Plantilla para la elaboración del Informe de Proyecto de Fin de Curso

Ana Carolina Espino
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
ana.carolina.espino@fing.edu.uy

Diego Rosolino
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
diego.rosolino@fing.edu.uy

***Resumen*—Con el avance de los sistemas que manejan grandes volúmenes de datos, se hace necesario contar con bases de datos que puedan escalar su performance. En particular MongoDB provee un mecanismo para conseguir esto a través de la fragmentación de colecciones en distintos servidores. Este documento presenta el detalle de las pruebas y resultados obtenidos en la elección y evaluación distintas claves de fragmentación en MongoDB para la fragmentación de colecciones en más de un servidor, aplicado a un caso real.**

I. Introducción

Hoy en día son cada vez más los sistemas que utilizan diferentes tipos de base de datos NoSQL. MongoDB es una base de datos NoSQL distribuida, basada en documentos y de uso general que ha sido diseñada para desarrolladores de aplicaciones modernas y para la era de la nube. Para admitir un gran conjunto de datos con operaciones de alto rendimiento, se puede escalar de dos formas: horizontal y vertical.

El escalado vertical implica aumentar la capacidad de un solo servidor, como usar una CPU más potente, agregar más RAM o aumentar la cantidad de espacio de almacenamiento. Las limitaciones en la tecnología disponible pueden impedir que una sola máquina sea lo suficientemente potente para una carga de trabajo determinada. Además, los proveedores basados en la nube tienen límites máximos basados en las configuraciones de hardware disponibles. Como resultado, existe un máximo práctico para el escalado vertical.

El escalado horizontal implica dividir el conjunto de datos del sistema y la carga en varios servidores, agregando servidores adicionales para aumentar la capacidad según sea necesario. Si bien la velocidad o capacidad general de una sola máquina puede no ser alta, cada máquina maneja un subconjunto de la carga de trabajo general, lo que potencialmente proporciona una mayor eficiencia que un solo servidor de alta velocidad y alta capacidad. Expandir la capacidad de la implementación solo requiere agregar servidores adicionales según sea necesario, lo que puede tener un costo general más bajo que el hardware de alta gama para una sola máquina. La compensación es una mayor complejidad en la infraestructura y mantenimiento de la implementación.

MongoDB admite escalado horizontal mediante fragmentación. La fragmentación divide una colección de datos grandes y los distribuye a varios servidores. Para fragmentar una colección, un administrador de base de datos debe decidir la estrategia de distribución del contenido de la colección. MongoDB usa la clave de fragmentación para distribuir los documentos de la colección entre fragmentos asignando un rango de valores a un fragmento.

El objetivo de nuestro trabajo es estudiar los efectos de las elecciones de las claves de fragmentación sobre un ejemplo de una colección en la realidad.

El resto del documento está organizado en las siguientes secciones. En la Sección II se detalla el marco teórico utilizado, se presentan los conceptos de fragmentación en mongoDB y su estructura. La Sección III trata de trabajos relacionados sobre el tema. En la Sección IV se describe el objetivo a realizar en el caso de estudio. Luego, en la Sección V se detallan los experimentos realizados, la infraestructura utilizada y los problemas obtenidos. Por último en la Sección VI se concluye el trabajo y propone posibles trabajos a futuro.

II. Marco teórico

MongoDB[1] es una base de datos no relacional orientada a documentos y de código abierto. Asimismo almacena datos en documentos con formato flexible similares a JSON, por lo que los campos pueden variar entre documentos y la estructura de datos puede cambiarse con el tiempo. Los documentos se agrupan en colecciones y una base de datos en MongoDB puede estar compuesta por varias colecciones.

Las bases de datos MongoDB, proveen una herramienta de fragmentación[2] que permite el escalado horizontal de la base y facilita la distribución de una base ya sea que se va a generar de cero como de una base ya existente.

Para realizar la fragmentación, MongoDB utiliza las denominadas claves de fragmentación.

Las claves de fragmentación se basan en atributos dentro de cada documento. Los valores de esos campos decidirán en qué fragmento residirá el documento, de acuerdo con los rangos de fragmentación y la cantidad de fragmentos. Estos datos se almacenan y mantienen en el conjunto de réplicas del servidor de configuración. Seleccionar una buena clave de fragmentación es fundamental para el rendimiento de la base de datos distribuida. Una buena clave podría mejorar el rendimiento, mientras que una mala clave puede degradarlo.

