Implementación de modelo de datos de sistema académico usando ReplicaSets de MongoDB en un ambiente local y en la nube

Juan Esteban Alarcón Bravo
jalarconb@unal.edu.co

Bases de Datos Avanzadas
Facultad de Ingeniería
Universidad Nacional de Colombia

13 de Junio de 2023

Abstract

En este artículo, exploramos el despliegue de un modelo de datos NoSQL en MongoDB, tanto en un entorno local como en la nube de AWS. Comparamos y contrastamos las ventajas y desafíos de cada enfoque, considerando aspectos como la adaptación del modelo de datos, la obtención y generación de fuentes de datos, y los procesos de despliegue.

En el despliegue local, analizamos la instalación de MongoDB, la gestión la tecnología ODBC y la conexión con un sistema de reporting (Power BI). También abordamos los recursos y limitaciones asociados con este enfoque.

Luego, nos centramos en el despliegue en AWS, utilizando Amazon Lightsail para implementar instancias de MongoDB en la nube. Evaluamos ventajas como escalabilidad, disponibilidad, seguridad y facilidad de administración, además de consideraciones económicas.

Concluimos resaltando las fortalezas y limitaciones de cada enfoque y ofreciendo alternativas para un trabajo futuro que pueda ser desarrollado con las bases de este artículo.

Keywords *— MongoDB, replicación, despliegue, AWS, Lightsail.*

1. Introducción

En la era digital actual, el manejo eficiente y escalable de grandes volúmenes de datos se ha convertido en una prioridad para muchas organizaciones [1]. Ante este desafío, las bases de datos NoSQL han emergido como una solución poderosa que permite almacenar, organizar y procesar información de manera flexible y altamente escalable [2].

En este artículo, exploraremos el despliegue de un modelo de datos NoSQL en MongoDB, una de las bases de datos NoSQL más populares y ampliamente utilizadas. Pondremos especial atención en comparar y contrastar el despliegue en un ambiente local versus la implementación en la nube utilizando Amazon Web Services (AWS). Analizaremos las ventajas y desafíos asociados con cada opción, así como las consideraciones clave a tener en cuenta al elegir la plataforma adecuada para nuestro modelo de datos.

Comenzaremos realizando la adaptación del modelo de datos utilizado en trabajos previos [3]. Luego, abordaremos lo relacionado con las fuentes de datos, su obtención y generación. Posteriormente, discutiremos los procesos de despliegue en un ambiente local. Exploraremos aspectos como la instalación de MongoDB, la creación y gestión de bases de datos y colecciones,

así como el diseño e implementación de índices para optimizar el rendimiento de nuestras consultas. Además, consideraremos los recursos y limitaciones inherentes a este enfoque, y cómo podemos superarlos para maximizar el potencial de nuestro modelo de datos.

A continuación, dirigiremos nuestra atención hacia AWS, uno de los principales proveedores de servicios en la nube. Exploraremos cómo desplegar nuestro modelo de datos NoSQL en un entorno gestionado por AWS, aprovechando los servicios de Amazon Lightsail para la implementación de instancias de MongoDB. Evaluaremos las ventajas de esta aproximación en términos de escalabilidad, disponibilidad, seguridad y facilidad de administración, así como las consideraciones económicas asociadas.

Finalmente, concluiremos nuestro análisis comparativo destacando las fortalezas y limitaciones de cada enfoque. Proporcionaremos recomendaciones prácticas basadas en las necesidades y recursos de su proyecto, para ayudar a tomar una decisión informada sobre el despliegue de un modelo de datos NoSQL en MongoDB.

