Comparaci´on BD NoSQL

Nilson Felix Laura Atencio

Andree Velasco Sucapuca

Resumen

Los sistemas de bases de datos NoSQL crecieron con las principales redes sociales, como Google, Amazon, Twitter y Facebook. Estas tenían que enfrentarse a desafíos con el tratamiento de datos que las tradicionales SGBDR no solucionaban. Con el crecimiento de la web en tiempo real existía una necesidad de proporcionar información procesada a partir de grandes volúmenes de datos que tenían unas estructuras horizontales más o menos similares. Estas compañías se dieron cuenta de que el rendimiento y sus propiedades de tiempo real eran más importantes que la coherencia, en la que las bases de datos relacionales tradicionales dedicaban una gran cantidad de tiempo de proceso

Palabras clave: NoSQL,

Abstract

The NoSQL database systems grew with the main social networks, such as Google, Amazon, Twitter and Facebook. They had to face challenges with the data processing that the traditional RDBMS did not solve. With the growth of the web in real time there was a need to provide information processed from large volumes of data that had more or less similar horizontal structures. These companies realized that performance and their real-time properties were more important than consistency, in which traditional relational databases
spent a large amount of processing time.

Keywords: NoSQL,

I. INTRODUCCIO´N

El problema de la escalabilidad de SQL fue reconocido por empresas Web 2.0, con grandes necesidades de datos e infraestructura, como Google, Amazon y Facebook. Ellos solos tuvieron que buscar soluciones propias a este problema, con tecnologías como BigTable, DynamoDB, y Cassandra.

Este interés creciente dio lugar a una serie de sistemas de gestión de base de datos NoSQL (DBMS), con un enfoque en el rendimiento, la fiabilidad y la coherencia. Se reutilizaron y mejoraron varias estructuras de indexación existentes con el propósito de mejorar la búsqueda y el rendimiento de lectura.

En primer lugar, había tipos de bases de datos NoSQL (de origen cerrado), desarrolladas por grandes empresas para satisfacer sus necesidades específicas, como BigTable de Google, que se cree es el primer sistema NoSQL y DynamoDB de Amazon.

El éxito de estos sistemas patentados, inició el desarrollo de varios sistemas de bases de datos de código abierto y de propietarios similiares siendo los más populares Hypertable, Cassandra, MongoDB, DynamoDB, HBase y Redis.

II. MATERIALES Y ME´TODOS

A. Materiales

- Virtualizaci´on activada en el BIOS
- Docker Desktop
- Windows 10 64bit: Pro, Enterprise o Education, con al menos 4GB de RAM.

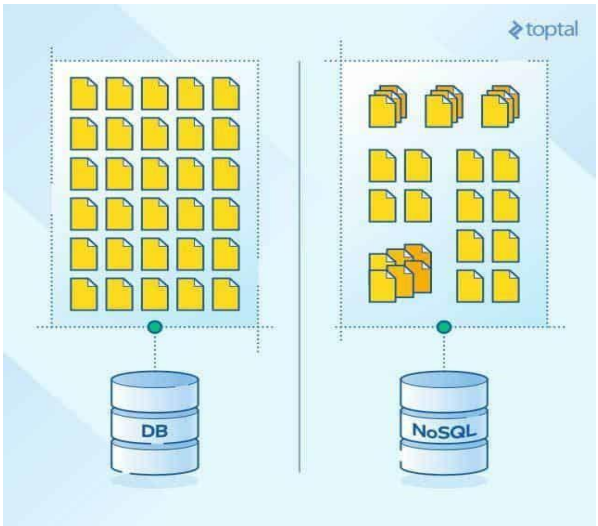
B. M´etodos

- Se utilizo como material art´ıculos y libros relaciona- dos a la base de datos NoSQL y sus tipos, as´ı como p´aginas web.

III. MARCO TEO´RICO

A. Diferencias

Una diferencia clave entre las bases de datos de NoSQL y las bases de datos relacionales tradicionales, es el hecho de que NoSQL es una forma de almacenamiento no estructurado.



Esto significa que NoSQL no tiene una estructura de tabla fija como las que se encuentran en las bases de datos relacionales.

Ventajas y desventajas de las bases de datos NoSQL

Ventajas

Las bases de datos de NoSQL presentan muchas ventajas en comparación con las bases de datos tradicionales.

- A diferencia de las bases de datos relacionales, las bases de datos NoSQL están basadas en key-value pairs
- Algunos tipos de almacén de bases de datos NoSQL incluyen diferentes tipos de almacenes como por ejemplo el almacén de columnas, de documentos, de key value store, de gráficos, de objetos, de XML y otros modos de almacén de datos.
- Algunos tipos de almacén de bases de datos NoSQL incluyen almacenes de columnas, de documentos, de valores de claves, de gráficos, de objetos, de XML y otros modos de almacén de datos.
- Podría decirse que las bases de datos NoSQL de código abierto tienen una implementación rentable. Ya que no requieren las tarifas de licencia y pueden ejecutarse en hardware de precio bajo.
- Cuando trabajamos con bases de datos NoSQL, ya sean de código abierto o tengan un propietario, la expansión es más fácil y más barata que cuando se trabaja con bases de datos relacionales. Esto se debe a que se realiza un escalado horizontal y se distribuye la carga por todos los nodos. En lugar de realizarse una escala vertical, más típica en los sistemas de bases de datos relacionales.

