LATAM Revista Latinoamericana de Ciencias Sociales y Humanidades, Asunción, Paraguay.
ISSN en línea: 2789-3855, 2025, Volumen VI

Rendimiento entre dos bases de datos no relacionales MongoDB y Redis en una Aplicación Web con microservicios

Performance between two non-relational databases, MongoDB and Redis, in a microservices web application

Pedro Aguilar Encarnación
ps.aguilare@uea.edu.ec
https://orcid.org/0009-0005-1664-2280
Universidad Estatal Amazónica
Puyo – Ecuador

Bernabé Ortega Tenezaca
bortega@uea.edu.ec
https://orcid.org/0000-0001-9693-6951
Universidad Estatal Amazónica
Puyo – Ecuador

Janick Cevallos Illicachi
jr.cevallosi@uea.edu.ec
https://orcid.org/0000-0002-1138-7357
Universidad Estatal Amazónica
Puyo – Ecuador

Efraín Merino Moya
ae.merinom@uea.edu.ec
https://orcid.org/0009-0000-0898-7357
Universidad Estatal Amazónica
Puyo – Ecuador

DOI: https://doi.org/10.56712/latam.v6i5.4715

Artículo recibido: 06 de julio de 2025
Aceptado para publicación: 28 de octubre de 2025.
Conflictos de Interés: Ninguno que declarar.

Redilat
Red de Investigadores Latinoamericanos

NÚMERO 5

DOI: https://doi.org/10.56712/latam.v6i5.4715

Rendimiento entre dos bases de datos no relacionales MongoDB y Redis en una Aplicación Web con microservicios

Performance between two non-relational databases, MongoDB and Redis, in a microservices web application

Pedro Aguilar Encarnación
ps.aguilare@uea.edu.ec
https://orcid.org/0009-0005-1664-2280
Universidad Estatal Amazónica
Puyo – Ecuador

Bernabé Ortega Tenezaca
bortega@uea.edu.ec
https://orcid.org/0000-0001-9693-6951
Universidad Estatal Amazónica
Puyo – Ecuador

Janick Cevallos Illicachi
jr.cevallosi@uea.edu.ec
https://orcid.org/0000-0002-1138-7357
Universidad Estatal Amazónica
Puyo – Ecuador

Efraín Merino Moya
ae.merinom@uea.edu.ec
https://orcid.org/0009-0000-0898-2430
Universidad Estatal Amazónica
Puyo – Ecuador

Artículo recibido: 06 de julio de 2025. Aceptado para publicación: 28 de octubre de 2025.
Conflictos de Interés: Ninguno que declarar.

Resumen

En el presente artículo científico se analiza el rendimiento de dos bases de datos utilizadas en aplicaciones web modernas, MongoDB y Redis. El estudio se centra en el comportamiento como parte de una aplicación web de micro directorio de bibliotecas, desarrollada con arquitectura de microservicios. La investigación se justifica en la necesidad de tomar decisiones informadas con respecto a la elección y uso de tecnologías que impactan directamente la eficiencia, escalabilidad y experiencia del usuario. El objetivo principal fue evaluar el desempeño de ambas bases de datos en un entorno serverless, considerando su impacto en el diseño y desarrollo de aplicaciones web. Se adoptó una metodología experimental, con un enfoque cuantitativo y un alcance comparativo. Para ello, se realizaron pruebas de carga y latencia utilizando la herramienta Apache JMeter, diseñando escenarios específicos que permitieron obtener resultados reproducibles y comparables. Entre los hallazgos relevantes se observó que MongoDB presentó un tiempo de carga inicial más rápido, mientras que Redis mostró una menor latencia en tiempos de respuesta. No obstante, Redis consumió más recursos de procesamiento y memoria que MongoDB. La discusión evidenció que, aunque ambos sistemas ofrecen ventajas particulares, la elección entre uno u otro debe considerar tanto el rendimiento como el uso de recursos según las necesidades específicas de la aplicación. En conclusión, este estudio permitió proporcionar una guía práctica para desarrolladores y profesionales en tecnología, destacando la importancia de equilibrar velocidad de respuesta y eficiencia en el consumo de recursos al seleccionar una base de datos.

Palabras clave: base de datos, latencia, tiempo de carga, microservicios, redis

Abstract

This scientific article analyzes the performance of two databases commonly used in modern web applications: MongoDB and Redis. The study focuses on their behavior within a micro-library directory web application developed using a microservices architecture. The research is justified by the need to make informed decisions regarding the selection and use of technologies that directly affect efficiency, scalability, and user experience. The primary objective was to evaluate the performance of both databases in a serverless environment, considering their impact on the design and development of web applications. An experimental methodology was adopted, with a quantitative approach and a comparative scope. Load and latency tests were conducted using the Apache JMeter tool, with specifically designed scenarios that enabled reproducible and comparable results. Among the key findings, MongoDB demonstrated a faster initial load time, while Redis exhibited lower response latency. However, Redis consumed more processing power and memory than MongoDB. The discussion revealed that although both systems offer distinct advantages, the choice between them should consider both performance and resource consumption, depending on the specific requirements of the application. In conclusion, this study provides practical guidance for developers and technology professionals, emphasizing the importance of balancing response speed and resource efficiency when selecting a database.