2. El Modelo NoSQL

2.1. Ventajas del modelo NoSQL frente al modelo SQL

El almacenamiento y la gestión de datos han evolucionado significativamente en los últimos años, y el modelo NoSQL ha surgido como una alternativa poderosa al tradicional modelo SQL [1]. A medida que las organizaciones lidian con la explosión de datos y los requisitos de escalabilidad, la flexibilidad del modelo NoSQL ha demostrado ser altamente beneficiosa [2]. A continuación, se presentarán algunas ventajas clave al utilizar el modelo NoSQL sobre el modelo SQL para almacenar datos:

Escalabilidad horizontal

El modelo NoSQL está diseñado para escalar horizontalmente de manera más sencilla [4]. A diferencia del modelo SQL, que generalmente requiere particionar y ajustar esquemas de datos existentes, el modelo NoSQL permite agregar nuevos nodos de almacenamiento sin alterar el rendimiento o la estructura de datos existente. Esto facilita la gestión de grandes volúmenes de datos y permite un crecimiento fluido a medida que las necesidades aumentan [5].

Esquema flexible

Mientras que en el modelo SQL se requiere un esquema rígido y predefinido, el modelo NoSQL permite un enfoque más flexible. No se requiere un diseño de tabla fijo, lo que significa que los campos y estructuras pueden variar entre documentos dentro de una colección. Esta flexibilidad permite adaptarse rápidamente a los cambios en los requisitos y evita la necesidad de realizar modificaciones costosas en la estructura de la base de datos.

Mayor rendimiento

El modelo NoSQL está diseñado para operaciones de lectura y escritura de alto rendimiento. Al evitar las operaciones de unión complejas y la normalización extrema, el modelo NoSQL puede ofrecer un rendimiento más rápido y eficiente para ciertos tipos de consultas y cargas de trabajo. Además, la distribución de datos y la replicación permiten una disponibilidad y rendimiento superiores, lo que resulta en tiempos de respuesta más rápidos y una mejor experiencia de usuario.

Datos no estructurados

El modelo NoSQL es particularmente adecuado para almacenar y gestionar datos no estructurados o semi estructurados [6], como documentos JSON, registros de eventos, datos de redes sociales y contenido multimedia. Dado que estos tipos de datos no siguen un formato fijo, el modelo NoSQL permite una manipulación y consulta más sencilla de dicha información, a diferencia del modelo SQL, que puede ser más

restrictivo en cuanto a la estructura y el formato de los datos.

2.2. Conversión de un modelo de datos SQL a un modelo NoSQL en MongoDB

Al migrar un modelo de datos SQL representado en un diagrama entidad-relación a un modelo NoSQL basado en MongoDB, se requiere una comprensión clara de las diferencias fundamentales entre los dos enfoques. A continuación, se explicará el proceso de conversión, centrándose en la transformación de las tablas SQL en colecciones MongoDB y las columnas en atributos de documentos NoSQL:

Tablas SQL a colecciones MongoDB

En MongoDB, los datos se organizan en colecciones, que son equivalentes a las tablas SQL. Cada documento en una colección representa una instancia de datos, similar a una fila en una tabla SQL. Al migrar un modelo de datos SQL, es necesario identificar las tablas y convertirlas en colecciones correspondientes en MongoDB.

Columnas SQL a atributos de documentos NoSQL

Una vez que se han creado las colecciones en MongoDB, es necesario mapear las columnas de las tablas SQL a los atributos de los documentos NoSQL. Cada columna se convierte en un atributo dentro de un documento, y los datos se almacenan en un formato más flexible, como documentos JSON. Es importante tener en cuenta las relaciones y dependencias entre las tablas SQL para garantizar que los datos se representen correctamente en el modelo NoSQL.

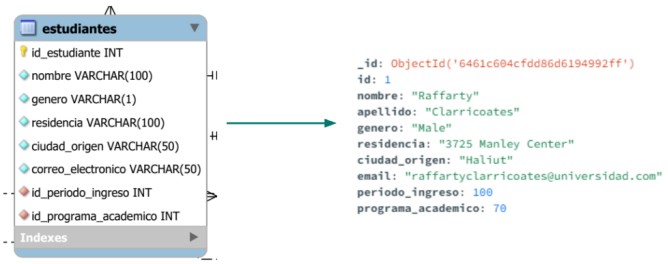

Figura 1: *Transformación del esquema de datos*

3. Acerca de los datos

3.1. Datos reales

En el desarrollo de este proyecto, se utilizaron datos reales de la Universidad Nacional de Colombia para algunos aspectos específicos, como las asignaturas, sedes y facultades. Estos datos se obtuvieron de fuentes públicas, ya que es información abierta y disponible para todas las personas.

