Análisis de claves de fragmentado basadas en rango para MongoDB

Miranda, Juan José
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
juan.miranda@fing.edu.uy

Calcagno, Martín
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
martin.calcagno@fing.edu.uy

***Resumen*—MongoDB es una de las bases de datos NoSQL bastante popular en la actualidad. Es orientada a documentos, de código abierto, y cuenta con un esquema flexible. Para aumentar el rendimiento, puede escalar fácilmente tanto vertical como horizontalmente. Para el escalado horizontal, MongoDB utiliza la técnica de fragmentación automática para dividir y distribuir los datos en varias máquinas. Sin embargo, al utilizar esta técnica, un administrador de base de datos debe elegir una clave de fragmentación para que MongoDB divida su colección. Seleccionar una clave adecuada podría mejorar el rendimiento y la capacidad de una base de datos. Por el contrario, una clave inadecuada disminuye el rendimiento y en algunos casos graves, puede provocar la interrupción del sistema. Por lo tanto, es importante elegir una clave adecuada. MongoDB tiene una sugerencia general sobre las propiedades de su clave de partición ideal. Por ejemplo, una buena clave de fragmentación debe tener un alto grado de aleatoriedad en cuanto a la escritura y debe contener una localidad alta para la lectura de consultas de rango. Para comprender el impacto de estas propiedades en una clave de fragmentación, este trabajo analiza, evalúa y compara dichas propiedades a analizar con las obtenidas por [2] siguiendo los mismos experimentos realizados con el fin de recrear el ambiente de pruebas y resultados. Para realizar los experimentos, se cuenta con un proceso que descarga documentos desde la API de Twitter [4] con el fin de obtener un conjunto de datos relativamente grande para la ejecución de los experimentos. Finalmente se discute y compara los resultados obtenidos en cuanto a los rendimientos de escritura y lectura para los distintos experimentos sobre la clava de fragmentación, con los obtenidos en [2]. Dichos resultados son similares pero no idénticos, por lo cual se discuten posibles alteraciones en los experimentos que pueden llegar a arrojar dichos datos.**

I. Introducción

Esta investigación se centra en el trabajo realizado por Pakorn Kookarinrat y Yaowadee Temtanapat, del departamento de tecnología de la Universidad de Thammasat en Tailandia [2]. En este, se realizan experimentos sobre diferentes claves de fragmentación. Este trabajo se utilizará como guía, intentando replicar de manera semejante los experimentos realizados. Dicho lo anterior, se introducen los conceptos que se analizarán a lo largo de esta investigación.

Una base de datos NoSQL se considera una nueva generación de sistemas de bases de datos que se centra en la escalabilidad y el rendimiento. MongoDB es uno de los sistemas de bases de datos NoSQL más utilizados en la actualidad. Para admitir un gran conjunto de datos con operaciones de alto rendimiento, MongoDB utiliza la fragmentación (sharding), un enfoque de escalabilidad horizontal. La fragmentación divide una colección de datos grandes y la distribuye a varios servidores. Para fragmentar una colección, un administrador de base de datos debe elegir una clave de fragmentación, una propiedad elegida para dividir uniformemente los datos particionados. Seleccionar una clave de fragmentación es fundamental para su rendimiento. Una clave adecuada podría mejorar el rendimiento, mientras que una clave inadecuada no solo puede degradarlo, sino que también podría provocar una interrupción del sistema a nivel general. Dado que una clave de fragmentación se usa para dividir particiones, si los datos tienen una clave de fragmentación con un valor similar, MongoDB no podrá dividir ni distribuir los datos. La inserción de datos terminaría en un servidor específico o en un pequeño conjunto de servidores. Este servidor soportaría la mayoría de las consultas del sistema hasta que no pueda seguir funcionando y se produzca una falla general del sistema.