Keywords: database, mongodb, latency, load time, microservices, redis

Cómo citar: Aguilar Encarnación, P., Ortega Tenezaca, B., Cevallos Illicachi, J., & Merino Moya, E. (2025). Rendimiento entre dos bases de datos no relacionales MongoDB y Redis en una Aplicación Web con microservicios. *LATAM Revista Latinoamericana de Ciencias Sociales y Humanidades 6 (5), 1936 – 1948. https://doi.org/10.56712/latam.v6i5.4715*

INTRODUCCIÓN

Estudios previos comparativos de bases de datos NoSQL y sus predecesoras relacionales como Thakaere et al. (2023) y da Silva & Lima (2023) establecen que las bases de datos NoSQL son más adecuadas que las bases de datos relacionales en procesamiento de Big Data y en el soporte de aplicaciones de gran escala. De igual manera se reafirma la premisa del teorema CAP, en la cual se afirma la relación inversa que indica que, un mayor nivel de consistencia influye directamente en el rendimiento y latencia. Gorbenko & Trasyunk (2020) y Rmis & Topcu (2020) por otra parte, demuestran en sus estudios, la cuantificación entre consistencia, disponibilidad y latencia, con lo cual se contribuye a sentar las bases metodológicas en la evaluación de sistemas distribuidos.

Las investigaciones centradas en las implementaciones que evalúan el rendimiento de MongoDB demuestran que su uso es versátil y robusto, sobre todo en el uso de aplicaciones de salud como telemedicina y cuidados paliativos, análisis deportivo y gestión de residuos, en las cuales destacan la capacidad para el manejo de datos complejos y dinámicos, Andreoli et al. (2021) realiza estudios de mejoras en el rendimiento de MongoDB, adicionalmente Ingo & Daly (2020) realiza pruebas internas exhaustivas para evidenciar la madurez y comportamiento en entornos de producción.

Balakayeva et al., (2019) desarrolla un estudio en el cual procesa grandes volúmenes de datos no estructurados, donde se demuestra que MongoDB presenta mejores características como simplicidad, eficiencia. De igual manera destacan la posibilidad de usar únicamente dos tablas con enlaces mutuos debido a que simplifica la resolución de problemas altamente complejos. Por otro lado Al-Wadi & Maaita, (2023) desarrolla una evaluación de alto rendimiento para la autenticación basada en contaseñas para arquitecturas de microservicios, determina que las bases de datos NoSQL son adecuadas para sistemas pequeños (1500 usuarios o menos), en tanto que las bases de datos SQL son más adecuadas para sistemas de escala media a grande, sin embargo, no existen diferencias significativas en el rendimiento entre ambas bases de datos.

Las aplicaciones en tiempo real requieren un control de latencia eficaz, ante lo cual se propone una versión modificada de MongoDB denominada RT-MongoDB la cual prioriza por cliente o por solicitud. Andreoli et al, (2021) realiza experimentaciones que concluyen que las solicitudes de mayor prioridad generan tiempos de respuesta muy estables y reducidos. Por otro lado un estudio sobre un sistema de software de registro de sensores para investigación biomédica, desarrollado en microservicios en contenedores Docker, en el que utiliza MongoDB, en el cual se logra confiabilidad con una tasa de rendimiento de 2000 solicitudes por segundo con baja latencia, e incluso reducción de costos (Single et al., 2023).

Boza et al., (2020) desarrollan un estudio en la cual se optimizan las caches en nube mediante computación sin servidor, para lo cual se utilizó una versión modificada de Redis(SPREDS) para la gestión de memoría. Su utilización demuestra un ahorro de costos significativos al optimizar problemas de asignación de memora en entornos Redis con varias instancias, sin embargo, Meng et al., (2021) desarrolla una evaluación de seguridad, en la cual se descubren vulnerabilidades en sistemas distribuidos que fueron probados de manera rigurosa, donde se destaca la importancia de pruebas de seguridad en este tipo de entornos.

Kim & Lee, (Kim et al., 2021) plantea una politica de replicación denominada ROVN la cual fue implementada en Redis en la cual se demuestra que se puede lograr ahorros de costros al colocar réplicas en memoria sin que ello conlleve a comprometer el rendimiento percibido por el usuario. Por otra parte Niswar et al., (2024) realiza un estudio sobre la comunicación entre microservicios que contenian bases de datos Redis y MySQL, con lo cual se ilustra un escenario de uso conjunto. La investigación se centró en los protocolos de investigación de comunicación con lo cual se confirma la coexistencia y uso complementario de estas bases de datos dentro de una misma arquitectura.