El uso de datos reales de la universidad permite que el modelo de datos refleje de manera más precisa los escenarios y requisitos del mundo real. Esto nos brinda la oportunidad de realizar pruebas y análisis más sólidos, basados en datos auténticos, y validar la efectividad y eficiencia de las implementaciones.

3.2. Generación de datos con Mockaroo

Para completar el conjunto de datos necesario para este proyecto, se utilizó la herramienta Mockaroo. Mockaroo es una plataforma en línea que permite generar datos ficticios de manera masiva y personalizada, simulando diferentes tipos de información de forma aleatoria.

El proceso de generación de datos con Mockaroo implica seguir los siguientes pasos:

Definir la estructura del esquema de datos

Primero, se debe definir la estructura del esquema de datos que deseamos generar. Esto incluye la especificación de los campos, tipos de datos, restricciones y relaciones entre ellos. Por

ejemplo, para un modelo de datos relacionado con estudiantes, podríamos tener campos como nombre, edad, dirección y asignaturas cursadas.

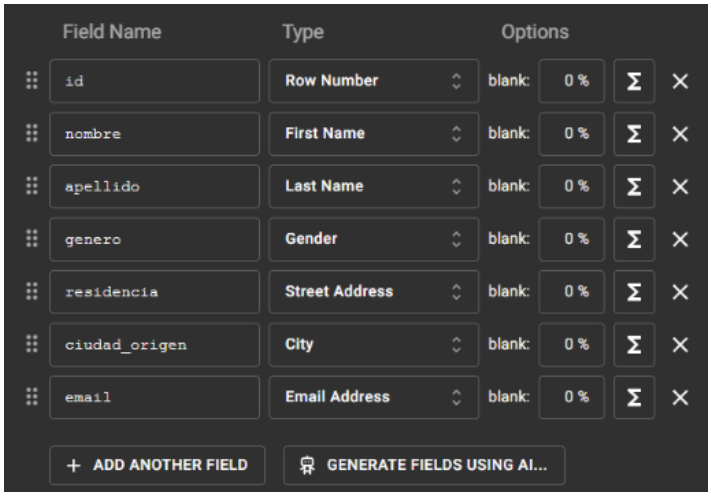

Figura 2: *Esquema de datos en Mockaroo*

Configurar las reglas de generación de datos

Una vez que se ha definido la estructura del esquema de datos, se deben configurar las reglas de generación para cada campo. Mockaroo ofrece una amplia gama de opciones, como generar datos aleatorios basados en patrones predefinidos, utilizar listas de valores personalizadas o incluso importar datos existentes.

Generar los datos en masa

Una vez que se han configurado todas las reglas de generación de datos, se puede proceder a la generación en masa. Mockaroo permite generar grandes volúmenes de datos en diferentes formatos, como CSV, JSON o SQL.

Figura 3: *Datos generados en formato JSON*

Al utilizar Mockaroo y generar datos aleatorios, podemos simular escenarios diversos y realistas que nos ayudan a probar la escalabilidad, rendimiento y eficiencia de nuestro modelo de datos NoSQL.

4. Replica Sets de MongoDB

En MongoDB, los Replica Sets son un componente fundamental que proporciona alta disponibilidad y tolerancia a fallos en el sistema de bases de datos. Un Replica Set es un conjunto de varios servidores de MongoDB interconectados que trabajan juntos para mantener una copia idéntica de los datos. Cada Replica Set incluye un nodo primario y varios nodos secundarios, y puede haber un nodo árbitro opcional para ayudar en la elección del primario en caso de fallas [7].