Desventajas

Por supuesto, las bases de datos NoSQL no son perfectas, y no siempre van a ser la elección ideal.

- La mayoría de las bases de datos NoSQL no admiten funciones de fiabilidad, que son soportadas por sistemas de bases de datos relacionales. Estas características de fiabilidad pueden resumirse en: “atomicidad, consistencia, aislamiento y durabilidad.” Esto también significa que las bases de datos NoSQL, que no soportan esas características, ofrecen consistencia para el rendimiento y la escalabilidad.
- Con el fin de apoyar las características de fiabilidad y coherencia, los desarrolladores deben implementar su propio código, lo que agrega más complejidad al sistema.
- Esto podría limitar el número de aplicaciones en las que podemos confiar para realizar transacciones seguras y confiables, como por ejemplo los sistemas bancarios.
- Otras formas de complejidad encontradas en la mayoría de las bases de datos NoSQL, incluyen la incompatibilidad con consultas SQL. Esto significa que se necesita un lenguaje de consulta manual, haciendo los procesos mucho más lentos y complejos.

NoSQL vs. Bases de datos relacionales

Esta tabla ofrece una breve comparación entre las funcionalidades de NoSQL y las bases de datos relacionales:

Feature	NoSQL Databases	Relational Databases
Performance	High	Low
Reliability	Poor	Good
Availability	Good	Good
Consistency	Poor	Good
Data Storage	Optimized for huge data	Medium sized to large
Scalability	High	High (but more expensive)

Cabe señalar que esta tabla muestra una comparación a nivel de la base de datos, no sobre los diversos sistemas de gestión de bases de datos que implementan ambos modelos. Estos sistemas proporcionan *sus propias técnicas patentadas* para superar los problemas y deficiencias encontradas en el sistema, además de intentar mejorar significativamente el rendimiento y la fiabilidad.

Tipos de almacenamiento de datos NoSQL

Key Value Store

En el tipo de almacén Key Value, se utiliza una tabla hash en la que una clave única apunta a un elemento.

Las claves pueden ser organizadas por grupos clave lógicos, requiriendo solamente estas claves para ser únicas dentro de su propio grupo. Esto permite tener claves idénticas en diferentes grupos lógicos. La siguiente tabla muestra un ejemplo de un almacén de valores clave, en el que la clave es el nombre de la ciudad y el valor es la dirección de Ulster University en esa ciudad.

Key	Value
"Belfast"	{"University of Ulster, Belfast campus, York Street, Belfast, BT15 1ED"}
"Coleraine"	{"University of Ulster, Coleraine campus, Cromore Road, Co. Londonderry, BT52 1SA"}

Algunas implementaciones del almacén de valores clave proporcionan mecanismos de almacenamiento en el caché, lo que mejora en gran medida su rendimiento.

Todo lo que se necesita para hacer frente a los elementos almacenados en la base de datos: es la clave. Los datos se almacenan en una forma de una cadena, JSON o BLOB (objeto grande binario).

Uno de los mayores defectos en esta forma de base de datos es la falta de consistencia a nivel de la base de datos. Esto puede ser añadido por los desarrolladores con su propio código, aunque esto suponga más esfuerzo y tiempo.

La base de datos NoSQL más famosa que se construye en un almacén de valores clave Key Value es DynamoDB de Amazon.

Almacén de documentos

Los almacenes de documentos son similares a los almacenes de valores clave, porque no tienen un esquema y se basan en un modelo de valor clave. Ambos carecen de coherencia en el nivel de base de datos, lo que hace posible que las aplicaciones proporcionen más fiabilidad.

Las diferencias más significativas son:

– En el almacén de documentos, los valores (documentos) proporcionan codificación XML, JSON o BSON (JSON codificado binario) para los datos almacenados.

La aplicación de base de datos más popular, que se basa en un almacén de documentos es MongoDB.

Almacenamiento en columnas

Los datos se almacenan en columnas, en lugar de almacenarse en filas, (como se hace en la mayoría de los sistemas de gestión de bases de datos relacionales).

Un almacén de columnas está compuesto por una o más familias de columnas que se agrupan de forma lógica en determinadas columnas en la base de datos. Una clave se utiliza para identificar y señalar a un número de columnas en la base de datos. Cada columna contiene filas de nombres o tuplas, y valores, ordenados y separados por comas.

