datos NoSQL relativamente actuales, MongoDB y MySQL basada en documentos, teniendo en cuenta su impacto en el rendimiento de la aplicación. al realizar solicitudes CRUD. Finalmente, independientemente de la complejidad de las consultas o la cantidad de datos, ambas bases de datos son adecuadas tanto para aplicaciones Big Data que requieren un gran volumen de datos como para bases de datos muy complicadas, con tiempos de respuesta muy rápidos.

En la revisión de la conferencia anterior, los autores prueban el rendimiento basándose en registros específicos como 1000 y 10000. Nuestro intento de abordar esto en este artículo es hacer un análisis comparativo del rendimiento de tres bases de datos de uso común: MongoDB, Cassandra y Redis. Las pruebas existentes se basan en el rendimiento general de 100.000, 250.000, 500.000, 750.000 y 1.000.000 operaciones. También mediremos la latencia promedio de diferentes escenarios de carga de trabajo que incluyen una combinación de actividades de lectura, escritura y actualización. Los autores emplean la ampliamente utilizada Yahoo Cloud Serving Benchmarking Tool (YCSB), que es una herramienta de medición del rendimiento para bases de datos NoSQL. Esto ayuda al usuario a comprender mejor el rendimiento de la base de datos y elegir qué sistema es mejor para una carga de trabajo determinada.

El resto de este estudio de investigación está organizado de la siguiente manera. La siguiente sección cubre el trabajo relacionado. La metodología se describe en la Sección 2. Los resultados y la discusión de las pruebas se presentan en la Sección 3, seguidos de la evaluación experimental. Finalmente, ofrecemos nuestras conclusiones y recomendaciones en la sección 4.

2. Metodología

Una herramienta compleja proporcionada por Yahoo se llama YCSB (Yahoo! Cloud Serving Benchmark). Es un nuevo enfoque de evaluación comparativa de código abierto que permite a los usuarios crear sus propios paquetes agregando parámetros de carga de trabajo adicionales o, si es necesario, escribiendo código Java. En Yahoo! Una investigación que incluyó datos de referencia para cuatro sistemas de uso común, se descubrió que: Apache HBase, Apache Cassandra, YahooPNUTS,! y una versión fragmentada de MySQL son los mejores en términos de rendimiento y elasticidad. En (17), el autor aboga por puntos de referencia de escalabilidad y sugiere que un buen lugar para comenzar es el YCSB. Él piensa que YCSB es el principal punto de referencia NoSQL en este momento. El punto de referencia analiza la escalabilidad, que incluye cómo el sistema evaluado escala cuando se agregan servidores adicionales y qué tan rápido se adapta a nuevos servidores, así como el rendimiento básico, como las características de latencia a medida que aumenta la carga del servidor(17) . La mayoría de los sistemas NoSQL son compatibles con esta herramienta, que es multiplataforma y fácil de personalizar para estas soluciones. Se ha utilizado en varios estudios de investigación(17–21) y se utiliza ampliamente para comparar el rendimiento de los sistemas de gestión de bases de datos NoSQL en la nube. Un generador de datos y un grupo de pruebas de rendimiento para agregar, actualizar y eliminar elementos conforman YCSB. Utilizando varias cargas de trabajo, podemos especificar la cantidad de registros que se cargarán, la cantidad de operaciones que se realizarán y la división entre lectura y escritura en cada prueba (Ejemplo: Carga de trabajo A: 50 por ciento de lecturas, 50 por ciento de actualizaciones). Estas cargas de trabajo se pueden cambiar y adaptar según el tipo de resultados de prueba previstos. En la siguiente imagen se muestra cómo interactúa YCSB con una base de datos:

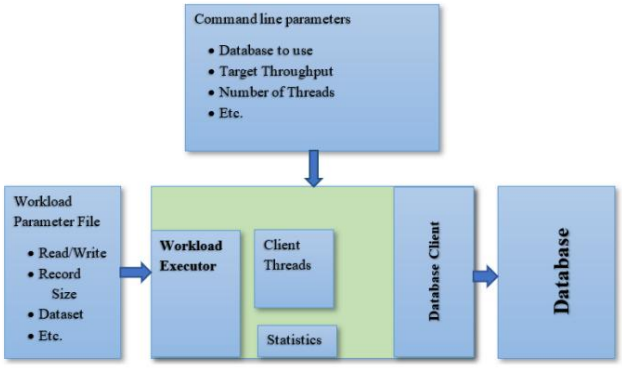

Figura 2. Arquitectura YCSB

Detalles de configuración de la prueba La

Tabla 1 contiene una lista de todos los componentes de hardware y software utilizados en nuestra investigación.