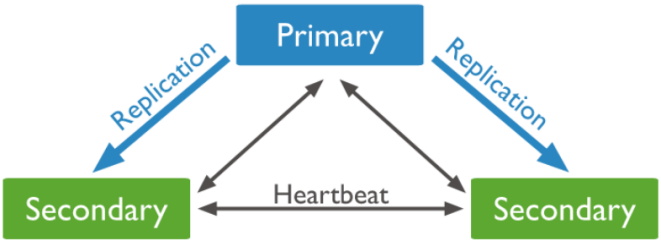

Figura 4: *Esquema básico de un Replica Set de MongoDB*

La principal función de los Replica Sets es ga-

rantizar la disponibilidad continua de los datos en situaciones de fallos de hardware, caídas de servidores o mantenimiento planificado. Cuando un nodo primario experimenta una interrupción, los nodos secundarios pueden elegir a uno de ellos como el nuevo primario para continuar atendiendo las solicitudes de lectura y escritura. Esto asegura que el sistema permanezca en funcionamiento y los datos sean siempre accesibles.

4.1. Ventajas de utilizar replicación

Alta disponibilidad y tolerancia a fallos

Al contar con múltiples copias de los datos distribuidas en nodos secundarios, los Replica Sets garantizan que la base de datos siga funcionando incluso en caso de fallos en el nodo primario. Esto mejora la disponibilidad y minimiza el tiempo de inactividad, evitando interrupciones costosas y asegurando la continuidad del negocio.

Recuperación ante desastres

En situaciones extremas, como fallas totales de servidores o centros de datos completos, los Replica Sets permiten la recuperación rápida y eficiente. Al tener copias actualizadas en los nodos secundarios, es posible promover uno de ellos como primario y reanudar el servicio sin pérdida de datos significativa.

Distribución de carga de trabajo

Los Replica Sets también permiten distribuir la carga de trabajo entre los nodos secundarios, lo que mejora el rendimiento y la capacidad de respuesta del sistema en general. Los nodos secundarios pueden manejar solicitudes de lectura, aliviando la carga del nodo primario y mejorando la escalabilidad horizontal.

5. Despliegue local

Cuando se trata de desplegar una base de datos NoSQL como MongoDB, existen diferentes opciones de implementación. Una de ellas es realizar el despliegue en un entorno local, donde configuramos y gestionamos nosotros mismos los servidores y los nodos del sistema. En este caso, el despliegue se realiza en una única máquina física y lógica, por lo que toda la información está físicamente en el mismo lugar pero lógicamente distribuida debido a la utilización de un Replica Set como la arquitectura de replicación de MongoDB.

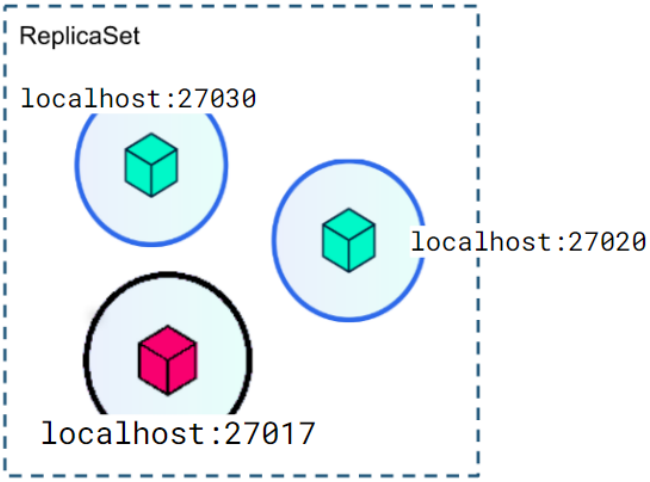

Figura 5: *Esquema de la replicación local*

5.1. La implementación

La arquitectura de Replica Set elegida para el ambiente local es la de un replica set con tres nodos, cada uno de ellos con igual ponderamiento a la hora de realizar votaciones. Uno de los nodos, el principal, tiene facultades de escritura y lectura aunque, en la práctica, todas las operaciones de lectura (como aquella que realiza Power BI) se hacen en los nodos secundarios.