Hay dos tipos de clave de fragmentación que un administrador podría seleccionar, una clave basada en rangos y una clave basada en una función de hash. Una clave basada en rangos divide los datos según un intervalo de valores. Una clave basada en rangos podría obtener un buen rendimiento de lectura, si solo se ejecuta un pequeño grupo de fragmentos para recuperar datos. Por el contrario, si una consulta debe leerse de forma independiente en muchos fragmentos y luego los resultados deben combinarse, el rendimiento se degradará notoriamente. Una clave basada en una función de hash calcula un valor hash de un campo seleccionado en una colección para que sea su clave.

Una clave basada en una función de hash podría obtener un buen rendimiento de escritura, si pudiera permitir consultas de escritura equitativas en varios fragmentos. La elección de utilizar qué tipo de clave de fragmentación depende principalmente de las aplicaciones a desarrollar. Sin embargo, una clave basada en una función de hash se admite únicamente en un solo campo y puede ser adecuada solo para algunas colecciones. Por ejemplo, en una aplicación que selecciona

un campo secuencial para que sea su clave de fragmentación, el hash puede mejorar el rendimiento de las inserciones. Dado que una clave de base de rango generalmente puede aplicarse a una variedad de sistemas, este trabajo se centra solamente en un tipo de clave, las cuales son las basadas en un rango especifico de valores.

Una buena clave de fragmentación debe tener tres propiedades principales, uniformidad, aleatoriedad, y localidad. En primer lugar, la uniformidad de la clave de fragmentación permite que el sistema distribuya datos de manera uniforme a varios servidores y facilite la división del contenido entre fragmentos. En segundo lugar, la aleatoriedad del valor de la clave podría extender fácilmente las operaciones de escritura a múltiples máquinas. Por último, una buena clave de fragmentación debe proporcionar la localidad para la consulta de lectura para evitar operaciones de lectura en varias máquinas.

Para comprender los efectos de estas tres propiedades en las claves de fragmentos para el rendimiento de MongoDB, se diseñan experimentos para mostrar su impacto, para luego analizar y evaluar sus resultados. Dichos experimentos están basados en los realizados por [2].

Para realizar los experimentos se obtuvieron datos de Twitter, descargados desde su API [4]. Estos son obtenidos en formato JSON, generados por usuarios en tiempo real. Es decir a medida que los usuarios publican sus tweets, se descargan, y se almacenan en una base de datos MongoDB. Posteriormente son filtrados, y a partir de ellos se generan los conjuntos de pruebas como se especifica en la sección III.

II. Antecedentes

II-A. MongoDB

MongoDB es una base de datos NoSQL orientada a documentos. Utiliza un modelo flexible para proporcionar un esquema dinámico, es decir puede ser modificado al momento de almacenar nuevos documentos y a si mismo no es restricción que todos sus documentos tengan los mismos campos. Los datos se almacenan en formato BSON, el cual es una serialización codificada en binario de documentos similares a JSON. La estructura de MongoDB es diferente a la de los sistemas de bases de datos relacionales tradicionales. Dado que su estructura principal es una base de documentos, una colección podría ser comparable a una tabla, mientras que un documento podría ser equivalente a un registro en una tabla.

Los documentos constan de pares clave, valor. Las claves son cadenas de caracteres y los valores pueden ser números, cadenas de caracteres, arreglos, u objetos. Un objeto es en sí mismo un conjunto de pares clave, valor. Por lo tanto, su estructura permite un anidamiento ilimitado. Una consulta puede ser resuelta enviando a la base de datos un formato similar a JSON para comparar con la colección en el sistema.

MongoDB usa fragmentación para aumentar su rendimientos en un escalado horizontal. Para habilitar la fragmentación, se le debe indicar a MongoDB qué datos y colecciones debe fragmentar, y qué atributo o atributos del documento se utilizarán como clave de fragmentación [3]. Los atributos que sirvan como clave de fragmentación deben estar presentes en todos los documentos, estos serán indexados para su uso posterior. Hay tres componentes principales en un clúster de fragmentación para MongoDB, los cuales son: los fragmentos (shards), configuración (config), y enrutador (router) (véase Fig. 1).