De acuerdo con Privalov & Stupina, (2024) y Esteves & Fernandez, (2019) Redis presenta una mejora significativa en su rendimiento en aplicaciones web, donde se reduce la latencia, usándolo como capa de caché en memoria. En cuanto a las arquitecturas de microservicios, Redis Streams permite comunicación asincrónica entre servicios, reduciendo problemas de latencia (Weerasinghe & Perera, 2023). Existe documentación sobre el uso en conjunto de Redis y MongoDB en aplicaciones de gestión de residuos, donde no se consideran estudios comparativos de roles. La latencia en la comunicación entre servicios y el llamado "efecto domino" son desafíos clave en microservicios, lo cual da relevancia a la optimización de la capa de datos (Santos et al., 2025; Weerasinghe & Perera, 2023).

Se identifica una brecha en las investigaciones sobre el uso de bases de datos NoSQL, los estudios revisados, examinan sus características de manera generalizada, en donde se toma en cuenta la persistencia de MongoDB y la evaluación del caché de Redis, sin embargo, no existe una comparativa directa o empírica del rendimiento dentro del mismo entorno de microservicios. Adicionalmente, el debate sobre Redis como almacenamiento principal (Adya et al., 2019) refuerza la necesidad de delimitar su rol como solución de alto rendimiento para tareas de apoyo.

La constante y creciente de arquitecturas de microservicios, de popularidad y del uso de las bases de datos NoSQL, requiere evaluaciones con rigurosidad sobre las tecnologías que impulsan su rendimiento. Tanto MongoDB como Redis en aplicaciones web con microservicios, requieren de razones ampliamente justificadas y estudiadas puesto que su elección no es simplemente una preferencia, muy por el contrario, corresponde a compromisos fundamentales en términos de latencia, rendimiento y escalabilidad. Se requiere que la evaluación del desempeño de cada base de datos en sus roles específicos como persistencia de documentos para MongoDB y cache comunicación para Redis, es fundamental y de nivel crítico para el diseño de sistemas eficientes y competitivos.

El objetivo principal de la presente investigación es la cuantificación del rendimiento donde se evalúa y compara empíricamente la latencia y rendimiento (throughput) de las operaciones de lectura y escritura en una base de datos documental (MongoDB) y un almacén de valores clave en minoría (Redis). La evaluación propuesta se realiza en un contexto de una aplicación web de microservicios con cargas de trabajo simuladas para reflejar escenarios típicos de uso.

METODOLOGÍA

La presente investigación se enmarca en la Ciencia del Diseño (Design Science Research, DSR), siguiendo las directrices de Hevner, que establecen la creación y evaluación de artefactos tecnológicos como medios para generar conocimiento científico aplicable. El estudio se desarrolló mediante la construcción de un prototipo funcional de aplicación web con arquitectura de microservicios en entornos serverless, cuyo objetivo fue comparar el rendimiento de las bases de datos NoSQL MongoDB y Redis en un entorno serverless, desplegado en la plataforma Vercel.

Diseño del estudio y contexto

El diseño adoptado fue experimental y comparativo, mediante la implementación de dos entornos equivalentes diferenciados únicamente por el motor de base de datos. El caso de uso seleccionado fue un micro-directorio de bibliotecas, integrado por microservicios de autenticación de usuarios y gestión de sugerencias de libros.

El contexto —aplicaciones web modernas basadas en microservicios y ejecutadas en plataformas serverless— responde a la necesidad de evaluar tecnologías NoSQL en escenarios con alta demanda de escalabilidad, baja latencia y eficiencia en el uso de recursos.

La pregunta de investigación que guió esta investigación fue:

- ¿Qué base de datos no relacional, MongoDB o Redis, ofrece un mejor rendimiento en términos de tiempo de carga, latencia y uso de recursos en una aplicación web de microservicios desplegada en un entorno serverless?