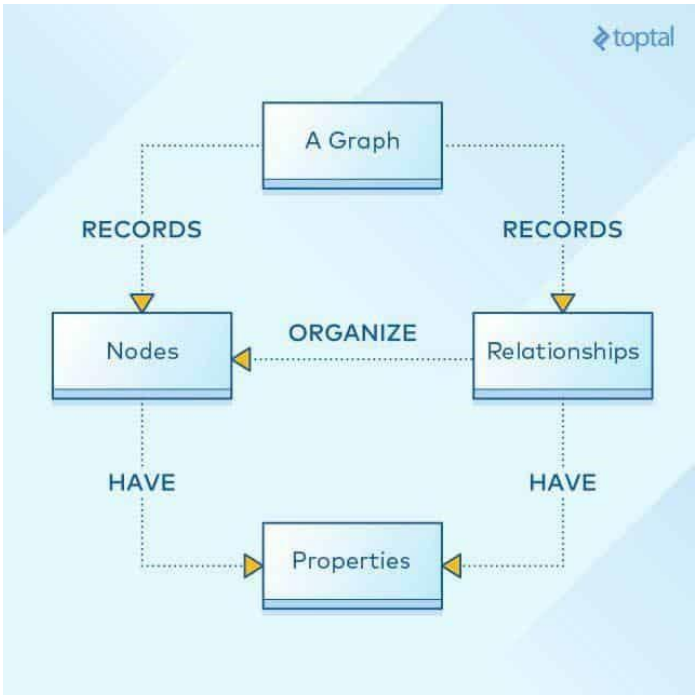
Los almacenes de columnas tienen acceso rápido de lectura y escritura a los datos almacenados. En un almacén de columnas, las filas que corresponden a una sola columna se almacenan como una sola entrada de disco, lo cual facilita el acceso durante las operaciones de lectura y escritura.

Las bases de datos más populares que usan el almacén de columnas incluyen Google BigTable, HBase y Cassandra.

Base gráfica

En una gráfica de una base de datos NoSQL, se utiliza una "estructura de gráfica dirigida" para representar los datos. El gráfico está compuesto por bordes y nodos.

Formalmente, un gráfico, es una representación de un conjunto de objetos, donde algunos pares de objetos están conectados por enlaces. Los objetos



interconectados están representados por abstracciones matemáticas, llamadas vértices, y los enlaces que conectan algunos pares de vértices se llaman bordes. **Un gráfico sería un tipo de representación de datos compuesto por un conjunto de vértices y de bordes que conectan entre sí, mostrando visualmente su relación matemática.**

Esto ilustra la estructura de una base de datos de manera gráfica, donde se usan bordes y nodos para representar y almacenar los datos. Estos nodos están organizados entre sí, y queda representado por los bordes entre los nodos. Tanto los nodos como las relaciones tienen propiedades definidas.

Las bases de datos de gráficos, suelen utilizarse en aplicaciones de redes sociales. Estas permiten a los desarrolladores centrarse más en las relaciones entre los objetos que en los propios objetos. En este contexto, de hecho permiten un entorno escalable y fácil de usar.

Actualmente, InfoGrid y InfiniteGraph son las bases de datos gráficas más populares.

Sistemas de gestión de Bases de datos NoSQL

Por una breve comparación de las bases de datos, la tabla siguiente, proporciona una breve comparación entre los diferentes sistemas de gestión de bases de datos NoSQL.

Key	Value
"Belfast"	{"University of Ulster, Belfast campus, York Street, Belfast, BT15 1ED"}
"Coleraine"	{"University of Ulster, Coleraine campus, Cromore Road, Co. Londonderry, BT52 1SA"}

MongoDB tiene un sistema flexible de almacenamiento de esquemas. Lo que significa que los objetos almacenados no tienen que tener la misma estructura o los mismos campos. MongoDB también tiene algunas características de optimización, que distribuye las colecciones de datos, mejorando el rendimiento y consiguiendo un sistema más equilibrado.

Otros sistemas de base de datos NoSQL, como Apache CouchDB, también se consideran bases de datos de tipo almacén de documentos. Por ello comparten muchas características con MongoDB, a excepción de que es posible acceder a la base de datos usando APIs RESTful.

REST es un estilo arquitectónico que consiste en un conjunto coordinado de restricciones arquitectónicas aplicadas a componentes, conectores y elementos de datos, todo esto dentro de la World Wide Web. Está basado en un protocolo de comunicaciones apilables, cliente-servidor, protocolo cacheable de comunicaciones, (por ejemplo, el protocolo HTTP).

Las aplicaciones RESTful utilizan peticiones HTTP para publicar, leer y eliminar datos.

En cuanto a bases de datos de bases de columnas, Hypertable es una base de datos NoSQL escrita en C ++ y basada en BigTable de Google. Hypertable soporta la distribución de almacenes de datos entre nodos para maximizar la escalabilidad, al igual que MongoDB y CouchDB.

Una de las bases de datos NoSQL más utilizadas es Cassandra, desarrollada por Facebook. Se trata de una base de datos de almacenes de columnas que incluye muchas características dirigidas a la fiabilidad y tolerancia de fallos.

Cassandra

Cassandra es un sistema de gestión de bases de datos desarrollado por Facebook, cuyo objetivo era crear un DBMS sin fallos y que proporcione la máxima disponibilidad.