2.2 Plan de pruebas de rendimiento

Usaremos las tres bases de datos cubiertas en este documento para nuestras pruebas: MongoDB, Cassandra y Redis. Usando Yahoo! En el marco de Cloud Serving Benchmark, pondremos a prueba cada una de las tres bases de datos. La herramienta YCSB se compone de dos partes: un cliente ycsb que crea la carga de trabajo y cargas de trabajo definidas, que son los escenarios que ejecutará el cliente.

Continuamos con la instalación y configuración de YCSB 0.17.0 después de descargar e instalar MongoDB 4.4, Cassandra 4.0.3 y Redis 6.2.6, pero primero necesitábamos instalar Java, Maven y Git en nuestra máquina. Cada prueba comenzaba con un

Tabla 1. Detalles de la configuración de la prueba

Hardware	Software
Procesador: Intel(R) Core(TM) i7-7500U CPU a 2,70 GHz 2,90 GHz RAM: 12,00 GB Disco duro: 1 TB	ventanas 10 , PC de 64 bits YCSB 0.17.0 MongoDB 4.4. Casandra 4.0.3 Redis 6.2.6

base de datos en blanco. Comenzamos creando seis cargas de trabajo en la herramienta YCSB. Las cargas de trabajo (detalladas a continuación) se ejecutan en las tres bases de datos después de que se hayan cargado los datos. Entre cada tarea se realizan comprobaciones del estado de la base de datos.

Los objetivos generales de las pruebas fueron: 1. Elegir cargas de trabajo que sean representativas de las aplicaciones actuales.

2. Utilice cantidades de datos similares a las que se encuentran en conjuntos de datos de "grandes datos".

3. Varíe las cantidades de carga de trabajo de lectura/escritura para comparar el rendimiento de las dos soluciones.

4. Para una mayor comparabilidad de los resultados y una comparación más clara, mantenga los mismos títulos de cargas de trabajo con las mismas tasas que en (1) .

Las cargas de trabajo son un conjunto de escenarios que incluyen una combinación de actividades de lectura, escritura y actualización. Las siguientes son las cargas de trabajo que Usamos en nuestras pruebas: • La carga de trabajo A tiene una proporción de 50% de lecturas y 50% de actualizaciones;

• La carga de trabajo B tiene una proporción de lecturas del 95 % y actualizaciones del 5 %. • Carga de trabajo C, todas las lecturas; • Carga de trabajo D, que consta de 95 % de lecturas y 5 % de inserciones;

• Carga de trabajo E, que consta de un 95% de escaneo y un 5% de inserción; • Carga de trabajo F, que se compone de 50% de lecturas y 50% de lectura-modificación-escritura; Se eligieron para las pruebas 100.000, 250.000, 500.000, 750.000 y 1.000.000 operaciones. También estamos seleccionando entradas para Uniforme utilizando la distribución predeterminada.

3. Resultados y discusión

Para todas las bases de datos y cargas de trabajo en todos los recuentos de operaciones, todas las pruebas se ejecutaron exitosamente, lo que confirma que no se produjo ningún error de inserción, lectura o actualización.

3.1 Carga de trabajo A (50% lectura y 50% actualización)

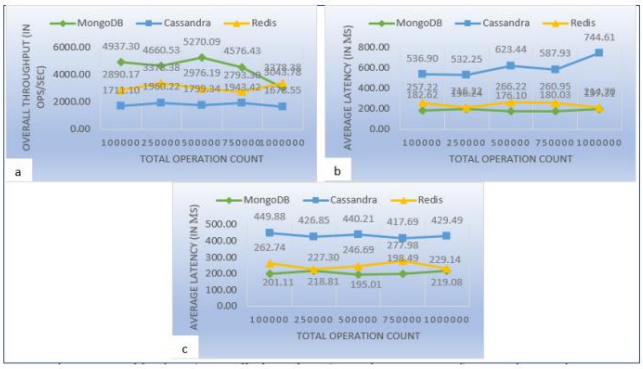

Fig 3. Carga de trabajo A: a) Rendimiento general (operaciones por segundo) frente a operaciones totales. b) Latencia promedio de lectura (en c) ms) vs Operaciones Totales. Latencia promedio de actualización (en ms) vs Operaciones Totales