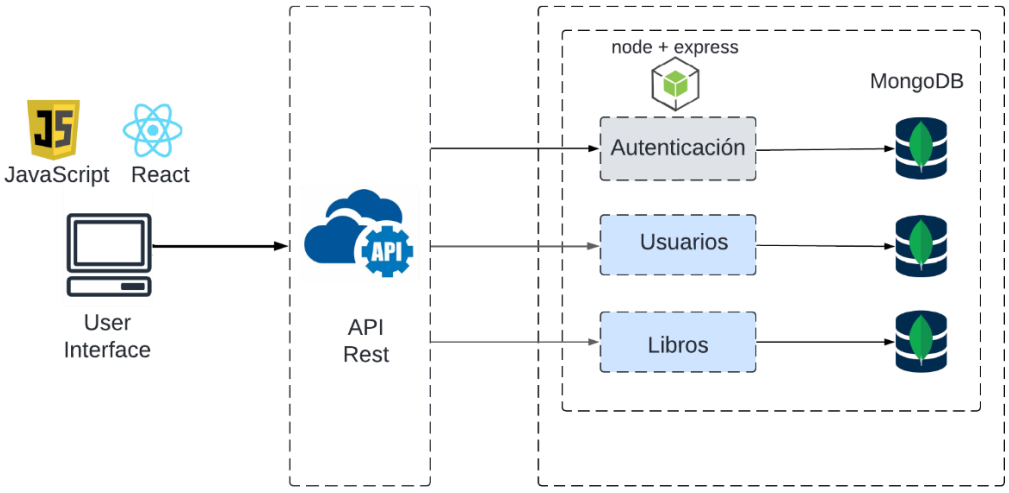
Construcción del artefacto

El artefacto central de este estudio fue un prototipo de aplicación denominado Micro-Directorio de Bibliotecas, concebido bajo una arquitectura de microservicios. Este prototipo integró dos módulos funcionales: un servicio de autenticación de usuarios (registro e inicio de sesión) y un servicio de gestión de sugerencias de libros (operaciones CRUD). Ambos microservicios fueron desarrollados en Node.js con Express.js, garantizando homogeneidad en la lógica de negocio y permitiendo aislar la base de datos como único factor diferenciador.

El frontend se implementó con React, HTML, CSS y JavaScript, mientras que el backend se desplegó en la plataforma Vercel, bajo un esquema serverless. Para asegurar la comparabilidad, se construyeron dos versiones idénticas del prototipo, diferenciadas únicamente por el motor de base de datos empleado: MongoDB (Figura 1) y Redis (Figura 2).

Figura 1

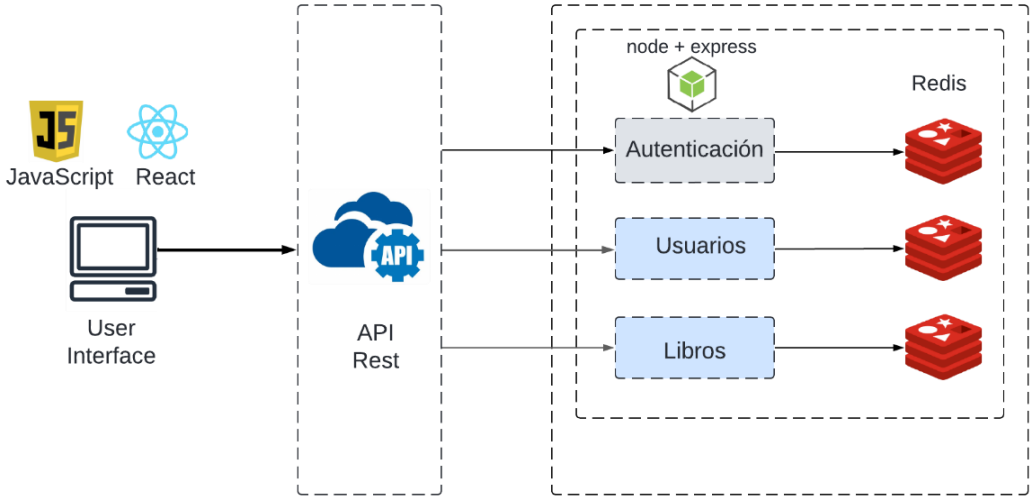
Stack Tecnológico y arquitectura de la aplicación web con el motor de base de datos MongoDB



Como soporte para la validación de la funcionalidad del sistema se utilizaron herramientas de verificación de endpoints RESTful (Postman e Insomnia), mientras que la infraestructura experimental se ejecutó en un entorno controlado con CPU multinúcleo, 16 GB de memoria RAM y almacenamiento SSD, lo que permitió mitigar sesgos atribuibles a limitaciones de hardware.

Figura 2

Stack Tecnológico y arquitectura de la aplicación web con el motor de base de datos Redis



Finalmente, se adoptaron prácticas de transparencia y ética informática, entre ellas: la preservación de los datos de prueba, el uso de entornos controlados para garantizar imparcialidad en las mediciones y la publicación del código fuente en un repositorio abierto de GitHub, facilitando la reproducibilidad del experimento.

Diseño del experimento

El diseño experimental se estructuró en fases que garantizaron rigurosidad y reproducibilidad. En primer lugar, se configuraron dos entornos paralelos, replicando de manera idéntica la arquitectura de despliegue y diferenciándolos únicamente por el motor de base de datos utilizado (MongoDB o Redis).

Posteriormente, se definieron los escenarios de prueba, concebidos para simular interacciones reales de usuarios en un sistema web de microservicios. Para este fin, se simularon 100 peticiones concurrentes mediante Apache JMeter, abarcando tanto procesos de autenticación como la ejecución de operaciones CRUD. Esta configuración permite someter a los sistemas a una carga representativa de uso intensivo y evaluar su comportamiento en situaciones de concurrencia elevada.

La ejecución de las pruebas siguió un protocolo de repetición, en el que cada escenario se corrió en tres iteraciones independientes, a fin de reforzar la consistencia de los resultados y descartar fluctuaciones derivadas de factores externos. Durante las ejecuciones, se recopilaron métricas claves asociadas al tiempo de carga inicial, latencia promedio y uso de recursos del sistema, lo que permitió obtener una visión integral del rendimiento.