Hay dos tipos de clave de fragmentación, una clave de hash y una de rango. Una fragmentación de clave de hash calcula

[1]https://www.mongodb.com/es/what-is-mongodb
[2]https://docs.mongodb.com/v4.0/sharding/

el valor de hash de un campo seleccionado en una colección para así poder dividir los documentos de la colección en los diferentes fragmentos. Una fragmentación con clave de hash proporciona una distribución de datos más uniforme en todo el clúster. Una fragmentación con clave basada en rangos implica dividir los datos en rangos contiguos determinados por los valores de la clave de la fragmentación. En este modelo, es probable que los documentos con valores de clave cercanos estén en el mismo fragmento.

De la sugerencia general en el documento de MongoDB[3], la elección de una buena clave de fragmentación debe tener en cuenta tres propiedades principales de los atributos candidatos a ser claves, que son: cardinalidad, frecuencia y monotonicidad. La cardinalidad de una clave de fragmentación determina el número máximo de fragmentos que se pueden crear. Esto puede reducir o eliminar la eficacia de la escala horizontal en el clúster. La frecuencia de la clave de fragmentación representa la frecuencia con la que se produce un valor determinado en los datos. Si la mayoría de los documentos contienen sólo un subconjunto de esos valores, los fragmentos que almacenan esos documentos se convierten en un cuello de botella dentro del clúster. La monotonicidad de una clave de fragmentación determina la distribución en los fragmentos, si el valor de la clave aumenta o disminuye de manera monótona tiene más probabilidades de distribuir las inserciones en un solo fragmento dentro del clúster.

Partiendo de esas propiedades, se listan características a nivel de la ejecución de operaciones que debería cumplir cualquier elección de una clave que se considere adecuada para la fragmentación [Zola(2019)] :

- Todas las inserciones, actualizaciones, y eliminaciones deben distribuirse uniformemente en todos los fragmentos que componen el clúster.
- Todas las consultas se deben distribuir uniformemente en todos los fragmentos que componen el clúster.
- Todas las operaciones se dirigirán únicamente a los fragmentos de interés: una actualización o eliminación no se debe mandar a un fragmento que no contiene los datos a ser modificados.
- De la misma manera, una consulta no debe enviarse a un fragmento que no contiene ninguno de los datos que se consultan.

El conseguir una clave que cumpla estas características evita los siguientes problemas:

- Baja escalabilidad en escritura: si la carga en escritura no se distribuye a todos los fragmentos, se genera una sobre carga de uno o alguno de estos fragmentos, no utilizando el potencial de la fragmentación para la mejora en el rendimiento.
- Baja escalabilidad en la lectura: De la misma manera que en el punto anterior, el no distribuir las consultas entre los distintos fragmentos, si bien no tiene el nivel de impacto que tiene la escritura, tampoco deja aprovechar las ventajas de la fragmentación al máximo.

La infraestructura del clúster de mongoDB consta de los siguientes elementos:

- **Mongos:** actúan como enrutadores de consultas, brindando una interfaz entre las aplicaciones cliente y el clúster fragmentado.
- **Servidores de configuración:** almacenan metadatos y ajustes de configuración para el clúster
- **Fragmentos:** cada fragmento contiene un subconjunto de los datos fragmentados. Cada frgamento se puede implementar como un conjunto de réplicas.

III. Trabajos relacionados

En el año 2015 se publicó el articulo “Analysis of range-based key properties for sharded cluster of MongoDB“ [Kookarinrat and Temtanapat(2015)] donde se presenta el análisis de distintos tipos de claves para la fragmentación basada en rangos de una colección en MongoDB que fue creada para este propósito, con atributos que cumplían ciertas propiedades que se deseaban explorar para medir que tan adecuado era elegir dichos atributos como clave de acuerdo a que ventajas o desventajas podían presentar. El trabajo realizado tenía por objetivo plantear algunos lineamientos para que los administradores de bases de datos encuentren más sencillo el problema de elegir una clave de fragmentación de acuerdo a su realidad.

IV. Planteo del caso de estudio

A partir de lo planteado en el artículo [Kookarinrat and Temtanapat(2015)] y de la documentación de MongoDB sobre la fragmentación de colecciones, nos planteamos la necesidad de aplicar los conceptos presentados a un caso de la realidad. El objetivo es realizar la comparación entre los conceptos teóricos y el comportamiento real de la base de datos distribuida ante las distintas elecciones de claves de fragmentación desde el punto de vista de un administrador de base de datos que se enfrenta al problema de realizar la distribución.