El rendimiento total de las bases de datos MongoDB y Cassandra cae precipitadamente a medida que aumenta el número de operaciones, pero el rendimiento general de Redis aumenta (Figura 3 a). Cuando se evaluó con 100.000 recuentos de operaciones, MongoDB tuvo un rendimiento 2,9 veces mayor y Redis tuvo un rendimiento 1,68 veces mayor que Cassandra. Sin embargo, una vez que la operación cuenta

se aumentan a 1.000.000, esto se reduce a sólo 1,81 veces para MongoDB y aumenta a 2 veces para Redis. Como resultado, aunque MongoDB ofrece un rendimiento superior, en este caso hay una mayor reducción en el rendimiento general.

La latencia promedio de lectura de la base de datos Cassandra aumenta constantemente (Figura 3b). En comparación con MongoDB y Redis, Cassandra tiene una mayor latencia de lectura. En comparación con Cassandra, MongoDB tiene una latencia de lectura del 34 por ciento y Redis tiene una latencia de lectura del 47,9 por ciento. Con 100.000 operaciones totales, o alrededor de 50.000 operaciones de lectura, la latencia de MongoDB aumenta progresivamente hasta el 26,5 por ciento, mientras que la latencia de Redis aumenta hasta el 28,8 por ciento.

En comparación con el rendimiento general y los gráficos de latencia de lectura, la latencia de actualización tiene una curva algo distinta. De 100.000 a 1.000.000 de operaciones, MongoDB y Redis tienen un aumento constante de la latencia, pero Cassandra tiene una latencia bastante estable. Sin embargo, en la Figura 3c se desprende claramente que MongoDB todavía tiene muy poca latencia al realizar operaciones de actualización.

3.2 Carga de trabajo B (95% de lectura y 5% de actualización)

En comparación con Redis y Cassandra, MongoDB ofrece muchas más operaciones por segundo en la carga de trabajo B. Redis y Cassandra garantizan un rendimiento estable a medida que aumenta la cantidad de operaciones. Sin embargo, el rendimiento de MongoDB puede variar.
Cuando se realizan 500.000 y 750.000 trámites disminuye drásticamente. Pero en comparación con Redis y Cassandra, MongoDB continúa conservando un rendimiento sustancialmente mayor (Figura 4a).

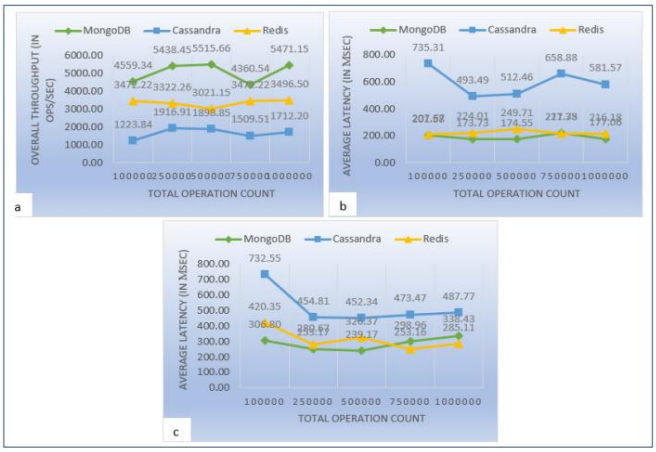

Fig 4. Carga de trabajo B: a) Rendimiento general (operaciones por segundo) frente a operaciones totales. b) Latencia promedio de lectura (en c) ms) vs Operaciones Totales. Latencia promedio de actualización (enms) vs Operaciones Totales

Como se podría predecir con una combinación de operaciones de lectura del 95 por ciento, MongoDB tiene una latencia de operaciones de lectura significativamente menor que Redis y Cassandra. Debido al gran volumen de operaciones activas, solo hay un pequeño aumento en la latencia de MongoDB en comparación con Redis y Cassandra (Figura 4b).

Una vez más, vemos que Cassandra todavía tiene una latencia de actualización mayor que Redis y MongoDB en la carga de trabajo B. Con un aumento
En los recuentos de operaciones, la latencia de Cassandra aumenta marginalmente, mientras que la latencia de MongoDB también aumenta (Figura 4c).