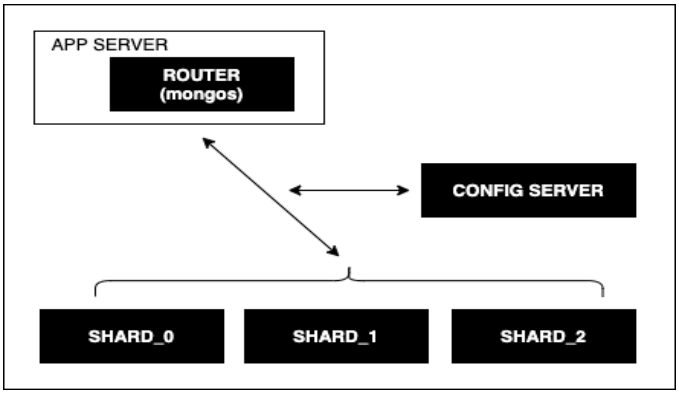

Figura 1: Estructura del servidor de fragmentos

- **Servidor de fragmentos:**
  Un fragmento es un componente que almacena parte de los datos. Por lo general, se construye a partir de la base de datos MongoDB y podría usar un conjunto de réplicas para aumentar la disponibilidad. Un cliente, generalmente, puede realizar una solicitud de lectura o escritura en un fragmento a través del enrutador. Un cliente también puede realizar una solicitud directamente a un fragmento, sin embargo, el cliente solo recibirá los datos que se encuentran en ese fragmento.

- **Servidor de configuración:** El servidor de configuración es el componente que almacena la configuración del clúster de fragmentos. Está construido a partir de MongoDB. Este componente de configuración recibirá solicitudes de configuración de lectura y escritura de un enrutador.

- **Servidor de enrutamiento:** Un enrutador es un punto de conexión de clústers de fragmentos. Un enrutador se construye a partir de un programa llamado MongoS. Un cliente administrará y solicitará datos del clúster de fragmentos a través de un enrutador. Cuando el enrutador recibe una solicitud, lee su configuración (información sobre las ubicaciones de los fragmentos que contienen los datos solicitados). De esta manera, el enrutador realiza una solicitud a esos fragmentos. Después de obtener los resultados de los fragmentos, el

enrutador los combina y envía al cliente.

III. Diseño de experimentos

Los experimentos realizados se centran en los efectos de las opciones de clave de fragmentación y el impacto en el rendimiento de la fragmentación sobre las operaciones de lectura y escritura. Como se mencionó anteriormente, las tres principales propiedades sugeridas de una clave de fragmentación deben de ser la uniformidad, la aleatoriedad, y la localidad. Dichas propiedades podrían afectar al clúster fragmentado en las operaciones de lectura y escritura. En esta sección, se explica como se generan los datos experimentales para evaluar dichas propiedades y el conjunto de consultas de escritura y lectura sobre los datos para medir los efectos de los rendimientos de escritura y lectura. La configuración del sistema y otras herramientas adicionales se explican en la sección III-E. La implementación de los experimentos tanto para escritura como lectura se pueden consultar en la plataforma GitLab [1], donde se encuentran los procesos de carga y los experimentos de lectura realizadas. También se puede encontrar un archivo que contiene los tweets utilizados para las pruebas, como también el proceso encargado de descargar los tweets.