Finalmente, los resultados fueron organizados en tablas comparativas y representaciones gráficas, las cuales facilitaron la identificación de patrones de comportamiento y el contraste de desempeño entre MongoDB y Redis bajo condiciones equivalentes de carga.

Variables y medidas

La variable independiente correspondió al motor de base de datos empleado (MongoDB vs Redis). Como variables dependientes, se midieron tres dimensiones críticas:

Tiempo de carga inicial (ms): preparación de configuraciones y primera respuesta.

Latencia promedio (ms): en operaciones CRUD y procesos de autenticación.

Uso de recursos del sistema: consumo de CPU, RAM y almacenamiento en disco (%).

Para la evaluación se utilizaron herramientas especializadas: Apache JMeter, encargado de simular cargas concurrentes y extraer métricas de rendimiento; y PerfMon, que permitió registrar el uso de CPU, memoria y disco durante la ejecución de las pruebas. Se recopilaron métricas de tendencia central y dispersión (media, mínimo, máximo, desviación estándar), además de indicadores de eficiencia como el rendimiento en solicitudes por segundo (req/s) y porcentaje de error.

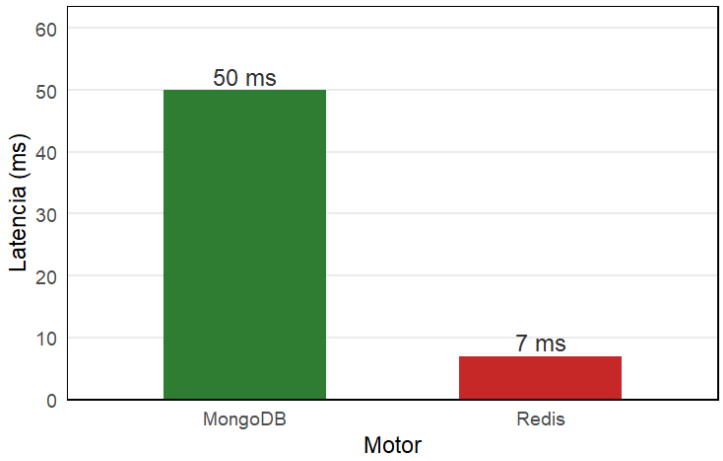
RESULTADOS

El estudio evidencia diferencias clave entre MongoDB y Redis al integrarse en una arquitectura web de microservicios. Los hallazgos se presentan en las siguientes dimensiones: tiempo de carga inicial y latencia (ambos expresados en milisegundos), y uso de recursos del sistema (CPU, RAM y almacenamiento en disco).

Latencia: En las pruebas de rendimiento se compararon los tiempos de respuesta de ambos motores, observándose una diferencia significativa: MongoDB registró una latencia promedio de 50 ms, mientras que Redis alcanzó 7 ms. Este resultado confirma que Redis ofrece una respuesta más ágil, particularmente adecuada para operaciones en tiempo real.

Gráfico 1

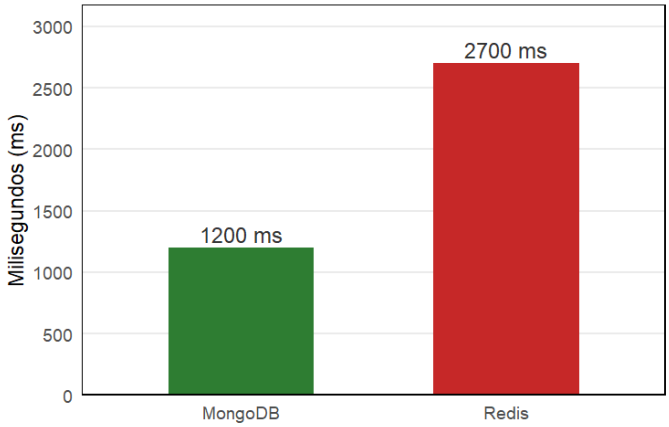
Latencia promedio (ms) de MongoDB y Redis



Tiempo de carga inicial. Bajo el mismo escenario de arranque y 100 transacciones, MongoDB completó el proceso en 1 200 ms y Redis en 2 700 ms (brecha: 1 500 ms). El diferencial es consistente con una inicialización más ligera en MongoDB, frente a Redis, cuyo diseño en memoria puede requerir rehidratación desde AOF/RDB y preasignación de memoria para garantizar latencias ultra-bajas en operación, encareciendo el boot. En entornos con arranques frecuentes (autoscaling/serverless), este comportamiento favorece el uso de MongoDB como almacenamiento primario, reservando Redis como capa de aceleración allí donde la latencia en ejecución sea crítica.