Carga de trabajo C (100% lectura)

En la carga de trabajo C, MongoDB tiene un total de operaciones por segundo sustancialmente mayor que Redis y Cassandra. El rendimiento de Cassandra cae a medida que aumenta el número de operaciones, pero el rendimiento de MongoDB comienza a disminuir marginalmente en 1.000.000 de operaciones. Con Redis, el rendimiento es casi constante durante todo el proceso. Pero en comparación con Redis y Cassandra, MongoDB todavía tiene un rendimiento considerablemente mayor (Figura 5a).

Con una operación de lectura del 100 %, la latencia en las operaciones de lectura es menor para MongoDB en comparación con Redis y Cassandra. Según la figura anterior, la latencia de MongoDB y Redis es casi consistente en todo el recuento de operaciones. En general, la latencia de MongoDB es menor que la de las otras dos bases de datos (Figura 5b).

3.4 Carga de trabajo D (5% de inserción y 95% de lectura)

En comparación con MongoDB y Cassandra, Redis tiene un total de operaciones por segundo sustancialmente mayor en la carga de trabajo D. Redis mejora el rendimiento mientras que Cassandra lo reduce a medida que aumenta el número de operaciones, aunque MongoDB experimenta una pequeña

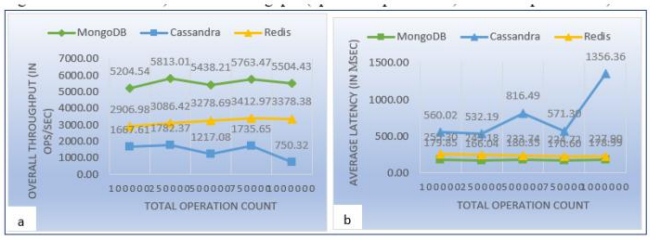

Fig 5. Carga de trabajo C: a) Rendimiento general (operaciones por segundo) frente a operaciones totales. b) Leer la latencia promedio (en μs) vs TotalOperaciones

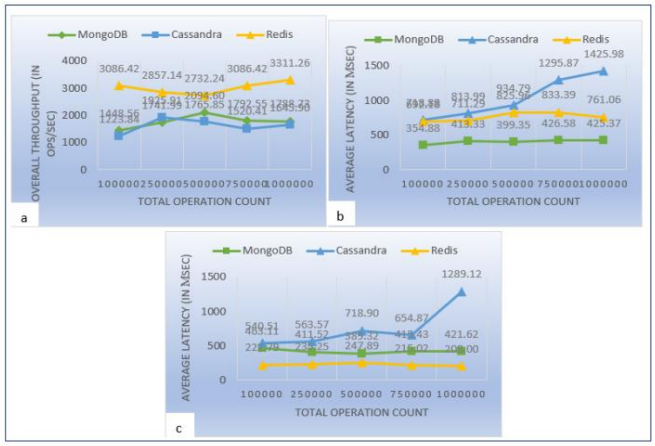

Fig 6. Carga de trabajo D: a) Rendimiento general (operaciones por segundo) frente a operaciones totales. b) Insertar latencia promedio (en c) Leer μs) vs Operaciones Totales. latencia promedio (en s) versus operaciones totales

Caída en el rendimiento cuando el número de operaciones alcanza el millón. La Figura 6a muestra que Redis mantiene una significativa mayor rendimiento que Cassandra y MongoDB.

La Figura 6b muestra que Cassandra todavía tiene una latencia de inserción mayor que Redis y MongoDB en la carga de trabajo D. En el caso de Cassandra, hay una modesta disminución en la latencia con un aumento en el recuento de operaciones, lo que es prácticamente consistente con MongoDB.

Con una operación de lectura del 95 %, la latencia en las operaciones de lectura es menor para Redis en comparación con MongoDB y Cassandra. De La latencia de MongoDB y Redis de la Figura 6c es casi consistente en todo el recuento de operaciones. En general, la latencia de Radis es menor que las otras dos bases de datos.

Para la carga de trabajo D, el rendimiento de Redis es mejor en caso de rendimiento general y operación de lectura.