Cassandra es principalmente una base de datos de almacenes de columnas. Algunos estudios se refieren a Cassandra como un sistema híbrido, inspirado en BigTable de Google, (base de datos de almacén de columnas), y en DynamoDB de Amazon, (base de datos de valor clave).

Esto se consigue proporcionando un sistema de valor clave. Pero las claves de Cassandra apuntan a un conjunto de familias de columnas, dependiendo del sistema de archivos distribuido "BigTable" de Google y de las características de disponibilidad de Dynamo (tabla hash distribuida).

Cassandra está diseñado para almacenar enormes cantidades de datos distribuidos a través de diferentes nodos. Cassandra es un DBMS diseñado para manejar cantidades masivas de datos, repartidos entre muchos servidores, mientras que proporciona un servicio altamente disponible sin un solo punto de fallo, lo cual es esencial para un gran servicio como Facebook.

Las principales características de Cassandra incluyen:

- **No hay ni un solo punto de fallo.** Para que esto se consiga, Cassandra debe funcionar como un racimo de nodos. Eso no significa que los datos de cada clúster sean los mismos, sin embargo si debe serlo el software de gestión. Cuando ocurre un fallo en uno de los nodos, los datos en ese nodo serán inaccesibles. Sin embargo, otros nodos (y datos) seguirán siendo accesibles.
- **Un Hashing distribuido** es un esquema que proporciona la funcionalidad de tabla hash, de manera que la adición o supresión de una ranura no cambia significativamente la asignación de claves a dichas ranuras. Esto proporciona la capacidad de distribuir la carga a los servidores o nodos según su capacidad y, a su vez, minimizar el tiempo de inactividad.
- **Interfaz de cliente relativamente fácil de usar.** Cassandra utiliza Apache Thrift para su interfaz de cliente. Apache Thrift ofrece un cliente RPC en varios idiomas, pero la mayoría de los desarrolladores prefieren alternativas de código abierto construidas sobre Apple Thrift, como Hector.
- **Otras características de disponibilidad.** Una de las características de Cassandra es la replicación de datos. Básicamente, refleja datos a otros nodos del clúster. La replicación puede ser aleatoria o específica para maximizar la protección de datos, colocándola por ejemplo en un nodo en un centro de datos diferente. Otra característica que se encuentra en Cassandra es la política de partición. La directiva de partición, decide en qué nodo se va a colocar la clave. Esto también puede ser aleatorio u ordenado. Al utilizar ambos tipos de políticas de partición, Cassandra puede lograr un equilibrio entre el equilibrio de carga y la optimización del rendimiento de las consultas.
- **Consistencia.** Funciones como la replicación, hacen que la consistencia sea un desafío. Esto se debe al hecho de que todos los nodos deben estar actualizados en cualquier punto en el tiempo con los valores más recientes. Sin embargo, Cassandra intenta mantener un equilibrio entre las acciones de replicación y las acciones de lectura/escritura proporcionando esta personalización al desarrollador.
- **Acciones de lectura / escritura.** El cliente envía una solicitud a un único nodo de Cassandra. El nodo, de acuerdo con la política de replicación, almacena los datos en el clúster. Cada nodo realiza primero el cambio de datos en el registro de confirmación y, a continuación, actualiza la estructura de la tabla con el cambio, ambos realizados de forma sincrónica. La operación de lectura es también muy similar, una petición de lectura se envía a un solo nodo y ese único nodo es el que determina qué nodo contiene los datos, de acuerdo con la política de partición/ ubicación.

MongoDB

MongoDB es una base de datos libre de esquemas, orientada a documentos, escrita en C ++. La base de datos está basada en el almacén de documentos, lo que significa que almacena valores (denominados documentos) en forma de datos codificados.

La elección del formato codificado en MongoDB es JSON. Es muy potente, porque incluso si los datos están anidados dentro de los documentos JSON, seguirá siendo consultable e indexable.

Las subsecciones que siguen, describen algunas de las características clave disponibles en MongoDB.

Shards / Fragmentos

Sharding es la partición y distribución de datos a través de múltiples máquinas (nodos). Un fragmento, es una colección de nodos MongoDB. A diferencia que Cassandra, donde los nodos estaban simétricamente distribuidos. El uso de fragmentos también implica la capacidad de escalar horizontalmente a través de múltiples nodos. En el caso de que haya una aplicación que utilice un único servidor de base de datos, se puede convertir en clúster fragmentado, con muy pocos cambios en el código de la aplicación original, por la forma en que Sharding es ejecutada por MongoDB. Oftware está casi desacoplado de las API públicas.

Lenguaje de consulta Mongo

Como se mencionó anteriormente, MongoDB utiliza una API RESTful. Para recuperar ciertos documentos de una colección db, se crea un documento de consulta que contiene los campos que deben coincidir con los documentos deseados.