III-A. Organización de los datos de prueba y operaciones de consulta

- **Experimentos para rendimiento de escritura:**
  Dos propiedades principales que podrían afectar el rendimiento de la escritura son la uniformidad y la aleatoriedad. El escalado horizontal depende de qué tan bien se puedan distribuir los datos de manera uniforme, la uniformidad permite un almacenamiento uniforme, mientras que la aleatoriedad ayuda a distribuir temporalmente la operación a varios fragmentos. Por lo tanto, las claves de fragmentos para las pruebas se dividen en 3 grupos: clave de fragmentación aleatoria, ordenada, y combinada.
  Una clave de fragmentación aleatoria tiene como objetivo realizar una prueba de uniformidad. Una clave de fragmentación en esta categoría se divide en 4 rangos, 1 a 1.000.000, 1 a 666.666, 1 a 333.333, y 1 a 3. Los primeros 3 rangos se utilizan para probar el efecto del límite de rango en el rendimiento de escritura, mientras que el último rango se utiliza para ver el efecto de los "jumbo chunks"[1] en el rendimiento, y que sucede si un fragmento agota sus recursos.
  Se elige una clave de fragmentación ordenada para probar el efecto en el rendimiento de escritura de fragmentación cuando los datos entrantes llegan en orden. Dado que una clave de fragmentación ordenada se acumularía en un fragmento único antes de que el chunk sea lo suficientemente grande como para dividirse. Por lo tanto, podría cargar un fragmento en todo momento.

  Una clave combinada es una combinación de claves ordenada y aleatoria. Este tipo de clave se elige en [2] ya que es una forma fácil que un administrador tiende a usar para ganar uniformidad y aleatoriedad para una clave de fragmentación. Se comparará el rendimiento de esta clave con los dos grupos anteriormente mencionados.

- **Experimentos para el rendimiento de lectura**
  MongoDB podría obtener un buen rendimiento de lectura si se ejecuta un pequeño grupo de fragmentos para recuperar los datos. Leer, en general, implica distribuir subconsultas a varios fragmentos y luego esperar a que estos resultados se combinen. Por lo tanto, entre más fragmentos involucradas, se esperaría un rendimiento de lectura menos eficiente. Para probar el rendimiento de lectura de fragmentación en los datos configurados en la sección anterior, se proporcionan los siguientes criterios de lectura, objetivo de clave única, objetivo de clave de rango, y por último objetivo en base a un índice.
  Una consulta de objetivo de clave única ("single key target query") es una consulta que solicita un único valor de la clave de fragmentación. Por lo tanto, se espera que la consulta se envíe a un solo fragmento.
  Una consulta de objetivo clave de rango (range key target query") implica un rango de documentos. Este tipo de consulta se divide en dos subtipos, un rango de datos pequeño y uno grande. Para una consulta de rango pequeño, se involucra aproximadamente un 1 % del rango en un chunk. Por lo tanto, es probable que cubra solo un fragmento. Una consulta de rango grande cubre 2 o más chunks. Es posible que solicite dos o tres fragmentos para responder dicha consulta.
  Finalmente, una consulta de objetivo de índice (ïndex target query") solicita datos que están indexados pero no por una clave de fragmentación. Al busca en el índice, posiblemente se realicen solicitudes a todos los fragmentos para obtener una respuesta.

III-B. Colección de pruebas

Para realizar los experimentos se obtienen datos desde los servicios de la red social Twitter. Se desarrolla un script que obtiene tweets que las personas registradas en dicha red social van publicando, y se almacenan en una base MongoDB local. Los datos obtenidos desde Twitter superan el millón de documentos y un total de almacenamiento de aproximadamente 1GB.

Al analizar los datos, se detectan notificaciones de eliminación de tweets, notificaciones de re-tweets, y notificaciones de alta de tweets. Por esto mismo se filtran los datos y se seleccionan únicamente las altas de tweets. Así mismo se

[1] chunks tan grandes que deben ser divididos, a menos que ellos representen un único valor de clave de fragmentación, por lo que no pueden ser divididos y superan el tamaño máximo de un chunk

filtran sus campos para que los datos a almacenar no ocupen un espacio innecesario en el espacio de almacenamiento.

Para realizar una prueba consistente, se generaron datos de acuerdo con las condiciones descritas en la Sección III.A. El esquema de los datos de prueba se muestra en la Tabla 1.

