Desempeño de sharding de una colección real en MongoDB

Grupo 6

Contexto: Sharding en MongoDB

- Sharding es un método provisto por MongoDB para dar escalabilidad horizontal en la base de datos documental.
- La base de datos se encuentra distribuida en varios servidores donde cada uno de estos es un “shard”.
- Existe una clave “shard key” a partir de la cual se genera la partición y recuperación de los datos desde los distintos servidores.
- En cada Shard se generan los llamados “Chunks”, cada chunk tiene asignado un rango de posibles valores de clave.

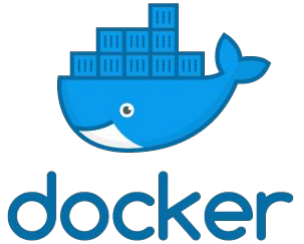
Propiedades de las claves

- **Cardinalidad**: cantidad de valores que puede tomar el atributo que será la clave. Determina el número máximo de chunks que se pueden crear.
- **Frecuencia**: qué tan frecuente es un valor de clave, en particular en comparación con los otros posibles valores de la misma.
- **Monotonicidad**: refiere a valores de clave que serán siempre crecientes o decrecientes a la hora de insertar nuevos documentos a la colección

Infraestructura utilizada

Contenedores de Docker con:

- Un router/Mongos
- Un servidor de configuración
- Tres shards con tres nodos cada uno



Se utilizó mongoDB 4.4 / 5.0

Datos utilizados

Colección “listings” de airbnb

Entre los datos contenidos en la colección se encuentran:

- Ubicación (Ciudad, latitud, longitud)
- Datos de la persona que publica (host)
- Disponibilidad
- Precio
- Cantidad de personas que se pueden hospedar
- Otros

*http://insideairbnb.com/get-the-data.html.

Consultas

1. Una ciudad (Londres) y cantidad de personas (4)
```
db.listings.find({ 'city': "London", 'accommodates': 4 })
```
2. Una ciudad (Londres) y cantidad de personas (4) con un rango de precios (mayor a 30 y menor a 60)
```
db.listings.find({ 'city': "London", 'accommodates': 4, 'price': { '$gte': 30, '$lte': 60 } })
```
3. Una ciudad (Londres) y cantidad de personas (4) con un rango de precios (mayor a 30 y menor a 60) y si el alojamiento se encuentra disponible
```
db.listings.find({ 'city': "London", 'accommodates': 4, 'price': { '$gte': 30, '$lte': 60 }, 'has_availability': "t" })
```
4. Una ciudad (Londres) y cantidad de personas (4) con un tipo de alojamiento (apartamento)
```
db.listings.find({ 'city': "London", 'accommodates': 4, 'property_type': "Entire apartment" })
```
5. 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)
```
db.listings.find({ 'latitude': { '$gte': 51.47000 , '$lte': 51.57000 }, 'longitude': { '$gte': -0.30000 ,'$lte': -0.10000},
'accommodates': 4 })
```

Claves

Las claves seleccionadas fueron:

- Cumpliendo las recomendaciones
  - ciudad-precio: se utilizan los atributos city y price
- Sin cumplir las recomendaciones
  - id: se utiliza el atributo identificador del documento generado por mongo
    - (No cumple la condición de monotonicidad)

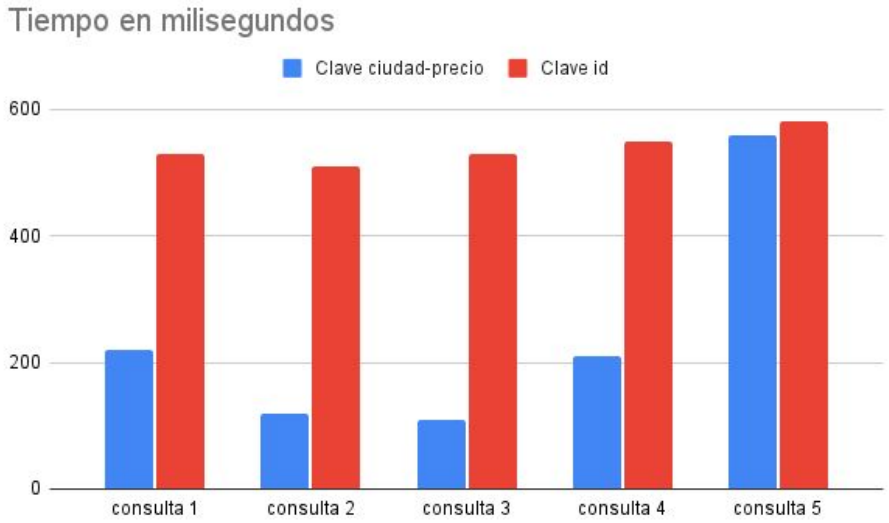
```
db.adminCommand({ shardCollection: "airbnb.listings", key: { "atributo_1": 1, ..., "atributo_N": 1 } })
```

```
db.adminCommand({ reshardCollection: "airbnb.listings", key: { "atributo_1": 1, ..., "atributo_N": 1 }
})
```

Pruebas y resultados

Se ejecutaron un total de 3 veces cada consulta, obteniendo el promedio del tiempo de ejecución para cada una de las claves definidas.

La ejecución se realizó con un script en python.



Conclusiones y Trabajos a futuro

- La clave seleccionada en función de las consultas y las recomendaciones de una buena clave de mongo obtuvo una mejor performance.
- La conclusión no es determinante si se consideran los siguientes puntos:
  - Muy pocas claves evaluadas.
  - El conjunto de datos fue pequeño.
  - Las pruebas no fueron realizadas sobre un cluster real.

Otras pruebas interesantes:

Pruebas de sharding sobre más de una colección del mismo sistema.

Muchas gracias