Universidad De La Salle Bajío

Facultad de Tecnologías de Información

MTIE 501

Modelos de arquitecturas orientadas a los servicios

Investigación 6 - "ElasticSearch"

Presenta:
Marco Antonio Ramos Herrera

Profesor:
José Luis Rosas Peimbert

León, Gto a 11 de Marzo de 2021

INDICE

INDICE 1
INTRODUCCIÓN 2
DESARROLLO 3
Bases de datos de índice invertido 3
ElasticSearch 5
Clustering y Sharding en bases de datos relacionales y NOSQL 6
Sharding 6
Clustering 7
CONCLUSIONES 7
REFERENCIAS 8

INTRODUCCIÓN

Relacional y NoSQL son dos tipos de sistemas de base de datos que se implementan normalmente en aplicaciones nativas en la nube. Se compilan de forma diferente, almacenan los datos de manera diferente y se accede a ellos de forma diferente. En esta sección, veremos ambos. Más adelante en este capítulo, veremos una tecnología de base de datos emergente denominada NewSQL.

Las bases de datos relacionales han sido una tecnología predominante para décadas. Están consolidados, probados y ampliamente implementados. Los productos de base de datos, las herramientas y la experiencia en competencia abundan. Las bases de datos relacionales proporcionan un almacén de tablas de datos relacionadas. Estas tablas tienen un esquema fijo, usan SQL (Lenguaje de consulta estructurado) para administrar datos y admiten garantías ACID.

Las bases de datos no-SQL hacen referencia a almacenes de datos no relacionales de alto rendimiento. Excel en sus características de facilidad de uso, escalabilidad, resistencia y disponibilidad. En lugar de combinar tablas de datos normalizados, NoSQL almacena datos no estructurados o semiestructurados, a menudo en pares clave-valor o documentos JSON. Las bases de datos no-SQL normalmente no proporcionan garantías ACID más allá del ámbito de una sola partición de base de datos. Los servicios de gran volumen que requieren un tiempo de respuesta de subsegundo favorecen los almacenes de los mismos.

DESARROLLO

Bases de datos de índice invertido

Las bases de datos de índice invertido crean una estructura de datos para llevar a cabo una búsqueda de texto completa. Almacena un mapeo de contenido, como palabras o números, a sus ubicaciones en una tabla , o en un documento o un conjunto de documentos (nombrados en contraste con un índice de avance , que se asigna de los documentos al contenido). El propósito de un índice invertido es permitir búsquedas rápidas de texto completo , a costa de un mayor procesamiento cuando se agrega un documento a la base de datos. El archivo invertido puede ser el archivo de la base de datos en sí, en lugar de su índice. Es la estructura de datos más popular utilizada en los sistemas de recuperación de documentos , utilizada a gran escala, por ejemplo, en los motores de búsqueda.

Hay dos variantes principales de índices invertidos: Un índice invertido a nivel de registro (o índice de archivo invertido o simplemente archivo invertido) contiene una lista de referencias a documentos para cada palabra. Un índice invertido a nivel de palabra (o índice invertido completo o lista invertida) contiene además las posiciones de cada palabra dentro de un documento. La última forma ofrece más funcionalidad (como búsquedas de frases), pero necesita más potencia de procesamiento y espacio para crearse.

Al contrario que las aproximaciones clásicas de bases de datos, donde cada uno de los documentos se almacena asociado a una clave, en un índice invertido se guardan partes del documento por las que buscar (normalmente palabras) y referencias a los documentos en los que aparecen.

En un índice podríamos guardar los siguientes documentos, cada uno de ellos asociado a su clave:

Clave	Documento
1	¿Qué es esto?
2	Esto es un texto
3	Este texto es otro texto

Si se quiere hacer una búsqueda de documentos en los que aparece la palabra “texto”, el proceso consistiría en ir documento a documento comprobando si contiene dicha palabra, lo cual, especialmente para gran número de documentos, puede suponer tiempos de búsqueda elevados.

En el caso de índices invertidos se realiza un proceso previo en el que, cada uno de los documentos se dividen en cadenas de texto buscable y se almacenan asociadas a los documentos en los que aparecen. Por ejemplo, para el caso anterior, dividiendo por palabras (y eliminando signos de puntuación) tendríamos:

Palabra	ID de documento
qué	1
es	1, 2, 3
esto	1, 2
un	2
texto	2, 3
este	3
otro	3

En este caso, saber en qué documentos se encuentra la palabra “texto” es directo, ya que, asociados a la palabra, se pueden recuperar los identificadores de los documentos en los que aparece (en este caso 2 y 3).