Esquema de las colecciones para las pruebas			
Nombre	**Ordenado**	**Aleatorio**	**Cantidad**
sort	X		1,000,000
m_sort	X		333,333
random		X	1,000,000
s_random_combined	X	X	333,333
s_random		X	333,333
m_random		X	333,333
b_random		X	666,666
random_combined	X	X	1,000,000

Figura 2: Tabla 1

III-C. Configuración del sistema

Las pruebas se realizan sobre una maquina que auspicia de servidor la cual utiliza un sistema operativo Windows 10, con recursos de 250GB de disco duro en estado solido (SSD), 16GB de memoria RAM, y 2GHz de velocidad.

Se configuran 6 maquinas virtuales en Docker, estas son: un servidor de MongoDB, un servidor de configuración, un enrutador, y 3 fragmentos (shards), en los que se configuran las claves de fragmentación.

Para la configuración de las claves de fragmentación se utiliza el siguiente esquema:

Esquema clave de fragmentos	
Clave	Fragmento
0 <= clave < 333.333	Shard 0
333.333 <= clave < 666.666	Shard 1
666.666 <= clave < 1.000.000	Shard 2

Figura 3: Organización de la clave de fragmentos.

III-D. Monitoreo de Docker y lectura de rendimientos

Para monitorear los recursos consumidos por las maquinas virtuales se utiliza la herramienta Docker-Stats que permite ver el consumo de recursos de la maquina. Para administrar los recursos utilizados por Docker, se utiliza en el caso de Windows un archivo de configuración llamado .wslconfig, en el que se puede asignar a Docker un límite máximo de memoria RAM y un límite máximo de cantidad de CPUs a utilizar.

Para cronometrar los tiempos de ejecución de los experimentos, tanto para lectura como escritura, se toma el tiempo actual de antes de comenzar las pruebas hasta el momentos de finalización de las mismas, obteniendo así los tiempos totales de ejecución.

III-E. Proceso de experimentos

Cada experimento se codifica como un script en Python3[2], utilizando la librería pymongo[3].

Para los experimentos de escritura, se lee el archivo JSON con los tweets y se va insertando uno a uno en la base de datos, agregándole su clave de fragmentación según corresponda (de forma ordenada: insertando documentos con claves que vayan incrementando en uno, y de forma aleatoria: insertando documentos con claves aleatorias, sin repetir, dentro del rango correspondiente).

Estos experimentos se deben realizar con distintos niveles de recursos de CPU y memoria RAM, en los experimentos a replicar se utiliza 30 %, 40 %, y 60 % de límites de estos recursos, esto es: 30 % de CPU y 4.8GB de RAM, 40 % de CPU y 6.4GB de RAM, y 60 % de CPU y 9.6GB de RAM, ya que la maquina utilizada en los experimentos a replicar es de 16GB de memoria RAM. Al estar utilizando Windows y no poder limitar los recursos utilizados por Docker porcentualmente, se le asignaran 25 %, 50 %, y 75 % de los recursos (ya que la asignación de CPUs es por cantidades enteras, 1 de 4, 2 de 4, y 3 de 4 respectivamente). Luego de correr los experimentos con estos recursos, se notó que 4GB de memoria RAM no eran suficientes para poder correr los scripts, por lo que se debió aumentar a 5GB. También se notó que el rendimiento de las pruebas utilizando el 50 % y 75 % de los recursos fue casi el mismo, en ambas pruebas casi nunca se llegaba al límite. Por esto mismo se resolvió disminuir la cantidad de memoria RAM utilizada en el segundo nivel de recursos, con el fin de realizar experimentos más variados llegando a la siguiente configuración de recursos a utilizar:

Configuraciones de límites de recursos		
Configuración	#CPUs	Memoria RAM
config1 (25 %)	1	5GB
config2 (50 %)	2	7GB
config3 (75 %)	3	12GB

Figura 4: Configuraciones de límites de recursos para experimentos de lectura

Para los experimentos de lectura, se realizan consultas a documentos según corresponda. Las consultas de tipo clave única realizan una consulta a un único documento (find_one), mientras que las restantes, consultas de rango e indexadas realizan consultas a varios documentos (find).