Gráfico 2

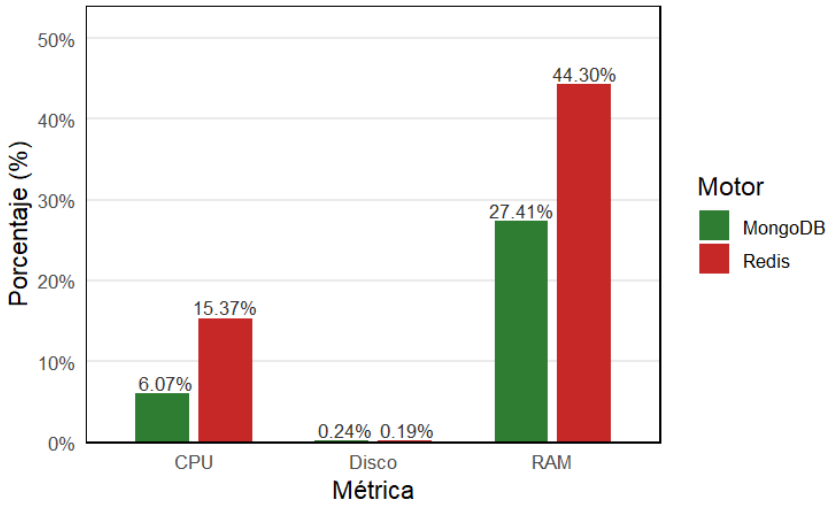
Tiempo de carga inicial (ms) de MongoDB y Redis



Uso de recursos. Bajo una carga equivalente, Redis mostró un mayor consumo de CPU y memoria que MongoDB. En CPU, Redis registró 15,37 % frente a 6,07 % de MongoDB; en RAM, 44,30 % frente a 27,41 %; y en disco, la diferencia fue marginal (0,19 % en Redis vs. 0,24 % en MongoDB). En conjunto, los valores describen un trade-off claro: Redis prioriza la rapidez de respuesta a costa de una huella operativa mayor, mientras que MongoDB mantiene un consumo de recursos más contenido.El presente estudio evidencia diferencias clave entre MongoDB y Redis al integrarse en una arquitectura web de microservicios. A continuación, se presentan los resultados en dos dimensiones: (i) tiempo de carga inicial (expresado en segundos) y latencia (en milisegundos), y (ii) uso de recursos del sistema, considerando CPU, RAM y almacenamiento en disco.

Gráfico 3

Uso de recursos del sistema (%): CPU, RAM y disco



Relación con estudios anteriores

Los resultados están de acuerdo con el resultado del estudio anterior. Por ejemplo, el análisis de análisis de rendimiento (Lennin y Loyo, 2020) en su disertación "Estudio comparativo de la base de datos NSOQL, que permite analizar la respuesta a los datos en el prototipo de red social de la universidad" disminuye con Redis durante la reacción. El trabajo también proporciona (Nayak et al., 2013) características de bases de NoSQL del sistema general en comparación con el relativo, lo que respalda la comprensión de las fortalezas específicas observadas en cada una.

Redis se encuentra como una solución efectiva en contexto, donde la latencia y al mismo tiempo son factores críticos, como aplicaciones de tiempo real, mensajes inmediatos o servicios que requieren una respuesta inmediata. En contraste, MongoDB es más adecuado para aplicaciones que requieren una carga inicial eficiente y un control de datos estructurado más complejo. Estudiar restricciones y pronósticos futuros

Cabe señalar que las pruebas se realizaron en un entorno controlado que podría limitar los resultados de los resultados al entorno de producción real. Además, factores como las especificaciones de hardware y la configuración de la prueba afectan el rendimiento medido. Se recomienda más investigación para analizar la creación de ambas bases de datos en un entorno dividido, evaluando su escalabilidad, tolerancia a la falla y rendimiento en varias cargas.

DISCUSIÓN

Interpretación de los Resultados

Según el estudio nos revela que MongoDB presentó un tiempo de carga inicial más rápido (1.2s), mientras que Redis nos demostró menor latencia en las operaciones de lectura y escritura (7 ms frente a 50 ms en MongoDB). Estos resultados coinciden con investigaciones previas (Lennin & Loyo, 2020; Nayak et al., 2013) mismo que destaca la eficiencia de Redis en contextos donde la velocidad de respuesta es crítica, así como la versatilidad de MongoDB para así poder manejar datos más elaborados de más dificultad y estructurados. La literatura reciente (Andreoli et al., 2021; Balakayeva et al., 2019) también respalda la idea de que MongoDB resulta más robusto en la gestión de grandes volúmenes de información, mientras que Redis se orienta al desempeño inmediato mediante almacenamiento en memoria.

En el análisis comparativo se puede confirmar que la elección de la base de datos no se puede depender únicamente de las métricas, datos de latencia y carga, sino de factores como equilibrio entre eficiencia, complejidad de datos y consumo de recursos. En este estudio, Redis nos pudo mostrar mayor consumo de CPU y memoria (15.37% y 44.30%, respectivamente), lo que encaja con la arquitectura en memoria, pero también compromete a ejecutar aplicaciones con recursos limitados.