Estos índices invertidos se pueden generar además con información adicional, como las posiciones que ocupan las palabras en el documento o las veces que aparecen, de tal forma que se pueda usar esa información para funcionalidades adicionales como resaltado de coincidencias u ordenación por relevancia:

Palabra	(Documento, Posición)
qué	(1,1)
es	(1,2) (2,2) (3,3)
esto	(1,3) (2,1)
un	(2,3)
texto	(2,4) (3,2) (3,5)
este	(3,1)
otro	(3,4)

Algo que se debe de tener en cuenta es:

- El uso de índices invertidos hace que las búsquedas de texto sean especialmente rápidas gracias al procesado previo de los documentos y la forma en la que se almacena la información…
- … sin embargo esta ventaja se consigue a costa de aumentar el tiempo de procesado durante la indexación de documentos.
- El modo en el que se guarda la información en un índice invertido hace que ciertas búsquedas sean especialmente costosas, como las que implican excluir ciertas palabras o incluir aquellas que tienen un sufijo común, ya que en esos casos se deben recuperar primero todas las palabras que cumplen la condición al igual que se haría en un índice clásico.

La indexación basada en un índice invertido es un componente central de un típico algoritmo de un motor de búsqueda. En ellos, es muy importante optimizar la velocidad de la consulta.

Motores de búsqueda verticales tales como Solr o Elasticsearch, emplean el índice invertido.

ElasticSearch

Dada la gran cantidad de información con la que cuentan algunos sitios web, solo se puede garantizar un alto nivel de funcionalidad implementando una búsqueda de texto completo. Elasticsearch es un software libre y se basa en la versión gratuita de Apache Lucene.

Elasticsearch ofrece las ventajas de sus predecesores e incluye otras características. Como en el caso de Lucene, la búsqueda se realiza mediante un índice, pero en lugar de examinar todos los documentos, el programa comprueba un índice de documentos que se ha creado previamente en el que todo el contenido se almacena de forma preparada. Este proceso requiere mucho menos tiempo que la consulta de todos los documentos.

Aunque Lucene ofrece total libertad en cuanto a dónde y cómo utilizar la búsqueda de texto completo, con este software se parte de cero. A cambio, Elasticsearch facilita los primeros pasos en la Web. Con Elasticsearch es posible construir un servidor de búsqueda estable en poco tiempo que también se puede distribuir fácilmente entre varios equipos.

Utilizando el llamado sharding, varios nodos (diferentes servidores) se unen para formar un clúster: Elasticsearch desglosa el índice y distribuye las partes individuales (shards) entre varios nodos, dividiendo así la carga de cálculo. Para proyectos grandes, la búsqueda de texto completo es mucho más estable y, en caso de que busques mayor seguridad, también puedes copiar los fragmentos a varios nodos.

Elaticsearch se basa, al igual que Lucene, en el lenguaje de programación orientado a objetos Java. El motor de búsqueda produce

los resultados en formato JSON y los entrega a través de un servicio web REST. La API facilita la integración de la función de búsqueda en un sitio web.

Además, Elasticsearch ofrece con Kibana, Beats y Logstash (conocidos juntos como Elastic-Stack) servicios adicionales que se pueden utilizar para analizar la búsqueda de texto completo. La empresa Elastic, que está detrás del desarrollo de Elasticsearch y que fue fundada por el inventor del programa, también ofrece servicios de pago, como el cloud hosting.

Clustering y Sharding en bases de datos relacionales y NOSQL

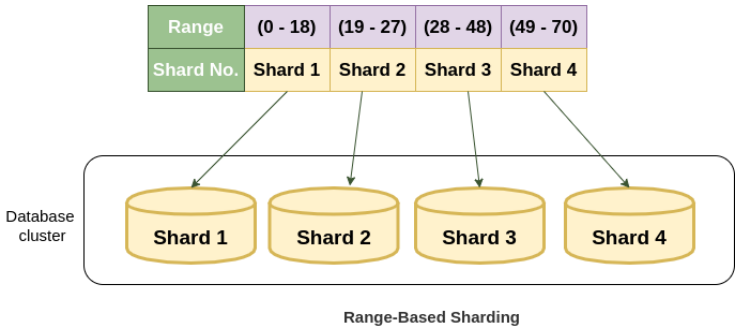
Si se comparan con las bases de datos relacionales, las bases de datos NoSQL son más escalables y ofrecen un mayor rendimiento; además, su modelo de datos aborda varias cuestiones que el modelo relacional pasa por alto:

- Grandes volúmenes de datos estructurados, semiestructurados y no estructurados en constante cambio
- Sprints de desarrollo ágiles, iteración rápida de los esquemas y generación frecuente de código
- Programación orientada a objetos flexible y fácil de usar
- Arquitectura de escalado horizontal distribuida geográficamente, en lugar de una arquitectura monolítica.

Sharding

Por el modo en el que están estructuradas, las bases de datos relacionales se suelen escalar de forma vertical: un solo servidor debe alojar toda la base de datos para garantizar un rendimiento aceptable de las operaciones JOIN y transacciones entre tablas. Esto encarece su coste con rapidez, limita la escalabilidad y crea un número de puntos de fallo relativamente pequeño para la infraestructura de bases de datos. La solución para que las aplicaciones puedan crecer de forma rápida es permitir la escalabilidad horizontal añadiendo servidores, en lugar de concentrar más capacidad en un único servidor.

