#Elasticsearch
Erick Madrigal Zavala
2018146983

Datos en: documentos e índices

Elasticsearch es un almacenamiento de documentos distribuidos. En lugar de almacenar la información en forma de tablas, Elasticsearch almacena complejas estrucutras de datos que se serializan como documentos JSON. Elasticsearch usa una estrucura de datos llamada indices invertidos que soporta busquedas de texto de manera muy rapida. Un indice inverso lista cada palabra unica que aparece en un docuemento e identifica cada documento en que la palabra se encuentra.
Elasticsearch indexa todos los datos de cada campo y cada campo de indexado tiene una estrucuta de datos dedicada y obtimizada. Elasticsearch tambien tiene la habilidad de ser "schema-less", que significa que los documentos pueden ser indexados sin tener que decir explicitamente como manejar cada uno de los campos que posee el documento. Cuando se activa el mapeo dinámico, Elasticsearch detecta y añade campos al indice automaticamente. El usuario puede definir reglas de control para el mapeo dinámico y además tomar el control total de como se guardan los datos en los indices por medio de definir explicitamente el mapeo.
Definir sus propios mapeos le permite:

* Distinguir entre string con el texto completo y campos que contengan un valor exacto de un string
* Realizar un analisis de lenguaje especifico del texto.
* Optimizar campos para coincidencias parciales.
* Usar formatos de fechas personalizadas.
* Usar tipos de datos como "geo_points" y "geo_shapes" las cuales no podrian ser detectados automaticamente.
  El analisis de cadenasque se aplica a un campo con todo el texto durante la indexación es tambien usado durante el tiempo de busqueda. Cuando se realiza un query de un campo de texto, el query del texto pasa por el mismo analizis antes de que los terminos sean buscados en el indice

Salida de información: busqueda y análisis

El REST API de Elasticsearch soporta querys estructurados querys de "full-text" y querys complejos que combinan los dos. Los querys estructurados son similares a los querys que se construyen para SQL. Los querys de "full-text" buscan todos los documentos que concuerden con el string del query y los devuelven ordenados por relevancia. Además soporta busqueda de terminos individuales y da sugerencias para el autocompletado.
El usuario además puede construir querys con el estilo SQL para buscar y agregar datos de manera nativa a Elasticsearch, los drivers de JDBC y ODBC permiten un amplio ranfo de aplicaciones de terceros que puden interactuar con Elasticsearch via SQL.

Analizando los datos

Las agregaciones de Elasticsearch le permiten al usuario crear resumenes comlpetos de los datos y obtener información sobre métricas, patrones y tendencias clave.
Debido a que las agregaciones aprovechan las mismas estructuras de datos que las búsquedas, también son muy rápidas. Esto le permite analizar y visualizar sus datos en tiempo

real. Los informes y cuadros de mando se actualizan a medida que cambian los datos para poder tomar medidas basadas en la información más reciente. Además, las agregaciones funcionan junto con las solicitudes de búsqueda. Puede buscar documentos, filtrar resultados y realizar análisis al mismo tiempo, sobre los mismos datos, en una única solicitud.
Puede utilizar funciones de "Machine Learning" para crear líneas de base precisas del comportamiento normal de sus datos e identificar patrones anómalos. Con el "Machine Learning", puede detectar:

* Anomalías relacionadas con desviaciones temporales en valores, recuentos o frecuencias.
* Rareza estadística.
* Comportamientos inusuales para un miembro de una población.

Escalabilidad y Resilencia: clusters, nodos y shards

Se pueden añadir servidores (nodos) a un clúster para aumentar su capacidad y Elasticsearch distribuye automáticamente los datos y la carga de consultas entre todos los nodos disponibles. En realidad, un índice de Elasticsearch no es más que una agrupación lógica de uno o más shards físicos, donde cada shard es en realidad un índice autónomo. Al distribuir los documentos de un índice en varios shards y distribuir esos shards en varios nodos, Elasticsearch puede garantizar la redundancia, que protege contra fallos de hardware y aumenta la capacidad de consulta a medida que se añaden nodos a un clúster. A medida que el clúster crece (o decrece), Elasticsearch migra automáticamente los fragmentos para reequilibrar el clúster.
Existen dos tipos de shards: primarios y réplicas. Cada documento de un índice pertenece a un shard primario. Un shard réplica es una copia de un shard primario. Las réplicas proporcionan copias redundantes de sus datos para protegerlos contra fallos de hardware y aumentar la capacidad de servir peticiones de lectura como la búsqueda o recuperación de un documento. El número de shards réplica puede modificarse en cualquier momento, sin interrumpir las operaciones de indexación o consulta.
Hay una serie de consideraciones de rendimiento y compensaciones con respecto al tamaño de los fragmentos y al número de fragmentos primarios configurados para un índice. Consultar muchos fragmentos pequeños hace que el procesamiento por fragmento sea más rápido, pero más consultas significa más sobrecarga, por lo que consultar un menor número de fragmentos más grandes podría ser más rápido.
Como punto de partida:

* Intenta mantener el tamaño medio de los fragmentos entre unos pocos GB y unas decenas de GB. Para casos de uso con datos basados en el tiempo, es habitual ver fragmentos de entre 20 GB y 40 GB.
* Evita el problema de los gazillion shards. El número de fragmentos que puede contener un nodo es proporcional al espacio disponible en el montón. Como regla general, el número de shards por GB de espacio de heap debe ser inferior a 20.
  Los nodos de un clúster necesitan conexiones buenas y fiables entre sí. Para ofrecer mejores conexiones, los nodos suelen ubicarse en el mismo centro de datos o en centros de datos cercanos. Sin embargo, para mantener una alta disponibilidad, también es necesario evitar cualquier punto único de fallo. En caso de que se produzca un corte importante en una ubicación, los servidores de otra ubicación deben ser capaces de tomar el relevo. ¿Cuál es la solución? La replicación entre clústeres (CCR).