Para realizar las consultas de clave indexada, se crea un índice a los hashtags [4] de los tweets, los cuales son consultados en estos experimentos.

[2]https://docs.python.org/3/

[3]https://pymongo.readthedocs.io/en/stable/

[4]Etiqueta que se emplea en el terreno de la informática para aludir a una cadena de caracteres que se inicia con el símbolo #, conocido como numeral.

Para realizar los experimentos de s-random, se insertan 333.333 documentos con claves de fragmentación que varían de 1 a 3 para observar el rendimiento del chunk. Las restantes consultas se realizan sobre la clave de fragmentación.

Pasos para la ejecución de los experimentos se siguen los siguientes pasos:

1. Configurar las claves de fragmentación en los servidores.
2. Para cada uno de los experimentos:
   a) Asignar los recursos correspondientes al experimento.
   b) Ejecutar el script de inserción de datos.
   c) Recopilar el tiempo de ejecución y el uso de recursos utilizando la herramienta de monitoreo.
   d) Ejecutar los scripts de lectura de datos.
   e) Recopilar el tiempo de ejecución, y el uso de recursos de cada prueba de lectura de la herramienta de monitoreo, es decir, el número de fragmentos involucrados por consulta.

IV. Resultados de los experimentos

Se realizan los experimentos de la forma descrita en la sección III A. llegando a los siguientes resultados:

IV-A. Resultados de rendimiento de escritura

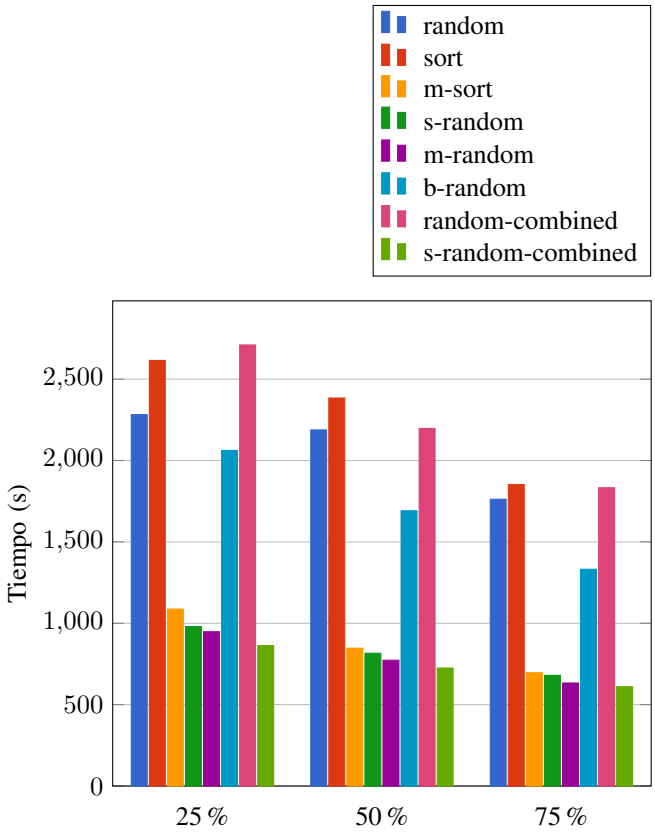

Figura 5: Tiempos de ejecución de experimentos de escritura.

En la figura 5, se puede ver que las claves de fragmentación aleatorias, ofrecen un rendimiento un poco mayor a las demás en cuanto a escritura, incluso en el caso de que se pueda generar un jumbo chunk. El límite de rango no se ve afectado significativamente en el rendimiento de escritura. Para una clave de fragmentación ordenada, el rendimiento de escritura podría degradarse en cualquier rango de datos, ya que los datos son insertados solamente en un fragmento en lugar de cualquier fragmento disponible para ser procesado en paralelo.

