Lloyd Quadros

CS 410 Technology Review

Overview:

ElasticSearch is a distributed, text-based search and analytics engine used for storing and retrieving data in free form. It is built on top of Apache Lucene which is the driver for ElasticSearch's querying and indexing speed. ElasticSearch users use a REST api to connect to their ElasticSearch cluster and to index and query documents which makes ElasticSearch very easy to integrate into any sort of web application.

How it works: (information from source 1)

Lowest Level: Inverted Indices

All data in an ElasticSearch cluster, at the very lowest level, is stored in a data structure known as an inverted index. As we learned in the text retrieval part of the course, inverted indexes store information about the documents in our cluster such as the terms they contain, the term frequency, etc. One thing to note is that ElasticSearch Inverted Index data is immutable. There are multiple reasons for why data in the inverted indexes should remain immutable such as reducing costly disk I/O and the amount of RAM needed to cache indices. Because data is immutable in our inverted indices, documents that are deleted from our cluster are simply marked for deletion, and document updates work by marking the old version of the document for deletion, and writing the updated version of the document to a new inverted index. Overtime, as one can imagine, the data stored in our inverted indices would be a little messy because of this way we update and delete documents, but the cluster makes sure to clean up it's inverted indices by ensuring that all documents not marked for deletion are moved into a single inverted index, and the ones that are marked for deletion are cleared off the disk.

Intermediate Level: Shards

All inverted indices are packaged into these containers called shards, and shards are managed by our cluster's nodes. This is where the distributed nature of ElasticSearch comes into play. Shards are capable of executing queries and indexing requests. The cluster makes sure that each shard manages the same number of documents so that no shard becomes overwhelmed with executing queries and indexing requests. There are 2 types of shards: primary shards and replica shards. Primary shards can only be created when the cluster is created. Therefore it is important to know how much data you plan to store in your cluster so that you can figure out how many primary shards would be sufficient to carry out your indexing and query requests. If you have too few shards, each shard will have lots of data which slows down cluster performance. On the contrary, if you have too many shards, for each query and index request, the cluster has to queue up an unnecessary amount of shards which also slows down cluster performance. Finding the sweet spot of the right amount of primary shards takes a lot of planning and experimentation. Replica Shards allow you to back up all of your primary shard data to another shard which the cluster can use in case your primary shards go offline.

Top Level: Nodes

All of your shards are managed by nodes. Unlike shards, nodes can be added or removed at any time. The job of the cluster nodes is to “wake up” their shards and assign them the task of querying or indexing data when a request comes into the cluster. The cluster assigns each shard to a node equally to ensure that data is properly backed up in the event of node failure, and to ensure that no nodes are being overwhelmed. Nodes offer concurrent execution of requests on its shards which boosts cluster performance and speed.

My Experience with ElasticSearch:

This semester, I had the opportunity to use ElasticSearch in my fall internship that I was partaking in. We spun up and deployed an ElasticSearch cluster in Google Cloud and used the Java REST high level client to interact with the cluster programmatically. Our cluster configuration was just 1 primary shard, 2 replica shards, 3 data nodes, and 1 master node. We used our java application to read in company documents and to index them into our cluster indexes. For my internship project this semester, I essentially had to analyze how well ElasticSearch would store company documents.

What I found was that ElasticSearch’s indexing and querying speed was extremely fast: I calculated an average of 10 ms querying and indexing for a 40 GB index where each document was around 32 MB. Additionally, ElasticSearch allows you to index documents of any schema as well as query documents using JSON formatted schema so it is extremely user friendly in-terms of indexing and querying operations. ElasticSearch also has SQL querying functionality which allows you to query documents using SQL if that is a language you are most comfortable with as well as Lucene syntax querying which allows you to perform true text based queries: you enter a string, and it returns a document similar to the string input. ElasticSearch by default returns the top 10 document hits and has functionality that allows the user to use what’s called a “point in time search” + scroll pagination that allows users to stale paginate search results without using very much memory to store the search context.

One major obstacle we encountered when trying to implement an ElasticSearch database to store our company inventory documents was large shard sizes. Because our data that we wanted to store in ElasticSearch was infinitely growing, this resulted in our shards getting very large in size which meant that search operations were getting slowed down drastically. To fix this problem, you can perform a new reindexing operation to get a new index with a new primary shard. However, this method is known to be very time consuming and to cause some down time. So the one downside to ElasticSearch that I found was that adding new primary shards at any time isn’t allowed which is very inconvenient for use cases like mine.

Sources:

1. https://github.com/elastic/elasticsearch-definitive-guide/tree/master/075_Inside_a_shard
2. https://www.elastic.co/guide/en/elasticsearch/reference/current/paginate-search-results.html
3. https://www.elastic.co/guide/en/elasticsearch/reference/current/sql-rest-overview.html