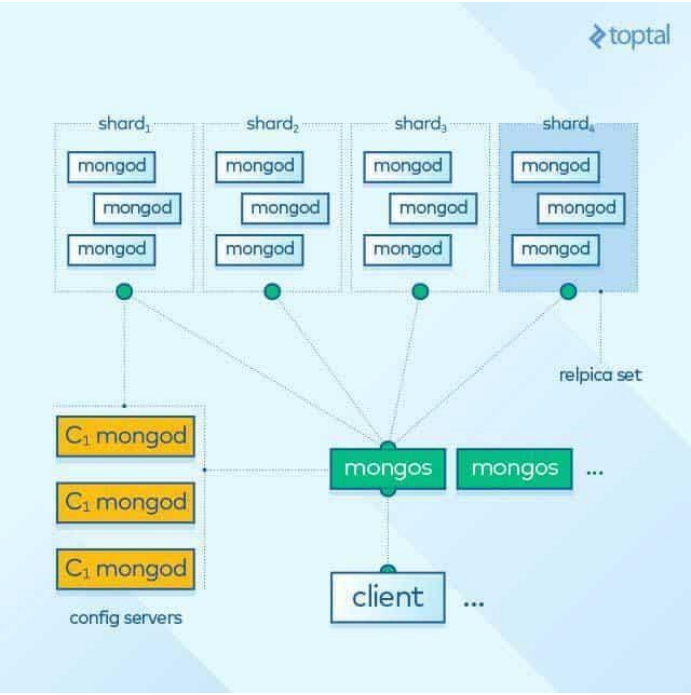
Acciones

En MongoDB, hay un grupo de servidores llamados enrutadores. Cada uno actúa como un servidor para uno o más clientes. Del mismo modo, el clúster contiene un grupo de servidores denominados servidores de configuración. Cada uno contiene una copia de los metadatos que indican qué fragmento contiene qué datos. Las acciones de lectura o escritura se envían desde los clientes a uno de los servidores de enrutador del clúster y son encaminadas automáticamente por ese servidor, a los fragmentos adecuados que contienen los datos con la ayuda de los servidores de configuración.

Un fragmento en MongoDB similar a Cassandra es que ambos tienen un esquema de replicación de datos, que crea un conjunto de réplicas de cada fragmento que contiene exactamente los mismos datos.

Hay dos tipos de esquemas de réplica en MongoDB: Master-Slave replication y Replica-Set replication. Replica-Set proporciona más automatización y mejor manejo para los fallos, mientras que MasterSlave suele requerir la intervención de un administrador. Independientemente del esquema de replicación, en cualquier punto de conjunto de réplicas, sólo un fragmento actúa como fragmento primario. Todos los fragmentos de réplica son fragmentos secundarios. Todas las operaciones de escritura y lectura pasan al fragmento primario y luego se distribuyen de forma uniforme, (si fuera necesario), a los otros fragmentos secundarios del conjunto.

En el gráfico de abajo, vemos la arquitectura de MongoDB explicada anteriormente, mostrando los servidores del enrutador en verde, los servidores de configuración en amarillo y los fragmentos que contienen los nodos MongoDB en azules.



Cabe señalar que el sharding (o compartir los datos entre fragmentos) en MongoDB es completamente automático, lo que reduce la tasa de fallos y hace MongoDB un sistema de gestión de base de datos altamente escalable.

Estructuras de indexación para bases de datos NoSQL

La indexación es el proceso de asociar una clave con la ubicación de un registro de datos correspondiente en un DBMS. Hay muchas estructuras de datos de indización utilizadas en las bases de datos NoSQL. Las siguientes secciones discutirán brevemente algunos de los métodos más comunes; La indexación de los árboles B, la indexación de los árboles T y la indexación de los árboles O2.

Indexación de árboles B

El árbol B es una de las estructuras de índice más comunes en DBMS.

En los árboles B, los nodos internos pueden tener un número variable de nodos secundarios dentro de un rango predefinido.

Una diferencia importante de otras estructuras de árbol, como AVL, es que el árbol B permite que los nodos tengan un número variable de nodos secundarios. Lo que va a significar menos equilibrio de árbol y más espacio perdido.

El B + -Tree es una de las variantes más populares de B-Trees. El B + -Tree es una mejora sobre B-Tree que requiere todas las claves para residir en las hojas.

Indexación de árboles T

La estructura de datos de un árbol T fue diseñada combinando características de AVL-Trees y B-Trees. (AVL-árbol y B-árbol).

Un árbol AVL es un tipo de auto-equilibrio binario de árboles de búsqueda, mientras que un árbol B es más desequilibrado, y cada nodo puede tener un número diferente de hijos.

En un árbol T, la estructura es muy similar a los árboles B y AVL.

Cada nodo almacena más de una tupla {key-value, pointer}. Además, la búsqueda binaria se utiliza en combinación con los nodos de múltiples tuplas para producir un mejor almacenamiento y rendimiento.

Un árbol T tiene tres tipos de nodos: Un T-Node que tiene un hijo derecho e izquierdo, un nodo de hoja sin hijos, y un nodo de media hoja con un solo hijo.

Se cree que los árboles T tienen un mejor rendimiento general que los árboles AVL.