Implicaciones

En términos teóricos, estos hallazgos refuerzan la validez del teorema CAP, que establece la relación inversa entre consistencia y rendimiento en sistemas distribuidos (Gorbenko & Tarasyuk, 2020). Asimismo, aportan evidencia empírica a la discusión sobre el uso diferenciado de MongoDB y Redis en arquitecturas de microservicios con entornos serverles. De otro lado donde lo vemos en el ámbito práctico, los resultados ofrecen a desarrolladores y arquitectos de software una guía clara:

Redis es más conveniente en aplicaciones en tiempo real, mensajería instantánea o servicios que requieren mínima latencia.

MongoDB resulta más adecuado para aplicaciones que manejan datos más estructurados, con requerimientos de inicio rápido y menor consumo de recursos.

También en el estudio sugiere la posibilidad de combinación hibrida de ambas tecnologías, aprovechando MongoDB como repositorio principal de datos y Redis como capa de caché o una comunicación asincrónica, estrategia que la literatura ya documentada en contexto de optimización de microservicios (Privalov & Stupina, 2024; Weerasinghe & Perera, 2023).

Limitaciones

El experimento se desarrolló en un entorno controlado, con hardware y parámetros de configuración específicos, lo que limita la generalización de los resultados hacia escenarios de productos heterogéneos. Factores como el escalado horizontal, la tolerancia a fallos, la coexistencia con otros servicios distribuidos o las variaciones en la carga rel de nuestros usuarios no fueren vulneradas. Asimismo, el estudio no exploró aspectos de seguridad ni vulnerabilidad, que han sido identificados como relevantes en investigaciones previas (Meng et al., 2021).

Trabajos futuros

De cara a trabajos futuros, conviene caracterizar la escalabilidad de MongoDB y Redis bajo cargas masivas y prolongadas, en entornos distribuidos y multirregionales, para trazar su comportamiento más allá de escenarios controlados. Del mismo modo, resulta necesario incorporar métricas de tolerancia a fallos y resiliencia —tiempos de detección y conmutación, continuidad del servicio—, especialmente en sistemas con alta disponibilidad. También merece atención la exploración de configuraciones híbridas, donde ambos motores coexistan dentro de un mismo ecosistema de microservicios, evaluando sus efectos en consistencia, latencia y costo. Finalmente, se propone profundizar en seguridad y optimización energética, dado que el consumo de recursos emerge como factor crítico para la sostenibilidad de la infraestructura digital.

CONCLUSIÓN

Este estudio tuvo como propósito comparar cómo funcionan MongoDB y Redis, dos sistemas de gestión de bases de datos no relacionales, dentro de una aplicación web estructurada mediante microservicios. Para ello, se empleó como herramienta JMeter que permitió hacer pruebas controladas sobre el tiempo de carga, latencias en las respuestas y consumo de recursos como memoria y capacidad de procesamiento. Los resultados permiten comprender el comportamiento de esas tecnologías en entornos distribuidos y serverless, facilitando la toma de decisiones informadas en el desarrollo de aplicaciones web modernas.

Se identificaron diferencias significativas entre MongoDB y Redis. MongoDB proporciona un tiempo de carga de arranque inicial más rápido, mientras que Redis ofrece tiempos de respuesta a solicitudes durante la ejecución. No obstante, Redis también incrementa el consumo de recursos del sistema en específico memoria y capacidad de procesamiento. Esto tiene sentido existe una coherencia con Redis, el cual fue desarrollado con un diseño orientado a la velocidad, priorizando el rendimiento en correspondencia de una mayor demanda de recursos. En cambio, MongoDB demostró una mejora y capacidad en escenarios que requieren un manejo de estructuras de datos más complejas, ideal para aplicaciones con mayor requerimiento de almacenamiento y consultas avanzadas.

Los resultados son consistentes con investigaciones previas, en las que se señalan a Redis como especialmente útil en escenarios de alta carga que requieren una lata velocidad de respuesta. Igualmente, se confirma que MongoDB tiene un mayor desempeño en contextos que requieren un manejo de datos complejos estructuralmente, dando respuesta a necesidades de almacenamiento robustas.

Las pruebas se realizaron en condiciones controladas. Factores como la capacidad computacional, configuración del sistema y la metodología empleada en pruebas, pueden influir en el rendimiento observado. Sin embargo, los resultados presentados pueden constituir una base útil para que programadores y desarrolladores seleccionen una base de datos más adecuada según requisitos específicos. Si el objetivo es optimizar la velocidad de respuesta ante solicitudes simultáneas, Redis puede resultar muy eficiente. En cambio, si el objetivo que se persigue es gestionar información compleja y garantizar un arranque ágil del sistema, MongoDB puede resultar muy conveniente. Finalmente es posible contar con la integración de ambas tecnologías para aprovechar sus fortalezas de manera complementaria otorgando flexibilidad a los sistemas.