Para llevar a cabo las pruebas se utilizaron datos de las publicaciones de Airbnb que están disponibles como datos de prueba, con licencia pública en Inside Airbnb.[4] La colección utilizada fue generada a partir de los archivos “listings” de varias de las ciudades que se encuentran en el repositorio de datos. La única modificación realizada sobre los datos fue agregar la columna “city” que incluye el nombre de la ciudad dada, dentro de la colección.

V. Experimentación

V-A. Naturaleza de los datos y operaciones

Airbnb es un sistema en el que se conectan anfitriones que son quienes publican sus hospedajes, la colección que los representa es denominada listings, y huéspedes que se quieran alojar en ellos.

Si bien no se encontró información respecto a cuales operaciones son las que se realizan con más frecuencia sobre la

[3]https://docs.mongodb.com/v4.0/core/sharding-shard-key/#choosing-a-shard-key

[4]http://insideairbnb.com/get-the-data.html

colección a utilizar, por estadísticas publicadas sobre Airbnb [Bustamante(2021)] , en particular sobre la relación entre las cantidades de los distintos tipos de usuarios, es notable que la proporción de huéspedes es muy superior a la de los anfitriones. Por esta razón en las pruebas realizadas, nos enfocamos en el estudio de como afectan las claves a las consultas sobre los listings, haciendo énfasis en los problemas sobre las lecturas a la base de datos.

Los experimentos constaron de realizar cinco consultas con dos claves de fragmentación diferentes. Para la elección de las claves se utilizaron las propiedades de cardinalidad, frecuencia y monotonicidad definidas en la Sección II que se tienen que tener en cuenta como mínimo al elegir una clave. Usando estas propiedades se creó una clave que las cumplieran y otra que no cumpliera alguna de ellas, con el fin de evaluar la importancia de estas propiedades. Para la clave correcta se utilizaron dos campos (ciudad y precio). La clave cumple con cardinalidad porque no contiene una cantidad acotada de valores posibles, siendo la combinación de ciudad y precio algo no acotado. La propiedad de frecuencia también la cumple porque no hay tampoco un par de valores que tengan una alta frecuencia en el conjunto de datos contra otros que tengan baja frecuencia de una manera destacable. Por último, se cumple la monoticidad porque para cada documento el par de valores de la clave no cumple que siempre sea creciente o decreciente. Además de estas propiedades principales, también se tuvo en cuenta que sean relevantes para las consultas a realizar. Si bien no obtuvimos datos de las consultas más utilizadas existen factores por los que los listings aparecen antes o después en la lista que se muestra a los usuarios y estos factores están directamente conectados con el tipo de consultas que hacen los usuarios y sus intereses [Griffiths(2020)]. Entre esos atributos destacables están la locación y el precio y es por esta razón que para la clave se eligieron los campos de ciudad y precio. Para definir la clave que no siga las propiedades, se utiliza el campo identificador del documento, el cual no cumple con la propiedad de monoticidad porque todo nuevo valor es siempre creciente. A su vez, en muchos casos un administrador de base de datos inexperto puede tomar la decisión de utilizar este campo ya que siempre está presente en todos los documentos.

Las consultas utilizadas para realizar los experimentos fueron:

- Una ciudad (Londres) y cantidad de personas (4)
- Una ciudad (Londres)y cantidad de personas (4) con un rango de precios (mayor a 30 y menor a 60)
- Una ciudad (Londres) y cantidad de personas (4) con un rango de precios (mayor a 39 y menor a 60) y si el alojamiento se encuentra disponible
- Un rango de longitud (mayor a 51.47 y menor a 51.57) y latitud (mayor a -0.3 y menor a -0.1) junto con la cantidad de personas (4)

V-B. Configuración del ambiente

El ambiente se generó utilizando contenedores de docker, en el que se encontraban 11 maquinas virtuales sobre una computadora con 12GB de memoria, CPU Intel i7 y un disco de 225GB. Las maquinas virtuales constaban de un servidor de configuración, un enrutador y tres fragmentos con tres replicas cada uno. La versión de mongoDB utilizada fue la 4.4, que posteriormente fue actualizada a la 5.0. Los datos utilizados, con el fin de realizar los experimentos, fueron un subconjunto de listings con un total de 1.3GB.

Para realizar las medición se utilizó un script[5] realizado en el lenguaje Python 3.8.5[6] el cual realiza las diferentes consultas y obtiene tres metricas: tiempo, memoria y cpu utilizada. Finalizando las pruebas nos dimos cuenta que las mediciones de cpu y memoria se tomaban de forma incorrecta y por lo tanto, no reflejaban el consumo verdadero en los fragmentos.

V-C. Resultados obtenidos

A continuación se detallan los resultados de los experimentos, donde se realizaron tres mediciones y se tomo un promedio con el fin de aproximar a un resultado correcto.