3.5 Carga de trabajo E (95 % escaneado y 5 % inserción)

En comparación con Redis y Cassandra, MongoDB tiene un total de operaciones por segundo sustancialmente mayor para la carga de trabajo E. Si bien MongoDB aumenta modestamente el rendimiento a medida que el recuento de operaciones alcanza 7500 000, Cassandra pierde rendimiento a medida que la operación el conteo aumenta. Con Redis, el rendimiento es casi constante durante todo el proceso. La Figura 7a muestra que MongoDB consistentemente mantiene un rendimiento significativamente mayor que Redis y Cassandra.

La Figura 7b muestra que Redis tiene una latencia de inserción mayor que Cassandra y MongoDB para la carga de trabajo E. En el caso de Cassandra, hay una modesta disminución en la latencia con un aumento en el recuento de operaciones, lo que es prácticamente consistente con MongoDB.

La Figura 7c muestra que Cassandra todavía tiene una latencia de inserción mayor que Redis y MongoDB en la carga de trabajo E. En el caso de Redis, hay una modesta disminución en la latencia con un aumento en el recuento de operaciones, lo que es prácticamente consistente con MongoDB. Por lo tanto, MongoDB es la mejor opción para manejar la carga de trabajo E.

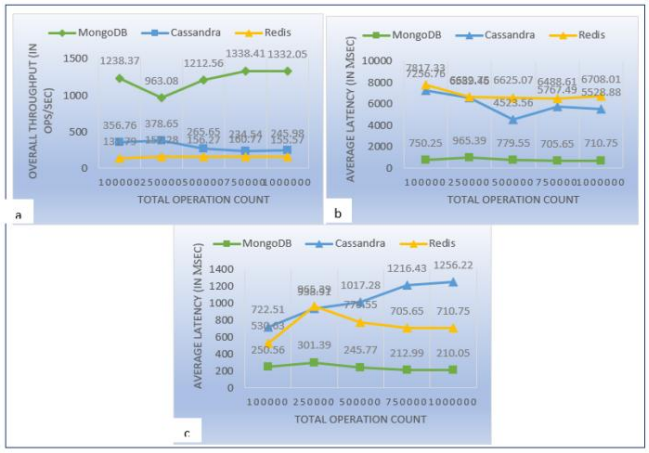

Fig 7. Carga de trabajo E: a) Rendimiento general (operaciones por segundo) frente a operaciones totales. b) Latencia promedio de escaneo (en c) Insertar μs) vs Operaciones Totales. latencia promedio (en μs) vs Operaciones Totales

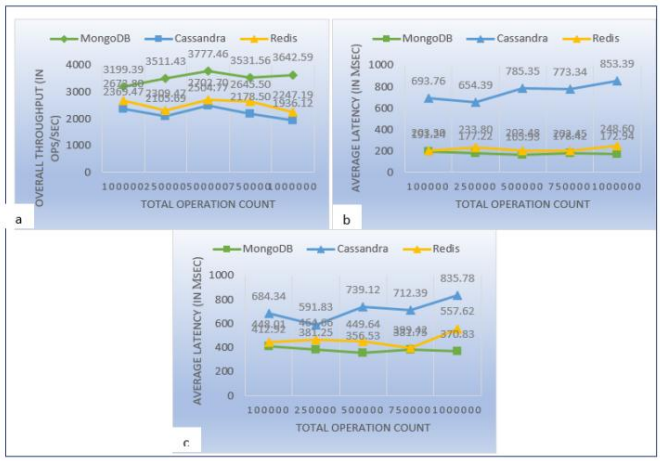

Fig 8. Carga de trabajo F: a) Rendimiento general (operaciones por segundo) frente a operaciones totales. b) Latencia promedio de lectura (en c) Latencia μs) vs Operaciones Totales. promedio de lectura-modificación-escritura (en s) versus operaciones totales

3.6 Carga de trabajo F (50% lectura y 50% lectura-modificación-escritura)

Nuevamente, MongoDB supera a Redis y Cassandra en la carga de trabajo F en términos de operaciones totales por segundo. Como con Casandra y Redis, su rendimiento cae a medida que aumenta el número de operaciones. Sin embargo, MongoDB muestra una pequeña ganancia en el rendimiento en Un millón de operaciones. La Figura 8a muestra que MongoDB mantiene consistentemente un rendimiento significativamente mayor que Redis y Casandra.

Con las operaciones de lectura de la carga de trabajo F, la latencia en las operaciones de lectura es menor para MongoDB en comparación con Redis y Cassandra. Según la Figura 8b, la latencia de MongoDB y Redis es casi consistente en todo el recuento de operaciones. En general, la latencia de MongoDB es más bajo que las otras dos bases de datos.