REFERENCIAS

Adya, A., Grandl, R., Myers, D., & Qin, H. (2019). Fast key-value stores: An idea whose time has come and gone. Proc. Workshop Hot Top. Oper. Syst., HotOS, 113-119. Scopus. https://doi.org/10.1145/3317550.3321434

Al-Wadi, R. A., & Maaita, A. A. (2023). Authentication and Role-Based Authorization in Microservice Architecture: A Generic Performance-Centric Design. Journal of Advances in Information Technology, 14(4), 758-768. Scopus. https://doi.org/10.12720/jait.14.4.758-768

Andreoli, R., Cucinotta, T., & Pedreschi, D. (2021). RT-MongoDB: A NoSQL Database with Differentiated Performance. En Helfert M., Ferguson D., & Pahl C. (Eds.), International Conference on Cloud Computing and Services Science, CLOSER - Proceedings (Vols. 2021-April, pp. 77-86). Science and Technology Publications, Lda; Scopus. https://doi.org/10.5220/0010452400770086

Balakayeva, G. T., Phillips, C., Darkenbayev, D. K., & Turdaliyev, M. (2019). Using NoSQL for processing unstructured big data. News of the National Academy of Sciences of the Republic of Kazakhstan, Series of Geology and Technical Sciences, 6(438), 12-21. Scopus. https://doi.org/10.32014/2019.2518-170X.151

Boza, E. F., Andrade, X., Cedeno, J., Murillo, J., Aragon, H., Abad, C. L., & Abad, A. G. (2020). On implementing autonomic systems with a serverless computing approach: The case of self-partitioning cloud caches. Computers, 9(1). Scopus. https://doi.org/10.3390/computers9010014

Da Silva, J. (2023). Protection, expertise and domination: Cyber masculinity in practice. Computers and Security, 133. Scopus. https://doi.org/10.1016/j.cose.2023.103408

Esteves, A., & Fernandes, J. (2019). Improving the latency of python-based web applications. En Bozzon A., Mayo F.J.D., & Filipe J. (Eds.), WEBIST - Proc. Int. Conf. Web Inf. Syst. Technol. (pp. 193-201). SciTePress; Scopus. https://doi.org/10.5220/0007959401930201

Gorbenko, A., & Tarasyuk, O. (2020). EXPLORING TIMEOUT AS A PERFORMANCE AND AVAILABILITY FACTOR OF DISTRIBUTED REPLICATED DATABASE SYSTEMS. Radioelectronic and Computer Systems, 4, 98-105. Scopus. https://doi.org/10.32620/reks.2020.4.09

Ingo, H., & Daly, D. (2020). Automated system performance testing at MongoDB. Proc. Workshop Test. Database Syst., DBTest. Proceedings of the Workshop on Testing Database Systems, DBTest 2020. Scopus. https://doi.org/10.1145/3395032.3395323

Kim, K., Shin, Y., Lee, J., & Lee, K. (2021). Automatically attributing mobile threat actors by vectorized ATT&CK matrix and paired indicator. Sensors, 21(19). Scopus. https://doi.org/10.3390/s21196522

Meng, R., Roychoudhury, A., Pîrlea, G., & Sergey, I. (2021). Greybox Fuzzing of Distributed Systems. CCS - Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 1615-1629. Scopus. https://doi.org/10.1145/3576915.3623097

Niswar, M., Safruddin, R. A., Bustamin, A., & Aswad, I. (2024). Performance evaluation of microservices communication with REST, GraphQL, and gRPC. International Journal of Electronics and Telecommunications, 70(2), 429-436. Scopus. https://doi.org/10.24425/ijet.2024.149562

Privalov, M. V., & Stupina, M. V. (2024). Improving web-oriented information systems efficiency using Redis caching mechanisms. Indonesian Journal of Electrical Engineering and Computer Science, 33(3), 1667-1675. Scopus. https://doi.org/10.11591/ijeecs.v33.i3.pp1667-1675

Rmis, A. M., & Topcu, A. E. (2020). Evaluating riak key value cluster for big data. Tehnicki Vjesnik, 27(1), 157-165. Scopus. https://doi.org/10.17559/TV-20180916120558

Santos, J., Reppas, E., Wauters, T., Volckaert, B., & De Turck, F. (2025). Gwydion: Efficient auto-scaling for complex containerized applications in Kubernetes through Reinforcement Learning. Journal of Network and Computer Applications, 234. Scopus. https://doi.org/10.1016/j.jnca.2024.104067

Single, M., Bruhin, L. C., Schütz, N., Naef, A. C., Hegi, H., Reuse, P., Schindler, K. A., Krack, P., Wiest, R., Chan, A., Nef, T., & Gerber, S. M. (2023). Development of an Open-source and Lightweight Sensor Recording Software System for Conducting Biomedical Research: Technical Report. JMIR Formative Research, 7. Scopus. https://doi.org/10.2196/43092

Thakare, A. O., Thakare, A. R., Tembhurne, O. W., & Reddy, S. N. (2023). NoSQL Databases: Modern Data Systems for Big Data Analytics—Features, Categorization and Comparison. International Journal of Electrical and C*omputer Engineering*