Adicionalmente, se enlazó la fuente de datos con Power BI para realizar analítica de datos. Esto se logró con la tecnología ODBC y el tablero de control resultante sigue el mismo diseño de trabajos anteriores [3].

5.2. Manual técnico para la implementación

La creación y configuración del Replica Set de tres nodos se encuentra en el *Anexo I - Setup Local*. La configuración de la conexión con un software de BI usando OBDC está en el *Anexo II - Conexión BI Local*. Las versiones de los programas y herramientas utilizadas están en el *Anexo IV - Versiones*.

5.3. Ventajas

Control total del entorno

Al realizar un despliegue local, se tiene un control total sobre el entorno del Replica Set de MongoDB. Es posible configurar y ajustar los nodos según las necesidades específicas, lo que brinda un nivel de personalización y adaptabilidad mayor en comparación con otras opciones de implementación. Esto es particularmente útil cuando se quieren realizar pruebas a pequeña escala sobre arquitecturas para trabajar un conjunto de datos o para modelar esquemas completos. La capacidad de controlar todos los elementos del sistema sin incurrir en costos adicionales da mucha libertad al desarrollador.

Mayor privacidad y seguridad de los datos

Al mantener el Replica Set de MongoDB en un ambiente local, se puede tener un mayor control sobre la privacidad y seguridad de los datos. Esto es especialmente relevante si trabajas con datos sensibles o regulados que requieren un nivel adicional de protección.

En un ambiente corporativo, por ejemplo, esto significaría poder hacer pruebas e implementaciones sin restricciones y sin necesidad de trabajar con herramientas de criptografía ni seguridad de redes. Esta alternativa en donde ninguna de las conexiones llega ni siquiera a la red LAN presenta un nivel mínimo de vulnerabilidades de seguridad.

Menor dependencia de servicios externos

Al evitar la dependencia de servicios externos, como proveedores de nube o infraestructura compartida, se reduce el riesgo de interrupciones del servicio y se minimizan los posibles impactos causados por fallas en la red o en los proveedores.

5.4. Desventajas

Complejidad en la configuración inicial

Configurar y mantener un Replica Set de MongoDB en un ambiente local puede requerir un conocimiento técnico más profundo. La configuración inicial y administración de los nodos y servidores requiere atención completa y es imperativo conocer todas las características de las herramientas utilizadas en el flujo de datos. Para modelos grandes o de más alta complejidad, esto puede llegar a ser u desafío que no vale la pena tomar.

Limitaciones de escalabilidad

A medida que los volúmenes de datos aumentan y las necesidades de rendimiento crecen, es posible que se deba invertir en hardware adicional y configuraciones más avanzadas para mantener el rendimiento y la capacidad adecuados.

Responsabilidad total de la administración y el mantenimiento

Al gestionar un Replica Set de MongoDB en un ambiente local, se asume la responsabilidad total de la administración y el mantenimiento del sistema. Esto incluye tareas como la monitorización, la gestión de copias de seguridad y la resolución de problemas, lo cual puede requerir un tiempo y recursos considerables. Adicionalmente, en caso de que el soporte de la arquitectura de datos esté a cargo de un equipo, es irreal pensar que todas ellas van a poder intervenir de manera adecuada usando solo un equipo, por lo que se debe tener una infraestructura robusta

local y, muchas veces, es más económico y viable optar por alternativas como el HaaS o alguna opción de cloud computing.

6. Despliegue en AWS

AWS Lightsail es un servicio de alojamiento en la nube que facilita el despliegue y la administración de aplicaciones y bases de datos. Al utilizar AWS Lightsail, es posible implementar un Replica Set de MongoDB distribuido en diferentes regiones geográficas, lo que brinda beneficios específicos en términos de disponibilidad y rendimiento.