La Figura 8c muestra que Cassandra tiene una mayor latencia de lectura, modificación y escritura que Redis y MongoDB en la carga de trabajo F. La latencia cambia con un aumento en el recuento de operaciones, algo que disminuye para MongoDB y un ligero aumento para Redis. Por lo tanto, MongoDB es la mejor opción para manejar cargas de trabajo F de este tipo.

4 Conclusión y recomendaciones

Después de analizar los resultados de las tres bases de datos NoSQL, MongoDB 4.4 como almacén de documentos, Cassandra 4.0.3 como almacén de columnas, y Redis 6.2.6 como almacén clave-valor, y después de ejecutar seis cargas de trabajo compuestas por 100000, 250000, 500000, 750000 y 1000000

operaciones, llegamos a la conclusión de que las numerosas optimizaciones utilizadas por los diseñadores de soluciones NoSQL para mejorar el rendimiento, como un buen funcionamiento de la memoria caché, tienen un impacto directo en el tiempo de ejecución.

De las pruebas de rendimiento de MongoDB, Cassandra y Redis, hemos aprendido algunas cosas. • Redis tiene el mejor rendimiento de lectura de todas las bases de datos. Esto se debe al hecho de que los datos se almacenan y recuperan mediante memoria volátil. • En términos de operaciones de lectura, MongoDB superó a Cassandra. Como resultado, la asignación de registros para MongoDB se carga en la RAM, lo que mejora el rendimiento de lectura. • MongoDB superó a Redis y Cassandra en lo que respecta a operaciones de escaneo. • Cassandra superó a Redis en términos de operaciones de escaneo. • Era más difícil trabajar con Cassandra cuando se trataba de leer y actualizar. Esto se debe principalmente a la falta de optimización. para este tipo de procedimientos. • En todas las cargas de trabajo excepto en la carga de trabajo D, MongoDB ha reducido significativamente la latencia en todas las operaciones.

Como consecuencia de nuestras pruebas e investigaciones, podemos concluir que MongoDB es una base de datos NoSQL de rendimiento superior.

Sin embargo, el estudio descrito en el artículo tiene una serie de deficiencias que pueden solucionarse con nuevos enfoques de investigación. Una de estas opciones para ampliar el estudio propuesto pasará por evaluar varias bases de datos NoSQL en la nube. El examen de bases de datos NoSql adicionales para realizar pruebas, con el fin de poder probar otros elementos de rendimiento, también podría ser una segunda ruta para el desarrollo y mejora de este artículo.

Referencias

1) Martins P, Abbasi M, Sá F. Un estudio sobre el rendimiento NoSQL. En: Avances en Sistemas Inteligentes y Computación. Publicaciones internacionales Springer. 2019; pág. 603–611. Disponible en: https://doi.org/10.1007/978-3-030-16181-1_57.

2) Seghier NB, Kazar O. Evaluación comparativa de rendimiento y comparación de bases de datos NoSQL: Redis vs MongoDB vs Cassandra usando la herramienta YCSB. 2021 Conferencia Internacional sobre Avances Recientes en Matemáticas e Informática (ICRAMI). 2021; pág. 1–6. Disponible en: https://doi.org/10.1109/ICRAMI52622.2021.9585956.

3) Nasar M, Kausar MA. Idoneidad de la base de datos influxdb para aplicaciones iot. Revista internacional de tecnología innovadora y exploración de ingeniería. 2019;8(10):1850–1857.

4) Kausar MA, Nasar M. Una técnica eficaz para la detección y prevención de SQLIA mediante la utilización de coincidencia de cadenas basada en CHECKSUM. Internacional Revista de investigación científica y de ingeniería. 2018;9(1):1177–1182.

5) Kausar MA, Nasar M. Bases de datos SQL versus NoSQL para evaluar su idoneidad para la aplicación de big data. Avances recientes en informática y Comunicaciones. 20211;14:1098–108.

6) Kausar MA, Nasar M, Moyaid A. Técnicas de prevención y detección de inyecciones SQL en aplicaciones web ASP .NET. Revista internacional de actualidad Tecnología e Ingeniería (IJRTE). 2019;8(3):7759–66.