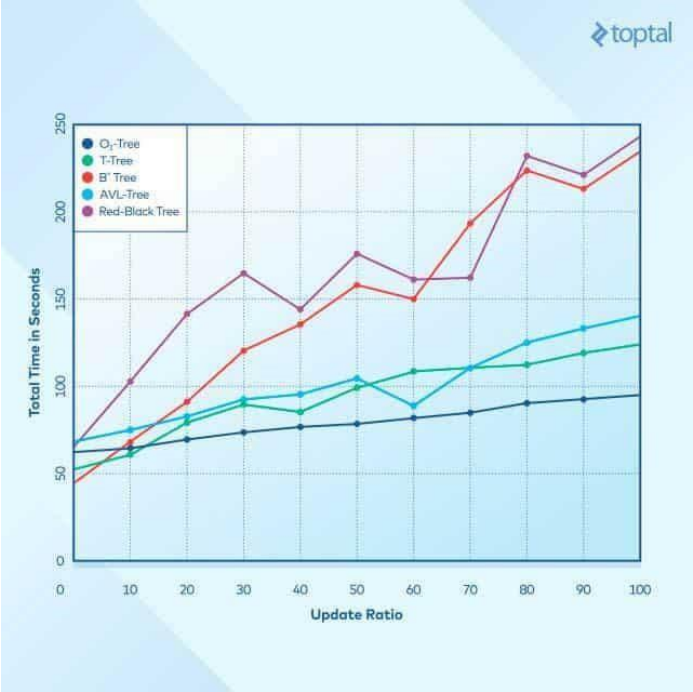
Indexación de árboles O2

El árbol O2 es básicamente una mejora sobre los árboles Rojo-Negro (RedBlack), una forma de un árbol Binary-Search, en el que los nodos hoja contienen el valor {key value, pointer}

El árbol O2, se propuso para mejorar el rendimiento de los actuales métodos de indexación. Un árbol de O2 de orden m ($m \geq 2$), donde m es el grado mínimo del árbol, satisface las siguientes propiedades:

- Cada nodo es rojo o negro. Pero la raíz es siempre negra.
- Si un nodo es rojo, entonces sus dos hijos son negros.
- Para cada nodo interno, todas las rutas simples desde el nodo hasta los nodos-hoja descendientes contienen el mismo número de nodos negros. Cada nodo interno tiene un único valor de clave.
- Los nodos de hoja son bloques que tienen entre $\lceil m / 2 \rceil$ y m pares «keyvalue, record-pointer».
- Si un árbol tiene un único nodo, entonces debe ser una hoja, que es la raíz del árbol, y puede tener entre 1 a m elementos de datos clave. Los nodos de hoja pueden ir hacia adelante y hacia atrás.

Aquí vemos una comparación directa de rendimiento entre árboles:



El orden del T-Tree, B + -Tree y el O2-Tree utilizado fue m = 512.

El tiempo se registra para las operaciones de búsqueda, inserción y supresión con relaciones de actualización que varían entre 0% -100% para un índice de 50M registros, con las operaciones que resultan en la adición de otros 50M registros al índice.

Está claro que con una proporción de actualización de 0-10%, B-Tree y T-Tree tienen mejores resultados que O2-Tree. Sin embargo, con la proporción de actualización aumentado, el índice de O2-Tree funciona significativamente mejor que otras estructuras de datos.

¿Cuál es el caso para NoSQL?

Las bases de datos NoSQL ganaron mucha popularidad debido a su alto rendimiento, alta escalabilidad y facilidad de acceso. Sin embargo, todavía carecen de las características que proporcionan consistencia y confiabilidad. Afortunadamente, una serie de DBMS NoSQL abordan estos retos ofreciendo nuevas características para mejorar la escalabilidad y la fiabilidad.

No todos los sistemas de base de datos NoSQL funcionan mejor que las bases de datos relacionales. MongoDB y Cassandra tienen un rendimiento similar, y en muchos casos mejor, que en las bases de datos relacionales en operaciones de escritura y eliminación.

No hay correlación directa entre el tipo de almacenamiento y rendimiento de un DBMS NoSQL.

Las implementaciones de NoSQL experimentan cambios, por lo que el rendimiento puede variar. Por lo tanto, las mediciones de rendimiento a través de tipos de base de datos en diferentes estudios, siempre deben actualizarse con las últimas versiones del software de la base de datos para que estos números sean exactos.

Aunque no podemos ofrecer un veredicto definitivo sobre su rendimiento, he aquí algunos puntos a tener en cuenta:

- La indexación tradicional de árbol B y árbol T se utiliza comúnmente en bases de datos tradicionales.
- Un estudio ofreció mejoras mediante la combinación de las características de múltiples estructuras de indexación para llegar al Árbol-O2.
- El Árbol-O2 superó a otras estructuras en la mayoría de las pruebas, especialmente con enormes conjuntos de datos y con altas proporciones de actualización.
- La estructura Árbol-B presentó el peor desempeño de todas las estructuras de indexación cubiertas en este artículo.

Se puede y se debe hacer más trabajo para mejorar la consistencia de los DBMSs NoSQL. La integración de ambos sistemas, NoSQL y bases de datos relacionales, es un área que debería ser explorada.