6.1. La implementación

Para este despliegue en particular, dos de los nodos se encuentran en una región y el nodo restante en otra. Esto simula de una manera más realista las condiciones de un entorno de producción al asegurar que las réplicas de los datos están separadas tanto física como geográficamente de una manera significativa. Uno de los nodos, el principal, tiene facultades de escritura y lectura aunque, en la práctica, todas las operaciones de lectura se hacen en los nodos secundarios.

Figura 6: *Instancias (Nodos) de AWS Lightsail*

6.2. Manual técnico para la implementación

La creación y configuración del Replica Set de tres nodos se encuentra en el *Anexo III - Setup AWS*. Las versiones de los programas y herramientas utilizadas están en el *Anexo IV - Versiones*.

6.3. Ventajas

Mayor disponibilidad y resiliencia

Al tener nodos distribuidos en diferentes regiones de AWS, se mejora la disponibilidad de los datos. En caso de una interrupción en una región, los nodos ubicados en otras regiones pueden continuar operando y brindar acceso a los datos. Esto ayuda a reducir el impacto de las fallas y garantiza una mayor resiliencia del sistema.

Reducción de la latencia

Al ubicar nodos en diferentes regiones geográficas, es posible reducir la latencia para los usuarios ubicados en diferentes partes del mundo. Los nodos más cercanos a los usuarios pueden manejar las solicitudes de manera más eficiente, lo que mejora la experiencia del usuario final y el rendimiento general del sistema. Aunque esta implementación no se hizo, es posible (y recomendable) configurar el ruteo para que siempre se apunte al nodo más cercano al usuario final o, en su defecto, al nodo con menor latencia y balance.

Cumplimiento normativo y redundancia geográfica

Al utilizar nodos en diferentes regiones, se pueden cumplir requisitos normativos específicos que exigen redundancia geográfica de los datos. Esto es especialmente relevante en casos en los que se deben cumplir regulaciones relacionadas con la privacidad y la protección de datos en diferentes ubicaciones geográficas.

6.4. Desventajas

Complejidad en la configuración y administración

El despliegue en AWS Lightsail con nodos en diferentes regiones puede ser más complejo

debido a la configuración de la replicación, la sincronización de datos y la administración de la conectividad entre los nodos. A un nivel técnico, la conexión no se hace simplemente a través de la dirección IP de loopback para conectarse con puertos de la misma máquina sino que se debe apuntar a nombres de dominio o IPs públicas de las otras instancias. Esto, además de aumentar la complejidad, también aumenta la vunerabilidad del sistema al estar ahora expuesto a ataques a la red.

Costos adicionales

La distribución de nodos en diferentes regiones puede generar costos adicionales debido a los gastos de transferencia de datos y el mantenimiento de los nodos en cada región. Incluso, muchas empresas prestadoras del servicio de cloud computing cobran un valor adicional por este tipo de despliegues en varias regiones.

Mayor complejidad en la resolución de problemas

La resolución de problemas puede ser más compleja al tener nodos distribuidos en diferentes regiones, lo que requiere un plan de contingencia sólido y consideración de los desafíos adicionales. También, el hecho de que la mayoría de estas herramientas cuenten solamente con un acceso a través de la interfaz de consola dificulta aún más las cosas, pues muchas herramientas cuya instalación y mantenimiento son triviales cuando se cuenta con una GUI, son difíciles de configurar a través de una terminal.

7. Reporting y BI

El proceso de negocio que se identificó como fundamental en la base de datos corresponde a las calificaciones obtenidas en cada asignatura por los estudiantes. Se conservaron las métricas de trabajos pasados para demostrar la factibilidad de utilizar modelos NoSQL en estos procesos; es posible llegar a resultados similares usando métodos diferentes:

7.1. Reporte general

Evolución de en el tiempo de las calificaciones obtenidas en las diferentes unidades de formación de la institución educativa y por lo atento aproximarse a la calidad de los procesos educativos:

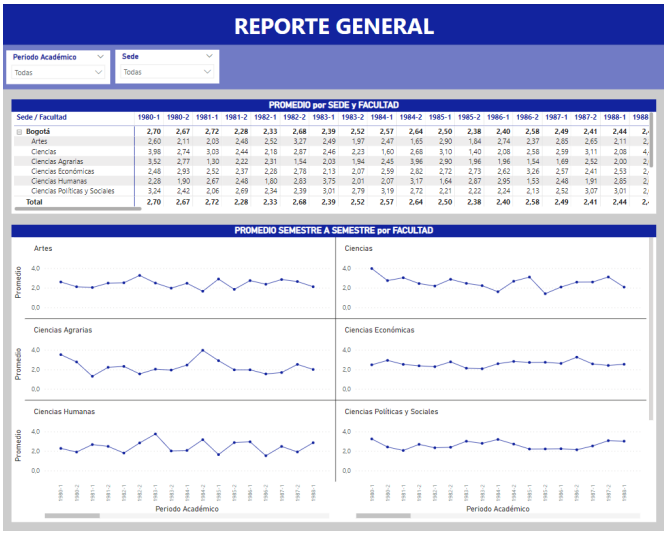

Figura 7: *Reporte General*

7.2. Reporte por estudiante

La evaluación individual de cada estudiante y la revisión de sus resultados académicos es fundamental no solo para la institución sino también para el mismo estudiante. Se muestra información por semestre, total y con métricas personalizadas como el PA y el PAPA:

Figura 8: *Reporte por Estudiante*

7.3. Mejores promedios

Varios estímulos están basados en este ranking académico. Los estudiantes y las asignaturas se organizan con base en los resultados de los cálculos de agregación:

Figura 9: *Mejores Promedios*

8. Conclusiones

Ttanto el despliegue local como el despliegue en AWS Lightsail ofrecen diferentes consideraciones a tener en cuenta al implementar un Replica Set de MongoDB. La elección entre estas dos alternativas dependerá de los requisitos y las necesidades específicas de cada proyecto.

El despliegue local brinda un mayor control y personalización sobre la configuración de un Replica Set. Además, ofrece una mayor privacidad y seguridad de los datos al mantenerlos en un entorno local. Sin embargo, la configuración inicial puede ser compleja y requiere un conocimiento técnico más profundo. Además, la escalabilidad puede ser limitada y puede haber una mayor responsabilidad en términos de administración y mantenimiento del sistema.

Por otro lado, el despliegue en AWS Lightsail con nodos en diferentes regiones proporciona mayor disponibilidad y resiliencia. En caso de una interrupción en una región, los nodos ubicados en otras regiones pueden seguir funcionando, mejorando la continuidad del servicio. Además, la distribución geográfica de los nodos ayuda a reducir la latencia y cumplir con requisitos normativos que exigen redundancia geográfica de los datos. Sin embargo, la configuración y administración presentan dificultades diferentes a las del despliegue local, como es el caso de la administración de redes y la necesidad de utilizar técnicas de seguridad y criptografía; y puede haber costos adicionales asociados a la transferencia de datos y el mantenimiento de los nodos en cada región. Además, la resolución de problemas puede volverse más compleja al no tener total acceso a las máquinas que hacen las veces de servidores o instancias.

9. Trabajo Futuro

A pesar de haber realizado un despliegue exitoso del Replica Set de MongoDB en un entorno local y en AWS, existen varias oportunidades para mejorar y ampliar aún más este proyecto. A continuación, se describen algunas áreas en las que se podría enfocar el trabajo futuro:

Conexión de AWS con software de BI

Una mejora interesante sería conectar el despliegue de AWS con un software de Business Intelligence (BI) como Power BI. Esto permitiría aprovechar las capacidades de visualización y generación de informes avanzados ofrecidos por el software de BI. Sin embargo, esta integración requeriría un desarrollo adicional para realizar las tareas de Extracción, Transformación y Carga (ETL) de los datos desde el Replica Set hacia el

software de BI.