La clave aleatoria obtiene un resultado un poco mejor que las claves ordenada y combinada, obteniendo resultados similares al trabajo a replicar. Esta diferencia no es tan notoria en cuanto a los tiempos de ejecución, esto se puede deber a que los equipos utilizados, los límites de recursos y los datos son distintos. Otra posible razón es una diferencia en eficiencia en cuanto a la ejecución de los scripts de los experimentos.

Todo esto siempre y cuando las comparaciones se hagan con respecto a la cantidad de documentos insertados, esto es, comparando los experimentos que insertan un millón de documentos por un lado (sort, random, y random-combined), los que insertan 666.666 documentos (b-random) por otro, y los que insertan 333.333 (m-sort, m-random) por otro. Quizás se puede suponer que los resultados mostrados en el trabajo a replicar están escalados según la cantidad de documentos a insertar, por eso las gráficas con son tan similares.

Por último, la limitación de recursos puede degradar el rendimiento de la fragmentación. Sin embargo, disminuye de manera uniforme según los recursos asignados. Por lo tanto, se puede suponer que en cualquier nivel de recursos, el patrón de rendimiento que afecta a la clave de fragmentación no cambiará.

IV-B. Resultados de los experimentos de lectura

En las figuras 6 y 7 se pueden ver los resultados obtenidos para los experimentos de lectura ejecutadas, y la cantidad de fragmentos involucrados en las consultas. De la figura 6 se puede concluir que los tiempos de lectura se degradan a medida que aumenta el rango de clave buscado.

La ejecución de una consulta de rango más amplio obtiene un peor rendimiento de lectura que una de rango más pequeño. Sin embargo, las consultas clave de indexada tienen un rendimiento deficiente en todo tipo de consultas, ya que necesitan usar todos los fragmentos para procesar el resultado. Un punto interesante que se observa en este tipo de consultas es que la cantidad de documentos obtenidos por dichas consultas es mayor a la de las restantes consultas. Por lo que es de suponer que va a tomar más tiempo en ejecutarse. Sin embargo, al realizar estas consultas para obtener una misma cantidad de datos que en las demás consultas, los tiempos no varían de forma significativa, por lo que se podría afirmar lo anteriormente mencionado.

En conclusión, una búsqueda por índice o una búsqueda de

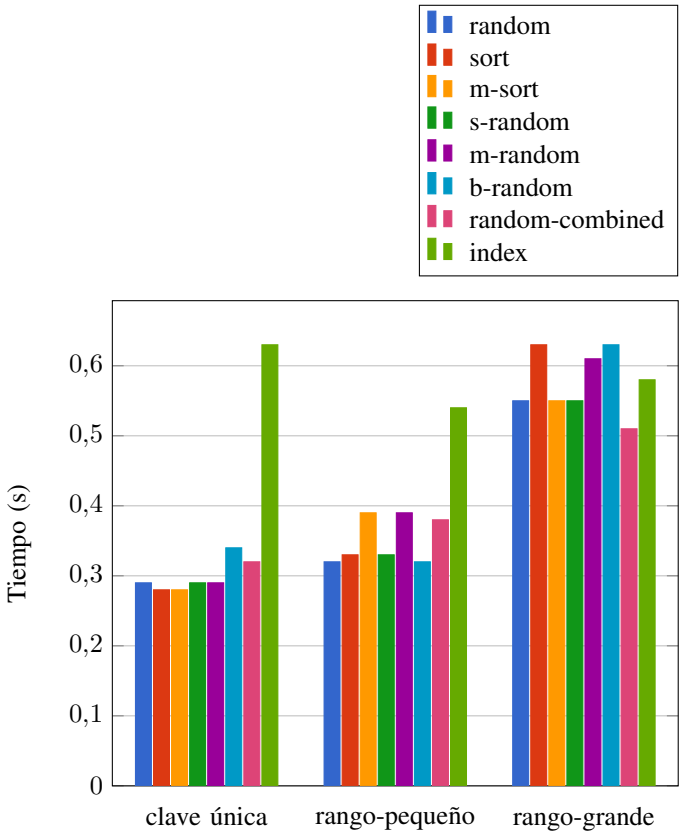

Figura 6: Tiempos de ejecución de experimentos de lectura.