Es posible aplicar el sharding a una base de datos SQL en un gran número de instancias de servidor, pero para ello normalmente se necesitan sistemas SAN y otras configuraciones complejas para que el hardware actúe como un único servidor. Como las bases de datos relacionales no proporcionan esta función de forma nativa, los equipos de desarrollo deben encargarse de implementar varias bases de datos de este tipo en diferentes máquinas. Los datos se almacenan en cada instancia de base de datos de forma autónoma. El código de la aplicación se desarrolla para distribuir los datos, distribuir las consultas y agregar los resultados de los datos en todas las instancias de la base de datos. Se debe desarrollar código adicional para manejar los fallos de los recursos, y realizar operaciones JOIN entre diferentes bases de datos, así como para aplicar otros requisitos, como el reequilibrio de datos y la replicación. Además, muchas de las ventajas de las bases de datos relacionales, como la integridad transaccional, se ven afectadas o eliminadas al utilizar el sharding manual.

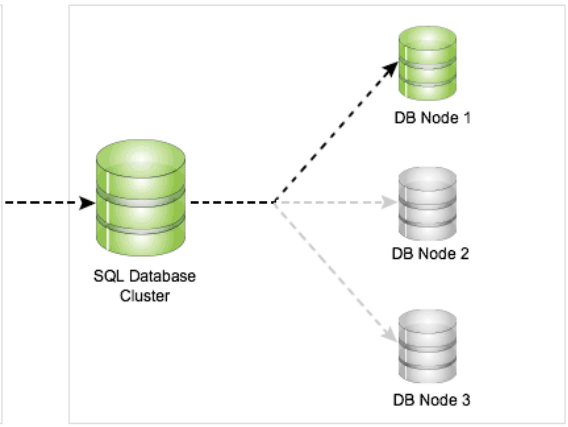


Range-Based Sharding

Por el contrario, las bases de datos NoSQL suelen admitir el auto-sharding, es decir, pueden distribuir los datos de forma nativa y automática entre un número arbitrario de servidores, sin que la aplicación deba tener constancia de la composición del grupo de servidores. Los datos y la carga de consultas se equilibran automáticamente entre los servidores, y cuando uno falla, se puede sustituir de forma rápida y transparente sin que la aplicación deje de funcionar.

Clustering

La mayoría de las bases de datos NoSQL también admiten replicación automática de bases de datos para garantizar la disponibilidad en caso de que se produzcan interrupciones de servicio o paradas de mantenimiento planificadas. Las bases de datos NoSQL más sofisticadas son autorreparables y ofrecen las funciones de conmutación por error y recuperación, así como la posibilidad de distribuir la base de datos entre varias regiones geográficas para soportar fallos regionales y permitir la localización de los datos. A diferencia de las bases de datos relacionales, las bases de datos NoSQL no necesitan utilizar otras aplicaciones o complementos con un precio elevado para implementar la replicación.



CONCLUSIONES

La agilidad de los sistemas y servicios es algo primordial en la actualidad, si bien ya con el hecho de poder contar infraestructura en la nube ya se puede garantizar que podemos agilizar los procesos en los sistemas debido a la flexibilidad en los recursos y a la rapidez en que se puede desplegar dicha infraestructura, pero hay que tomar en cuenta la cantidad de información que se maneja actualmente en los sistemas o en los sitios de internet, es por ello que implementar herramientas como ElasticSearch brindan aún mayor beneficio al momento de generar consultas o búsquedas en los sitios comentados debido a su escalabilidad y principalmente por la velocidad en la que se pueden ejecutar dichas consultas.

Ahora, por el lado de las bases de datos relacionales y NoSQL, pues cada una tiene sus ventajas y desventajas, y la diferencia entre usar una u otra dependería del mismo sistema, aunque en la actualidad como la mayoría de los sistemas son de gestión de contenido, aplicaciones móviles, análisis en tiempo real, las cuales generan un crecimiento rápido, la mejor opción son bases de datos NOSQL.

REFERENCIAS

https://aws.amazon.com/es/getting-started/hands-on/design-a-database-for-a-mobile-app-with-dynamodb/5/

https://paradigma-digital.medium.com/3-claves-para-entender-elasticsearch-bd31830e44e7

https://es.wikipedia.org/wiki/%C3%8Dndice_invertido#:~:text=Un%20%C3%ADndice%20invertido%20es%20una,una%20b%C3%BAsqueda%20de%20texto%20completa.

https://es.qaz.wiki/wiki/Inverted_index

https://copro.com.ar/Indice_invertido.html

https://docs.microsoft.com/es-es/dotnet/architecture/cloud-native/relational-vs-nosql-data

https://www.mongodb.com/es/collateral/top-5-considerations-when-evaluating-nosql-databases

https://www.muylinux.com/2021/01/25/amazon-bifurca-elasticsearch-open-source/

https://www.ionos.mx/digitalguide/servidores/configuracion/que-es-elasticsearch/

https://www.elastic.co/es/elasticsearch/

https://apiumhub.com/es/tech-blog-barcelona/usar-elasticsearch-ventajas-libros/