7) Bagga S, Sharma A. Un estudio comparativo de bases de datos NoSQL. En: Apuntes de conferencias sobre ingeniería eléctrica. Springer Singapur. 2021; pág. 51–61.

8) Gorbenko A, Romanovsky A, Tarasyuk O. Interacción entre la coherencia y el rendimiento de Cassandra NoSQL: un enfoque de evaluación comparativa. Comunicaciones en Informática y Ciencias de la Información. 2020;1279:168–184. Disponible en: https://doi.org/10.1007/978-3-030-58462-7_14.

9) Eddelbuettel D. Una breve introducción a Redis. . Disponible en: https://arxiv.org/pdf/2203.06559.pdf.

10) Meier A, Kaufmann M. Bases de datos Nosql. Bases de datos InSQL y NoSQL. Springer Vieweg. 2019. Disponible en: https://doi.org/10.1007/978-3-658-24549-8_7.

11) Meier A, Kaufmann M. Bases de datos SQL y NoSQL. Berlín/Heidelberg, Alemania; Medios especializados Wiesbaden. Medios especializados de Springer Wiesbaden. 2019. Disponible en: https://doi.org/10.1007/978-3-658-24549-8.

12) Diogo M, Cabral B, Bernardino J. Modelos de consistencia de bases de datos NoSQL. Internet del futuro. 2019;11(2):43–43. Disponible en: https://doi.org/10.3390/fi11020043.

13) Győrödi CA, Dumşe-Burescu DV, Zmaranda DR, Győrödi RŞ. Un estudio comparativo de MongoDB y MySQL basado en documentos para aplicaciones de Big Data Gestión de datos. Big Data y Computación Cognitiva. 2022;6(2):49–49. Disponible en: https://doi.org/10.3390/bdcc6020049.

14) Kaur R, Sahiwal JK. Una revisión de comparación entre bases de datos NoSQL: MongoDB y CouchDB. Revista Internacional de Tecnología Reciente y Ingeniería. 2019; pág. 892–898. Disponible en: https://www.ijrte.org/wp-content/uploads/papers/v7i6s/F03820376S19.pdf.

15) Andór CF. Análisis de métricas de tiempo de ejecución en la evaluación comparativa del rendimiento de bases de datos NoSQL. 2021 Conferencia Internacional sobre Software y Telecomunicaciones y Redes de Computadoras (SoftCOM). 2021; pág. 1–6. Disponible en: https://doi.org/10.23919/SoftCOM52868.2021.9559083.

16) Győrödi CA, Dumşe-Burescu DV, Zmaranda DR, Győrödi RŞ. Un estudio comparativo de MongoDB y MySQL basado en documentos para aplicaciones de Big Data Gestión de datos. Big Data y Computación Cognitiva. 2022;6(2):49–49. Disponible en: https://doi.org/10.3390/bdcc6020049.

17) Diogo M, Cabral B, Bernardino J. Modelos de consistencia de bases de datos NoSQL. Internet del futuro. 2019;11(2):43–43. Disponible en: https://doi.org/10.3390/bdcc6020049.

18) Martins P, Tomé P, Wanzeller C, Sá F, Abbasi M. Estudio comparativo de rendimiento NoSQL. EnConferencia Mundial sobre Sistemas y Tecnologías de la Información. Cham. Saltador. 2021. Disponible en: https://doi.org/10.1007/978-3-030-72651-5_41.

19) Pandey R. Evaluación comparativa de rendimiento y comparación de bases de datos basadas en la nube MongoDB (NoSQL) vs MySQL (Relacional) usando YCSB. Reporte técnico. 2020. Disponible en: https://doi.org/10.13140/RG.2.2.10789.32484.

20) Matallah H, Belalem G, Bouamrane K. Evaluación de bases de datos NoSQL: MongoDB, Cassandra, HBase, Redis, Couchbase, OrientDB. Revista Internacional de Ciencia del Software e Inteligencia Computacional (IJSSCI). 2020;12(4):71–91. Disponible en: https://doi.org/10.4018/IJSSCI.2020100105.

21) Matallah H, Belalem G, Bouamrane K. Estudio comparativo entre la base de datos relacional MySQL y la base de datos MongoDB NoSQL. Revista Internacional de Ciencia del Software e Inteligencia Computacional. 2021;13(3):38–63. Disponible en: https://doi.org/10.4018/IJSSCI.2021070104.