	Key		Small Range		Big Range	
	Number of Shards	Execution Time	Number of Shards	Execution Time	Number of Shards	Execution Time
sort	1	0.28	1	0.33	2	0.63
medium sort	1	0.28	1	0.39	2	0.55
random	1	0.29	2	0.32	3	0.55
small random	1	0.29	1	0.33	3	0.55
medium random	1	0.29	2	0.39	2	0.61
big random	1	0.34	1	0.32	3	0.63
index	3	0.63	3	0.54	3	0.58
combine random+sort	1	0.32	1	0.38	2	0.51

Figura 7: Tiempos de ejecución y uso de fragmentos

rango amplio podrían degradar el rendimiento del sistema debido a la cantidad de máquinas involucradas que entregan los resultados. Por otro lado, las consultas de clave única o el rango pequeño de búsqueda tienen mejores rendimientos que las consultas de clave de índice o las consultas de clave de rango amplio.

La cantidad de fragmentos involucrados tiene un efecto significativo en el rendimiento de lectura. Cuantos más fragmentos se involucren en la lectura, peor será el rendimiento. Por lo tanto, la localidad tiene un impacto en el rendimiento de las lecturas.

V. Conclusiones y trabajos futuro

MongoDB se convierte en una opción popular para el uso de bases de datos NoSQL en la actualidad porque es un software de código abierto que puede soportar big data[5] sin un esquema restringido como un sistema de base de datos relacional tradicional. Sin embargo, para obtener un beneficio de su escalamiento horizontal, se debe elegir una clave de fragmentación adecuada, lo cual no es una tarea fácil.

Para comprender los efectos de una clave de fragmentación en el rendimiento de escalado de MongoDB, este documento evalúa y compara los resultados con experimentos realizadas en otra investigación para determinar el efecto de seleccionar la clave de fragmentación en base a rangos. Se simula el entorno y se mide el rendimiento de lectura / escritura en una variedad de experimentos. En comparación con los resultados obtenidos por [2] se puede concluir que los mismos son similares en cuanto a los experimentos de lectura y escritura (las diferencias entre tiempos de ejecución pueden deberse a que los equipos utilizados son distintos, el hecho de tener distintos conjuntos de datos, distintas limitaciones de recursos, y quizás distintos niveles de eficiencia de los scripts implementados). En ambos trabajos, se puede concluir que una clave de fragmentación con aleatoriedad y buena localidad puede ofrecer un buen rendimiento tanto de escritura, como de lectura.

Además, en caso de que una clave de fragmentación tenga valores casi ordenados, la combinación de un pequeño rango de valores aleatorios podría proporcionar un rendimiento aceptable tanto para lectura como para escritura.

Así mismo queda pendiente el T-Test para los experimentos realizados. El T-Test es un tipo de estadística deductiva. Y se utiliza para determinar si hay una diferencia significativa entre las medias de dos grupos, en este caso sería entre dos experimentos de escritura o lectura.

Referencias

[1] Juan Miranda / Martín Calcagno. Repositorio del proyecto. https://gitlab.fing.edu.uy/juan.miranda/bdnr, 2021. Online; accessed 21 July 2021.

[2] P. Kookarinrat and Y. Temtanapat. Analysis of range-based key properties for sharded cluster of mongodb. In *2015 2nd International Conference on Information Science and Security (ICISS)*, pages 1–4, 2015.

[3] MongoDB. Sharding en MongoDB. https://docs.mongodb.com/manual/sharding/, 2021. Online; accessed 24 June 2021.

[4] Twitter. Twitter API. https://developer.twitter.com/en/docs/twitter-api, 2021. Online; accessed 19 June 2021.

[5] https://www.mongodb.com/es/big-data-explained