Consultas	Clave ciudad y precio	Clave identificador
1	0,22	0,53
2	0,12	0,51
3	0,11	0,53
4	0,21	0,55
5	0,56	0,58

Cuadro I: Medición del tiempo de ejecución en segundos de las consultas

Se puede observar que todas las consultas realizadas con la clave ciudad-precio se ejecutan en un menor tiempo que las que utilizan la clave de identificador, esto se explica porque en varias consultas se utilizan la clave o parte de la clave ciudad-precio como filtro y esto hace que el mongos pueda identificar en que fragmentos se encuentran estos datos para poder dirigir específicamente las consultas a estos fragmentos.

V-D. Problemas encontrados

Los principales problemas que nos enfrentamos estuvieron presente al momento de realizar los experimentos, teniendo muchas dificultades para cargar los datos de prueba. El proceso de carga fue el que más tiempo se necesitó, ya que demoraba demasiado tiempo en cargarse la base y además consumía demasiados recursos en la computadora donde se estaban ejecutando los contenedores docker lo que llevaba a que la computadora se trancara. Asimismo, es necesario recargar la base cuando se quería cambiar la clave de fragmentación ya que en versiones anteriores a mongo 5.0 no es posible hacerlo [7] y estando en esa versión se necesita de muchos recursos disponibles los cuales no contabamos.

Otro problema encontrado fue al momento de verificar las mediciones, las mediciones de cpu y memoria no estaban reflejando correctamente el consumo que se hacia de ese

[5]https://gitlab.fing.edu.uy/diego.rosolino/bdnr/-/blob/master/measurer.py
[6]https://www.python.org/downloads/release/python-385/
[7]https://docs.mongodb.com/manual/faq/sharding/
#can-i-select-a-different-shard-key-after-sharding-a-collection-

recurso en cada fragmento, por lo tanto optamos por no presentarlos en la Sección V-C.

VI. Conclusiones y trabajo futuro

A partir de los resultados de las pruebas se observa que el tiempo de ejecución de las consultas con la clave que sigue las recomendaciones es mejor que a una clave que no. Esta mejora en tiempos de ejecución es aproximadamente el doble para una misma consulta, debemos aclarar que el tiempo de estas consultas para ambas claves fue menos de un segundo pero en un conjunto de datos pequeños, con un aumento en los datos llevaría a una mayor diferencia entre los tiempos de ejecución. Por último, podemos observar que una clave también tiene que tener en cuenta las consultas que se van a realizar sobre la base de datos, las primeras cuatro consultas contenían un atributo de la clave y en comparación con la quinta tuvieron un menor tiempo de ejecución.

En cuanto a trabajo futuro, al trabajar con este tipo de datos nos topamos con la necesidad de extender el estudio de como se comporta la fragmentación y la elección de las claves si se trabaja con más de una colección. En el sitio donde descargamos la colección en la que nos enfocamos, ya existen otras colecciones relacionadas a la misma, como por ejemplo el calendario con precios por día de las distintas publicaciones. Sería interesante ver si la elección individual de las claves en cada una de las colecciones, afecta las consultas que involucran a más de una de ellas o si la elección de esas claves tendría que ser pensada desde el punto de vista de la combinación de las colecciones.

Por otro lado dados los problemas con los que nos encontramos al realizar las pruebas con docker, consideramos interesante que se puedan realizar las pruebas sobre una infraestructura con un clúster real pudiendo así trabajar con una carga de datos mayor y quizás también aumentando la cantidad de fragmentos.

Referencias

[Bustamante(2021)] Jaleesa Bustamante. Airbnb Statistics. https://ipropertymanagement.com/research/airbnb-statistics, 2021. Sitio web, Accedido: Julio 2021.

[Griffiths(2020)] Keiran Griffiths. Top Airbnb Ranking Factors. https://www.liftylife.ca/airbnb-ranking-factors, 2020. Sitio web, Accedido: Julio 2021.

[Kookarinrat and Temtanapat(2015)] Pakorn Kookarinrat and Yaowadee Temtanapat. Analysis of range-based key properties for sharded cluster of mongodb. *2015 2nd International Conference on Information Science and Security (ICISS)*, pages 1–4, 2015. URL http://ieeexplore.ieee.org/document/7370983/.

[Zola(2019)] William Zola. On Selecting a Shard Key for MongoDB. https://www.mongodb.com/blog/post/on-selecting-a-shard-key-for-mongodb, 2019. Sitio web, Accedido: Julio 2021.