NoSQL comercializa funciones de confiabilidad y consistencia para un rendimiento y proceso de escalabilidad extremo. Esto lo convierte en una solución especializada, ya que el número de aplicaciones que pueden depender de las bases de datos NoSQL sigue siendo limitado. Así aunque la especialización puede ser poco flexible, si queremos un trabajo especializado, rápido y eficaz lo más indicado va a ser NoSQL.

IV. RESULTADOS

A. Creacion de base de datos NoSQL con MongoDB

- MongoDB es una base NoSQL orientada a documentos
- Permite guardar documentos en formato de JSON
- Tiene esquema flexible, es decir que podemos cambiar la estructura de nuestros documentos sin ningu´n problema
- MongoDB est´a preparado para escalar f´acilmente de manera horizontal
- Dado que aprendimos ECMAScript vamos a utilizar un motor de base de datos que nos permite seguir utilizando este lenguaje para guardar nuestros datos.

B. Instalar MongoDB en Docker

- Ingresar sus credenciales creadas en Docker Hub para iniciar sesi´on en el aplicativo.Ubicar la apli- caci´on PowerShell, ejecutarla como Administrador. En la ventana de comandos de PowerShell escribir lo siguiente.

- Para instalar MongoDB primero tenemos que eje- cutar el siguiente codigo.

```
PS C:\> docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
35b42117c431: Pull complete
ad9c569a8d98: Pull complete
293b44f45162: Pull complete
0c175077525d: Pull complete
4e73525b52ba: Pull complete
a22695a3f5e9: Pull complete
420eb4b7be5d: Pull complete
017ec49b70bf: Pull complete
26470656e2db: Pull complete
6fec56c7382a: Pull complete
12f574d1345c: Pull complete
7102859c924d: Pull complete
555c1275dd6f: Pull complete
Digest: sha256:01dc9fb0b7aae875678047e2d8550beb6fc34b7e76c60a1e7d7048f6700dead0
Status: Downloaded newer image for mongo:latest
```

- Verificar que el contenedor se este ejecutando cor- rectamente mediante el comando:

```
PS C:\> docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
e711c6ddc67         mongo:3.5           "docker-entrypoint.s…"   35 minutes ago      Up 35 minutes       27017/tcp           mongobongo
```

- Proceder a verificar la imagen con el siguiente co- mando:

```
PS C:\> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mongo               latest              a3639b2a0f13        4 days ago          412MB
PS C:\>
```

- Seguidamente ejecutar el comando.Como respuesta se visualizar´a un ID que corresponde al contenedor:

```
PS C:\> docker run --name virginia-mongo -d mongo
093a88bc0df784cd59774211c1d23a78ecb5d03997b4fd2606cd300d36543dd3
PS C:\>
```

```
PS C:\Program Files\MongoDB\Server\4.0\bin> docker stop virginia-mongo
virginia-mongo
PS C:\Program Files\MongoDB\Server\4.0\bin> docker rm virginia-mongo
virginia-mongo
```

- Para conectar a nuestro localhost

```
PS C:\> cd 'C:\Program Files\MongoDB\Server\'
```

- Para exponer ese puerto para nuestro que podamos acceder al contenedor

```
PS C:\Program Files\MongoDB\Server> ls


    Directorio: C:\Program Files\MongoDB\Server


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        23/06/2019     11:34                4.0
```

- MongoDB est´a preparado para escalar f´acilmente de manera horizontal

```
PS C:\Program Files\MongoDB\Server\4.0> cd bin
PS C:\Program Files\MongoDB\Server\4.0\bin> ls


    Directorio: C:\Program Files\MongoDB\Server\4.0\bin


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        28/05/2019     21:36       18249411 bsondump.exe
-a----        28/05/2019     21:59           1568 InstallCompass.ps1
-a----         3/04/2018     18:58        2462720 libeay32.dll
-a----        28/05/2019     21:59       18461184 mongo.exe
-a----        28/05/2019     21:34            570 mongod.cfg
-a----        28/05/2019     22:01       32702976 mongod.exe
-a----        28/05/2019     22:01      360779776 mongod.pdb
-a----        28/05/2019     21:41       19350603 mongodump.exe
-a----        28/05/2019     21:38       18841845 mongoexport.exe
-a----        28/05/2019     21:38       18683050 mongofiles.exe
-a----        28/05/2019     21:39       19029681 mongoimport.exe
-a----        28/05/2019     21:40       19419576 mongorestore.exe
-a----        28/05/2019     21:58       16907264 mongos.exe
-a----        28/05/2019     21:58      188887040 mongos.pdb
-a----        28/05/2019     21:37       18930780 mongostat.exe
-a----        28/05/2019     21:41       18518542 mongotop.exe
-a----         3/04/2018     18:58         357888 ssleay32.dll
```

- MongoDB est´a preparado para escalar f´acilmente de manera horizontal

```
PS C:\Program Files\MongoDB\Server\4.0\bin> docker info virginia-mongo
"docker info" accepts no arguments.
See 'docker info --help'.

Usage:  docker info [OPTIONS]

Display system-wide information
```