Integración con un warehouse

Otra opción para ampliar el proyecto sería considerar la conexión del Replica Set con un warehouse, como Amazon Redshift. Los warehouses son plataformas diseñadas para el almacenamiento y análisis de grandes volúmenes de datos. Al integrar el Replica Set con un warehouse, se podría obtener un almacenamiento optimizado y estructurado de los datos, lo que facilitaría su análisis y consulta eficiente.

Agregar nodos árbitro y escalar el Replica Set

Conforme el volumen de datos y la carga de trabajo aumenten, puede resultar beneficioso agregar nodos árbitro al Replica Set. Los nodos árbitro actúan como votantes neutrales en la elección del nodo primario y contribuyen a mejorar la disponibilidad y la resiliencia del sistema. Además, es posible escalar horizontalmente el modelo del Replica Set mediante la adición de más nodos, lo que permite un mayor rendimiento y capacidad de procesamiento.

Explorar una alternativa de replica con Sharding

Una alternativa interesante a considerar es el uso de Sharding en MongoDB. Sharding es una técnica que permite distribuir los datos en diferentes servidores, llamados shards, lo que permite manejar conjuntos de datos mucho más grandes y mejorar la capacidad de escalabilidad. Sería válido explorar la viabilidad de implementar Sharding en lugar del Replica Set actual, evaluando los beneficios y desafíos que esta alternativa puede ofrecer en términos de rendimiento, disponibilidad y administración de los datos.

Anexos

Para profundizar más en lo tratado en este artículo, se ha creado un repositorio que contiene, entre otras cosas, la data utilizada en los modelos aquí presentados y algunos manuales técnicos y similares.

Los anexos que se presentan a continuación están en dicho repositorio:

- **Anexo I: Setup Local** – Manual técnico del despliegue en un ambiente local. Instalación de herramientas, setup del cluster y los nodos, etc.
- **Anexo II: Conexión BI Local** – Manual técnico del empalme del despliegue local con Power BI. Instalación y configuración de MongoDB BI Connector, ejecución de *mongosql* e import de datos desde Power BI.
- **Anexo III: Setup AWS** – Manual técnico del despliegue en AWS Lighsail. Setup de instancias, ajustes de networking y configuración del cluster.
- **Anexo IV: Versiones** – Información acerca de las versiones de los programas y herramientas utilizados.

Referencias

[1] M. Kausar y M. Nasar, "SQL Versus NoSQL Databases to Assess Their Appropriateness for Big Data Application," *Annalen der Physik*, vol. 314, n.º 4, págs. 1098-1108, 2021. DOI: http://dx.doi.org/10.2174/2213275912666191028111632.

[2] A. K. Zaki, "NoSQL DATABASES: NEW MILLENNIUM DATABASE FOR BIG DATA, BIG USERS, CLOUD COMPUTING AND ITS SECURITY CHALLENGES," *International Journal of Research in Engineering and Technology*, págs. 403-409, 2014.

[3] J. E. Alarcón, "Implementación de modelo de datos de Sistema Académico usando MySQL," *unpublished*, 2023.

[4] J. Pokorny, "NoSQL databases: a step to database scalability in web environment," *International Journal of Web Information Systems*, vol. 9, págs. 69-82, 2013. DOI: https://doi.org/10.1108/17440081311316398.

[5] S. Ramzan, I. S. Bajwa, B. Ramzan y W. Anwar, "Intelligent Data Engineering for Migration to NoSQL Based Secure Environments," *IEEE Access*, vol. 7, págs. 69 042-69 057, 2019. DOI: https://doi.org/10.1109/ACCESS.2019.2916912.

[6] J. J. Camargo-Vega, J. F. Camargo-Ortega y L. Joyanes-Aguilar, "Conociendo Big Data," *Revista Facultad De Ingeniería, UPTC*, vol. 7, págs. 69 042-69 057, 2014. DOI: https://doi.org/10.19053/01211129.3159.

[7] MongoDB, *MongoDB: Replication*. dirección: https://www.mongodb.com/docs/manual/replication/.