- MongoDB est´a preparado para escalar f´acilmente de manera horizontal

```
PS C:\Program Files\MongoDB\Server\4.0\bin> docker inspect virginia-mongo
[
    {
        "Id": "093a88bc0df784cd59774211c1d23a78ecb5d03997b4fd2606cd300d36543dd3",
        "Created": "2019-06-23T16:44:02.229635Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "mongod"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 2547,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2019-06-23T16:44:06.209002Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
        "Image": "sha256:a3639b2a0f136f04004d639c6158b7f5c7a0947597f14277c95d762b5aa0b42",
        "ResolvConfPath": "/var/lib/docker/containers/093a88bc0df784cd59774211c1d23a78ecb5d03997b4fd2606cd300d36543dd3/
esolv.conf",
        "HostnamePath": "/var/lib/docker/containers/093a88bc0df784cd59774211c1d23a78ecb5d03997b4fd2606cd300d36543dd3/ho
stname",
        "HostsPath": "/var/lib/docker/containers/093a88bc0df784cd59774211c1d23a78ecb5d03997b4fd2606cd300d36543dd3/hosts
",
        "LogPath": "/var/lib/docker/containers/093a88bc0df784cd59774211c1d23a78ecb5d03997b4fd2606cd300d36543dd3/093a88b
c0df784cd59774211c1d23a78ecb5d03997b4fd2606cd300d36543dd3-json.log",
```

- En PowerShell ejecutar el siguiente comando ,Ver- ificar la eliminaci´on del contenedor con ejecutando

- Seguidamente ejecutar el comando:Como respuesta se visualizar´a un ID que corresponde al contenedor

```
PS C:\Program Files\MongoDB\Server\4.0\bin> docker run -p 27017:27017 --name virginia-mongo -d mongo
51e25f2ab7b85f1176fbdb8e8ba0d11da0122c641ccb6cfb2a8a0764f5da0642
```

- Para conectar a nuestro localhost

```
PS C:\Program Files\MongoDB\Server\4.0\bin> .\mongo.exe
MongoDB shell version v4.0.10
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("b69312f1-b904-4a0e-a09a-5dd1ab040d3b") }
MongoDB server version: 4.0.10
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2019-06-23T10:06:29.843-0700 I CONTROL  [initandlisten]
2019-06-23T10:06:29.843-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-06-23T10:06:29.843-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration
nrestricted.
2019-06-23T10:06:29.844-0700 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
```

C. Inserci´on y consulta de datos

- conexion de Mongo,instancia de Mongo Corriendo dentro de este contenedor docker

```
> db
test
> db.getCollection("test")
test.test
> db.getCollection("test").find("").count()
0
> db.getCollection("test").insert("{'name':' virginia aquino'}"
```

- Se hace una prueba de Mongo DB,disparar ,para que asi la base de datos consiga obtener la prueba de recoleccion de datos

```
> db.getCollection("test").insert("{'name':'virginia aquino'}")
2019-06-23T12:26:32.737-0500 E QUERY    [js] Error: argument to bsonsize has to be an object :
Bulk/addToOperationsList@src/mongo/shell/bulk_api.js:611:28
Bulk/this.insert@src/mongo/shell/bulk_api.js:654:20
DBCollection.prototype.insert@src/mongo/shell/collection.js:318:13
@(shell):1:1
> db.getCollection("test").insert({'name':'virginia aquino'})
WriteResult({ "nInserted" : 1 })
```

DISCUSION Y CONCLUSIONES

NoSQL permite el manejo de grandes volu´menes de datos y la posibilidad de tener un sistema dis- tribuido.

Las caracter´ısticas de las bases de datos NoSQL re- sponden a las necesidades actuales de las diferentes organizaciones, por lo que son una alternativa de- bido a su capacidad y a la velocidad. La integraci´on de ambos, NoSQL y bases de datos relacionales, es un ´area que deber´ıa ser explorada. Las base de datos NoSQL comercializa funciones de confiabilidad y consistencia para un rendimiento y proceso de escalabilidad extremo, convirtiendolo en una soluci´on especializada, ya que el n´umero de aplicaciones que pueden depender de las bases de datos NoSQL sigue siendo limitado

[1] Amazon. La bases de datos clave-valor definida. Accessed: 2019-06-20. [5] S. A. M´endez Aguilar. *USO DE BASES DE DATOS NOSQL DOCUMENTALES PARA CREAR SITIOS WEB* [2] Amazon. ¿que´ es una base de datos de documentos? Accessed: 2019-06-20. [3] GeeksforGeeks. Difference between sql and nosql. Accessed: 2019-06-20. [7] Kyocera Document Solutions. ¿que´ son las bases de datos documentales? Accessed: 2019-06-20. *TECTURA Y EJEMPLOS DE APLICACIO´N*. 2014. *DE ALTO RENDIMIENTO*. 2013. [6] Nicolas Nisnardi. Base de datos. Accessed: 2019-06-22. *[4]* R. Herranz G´omez. *BASES DE DATOS NOSQL: ARQUI-*