Andrew Hoeft (arhoeft2)

Apache Lucene-based Databases: Solr vs. Elasticsearch

Introduction

This technology review seeks to understand the differences between Apache Solr and Elasticsearch – the two primary competing databases which leverage the Apache Lucene library. These databases have been increasingly turned to for scalable, performant search against massive text corpora and underpin such use cases as application and web search, log aggregation, geospatial search, and business analytics.

Apache Lucene is the original Java-based fast search library that makes both of these databases possible. Currently on version 8.7, Lucene is capable of rapidly indexing high data volumes and accomplishing a number of other novel tasks: spell-checking text, identifying and highlighting search hits, and faceting search results. This last task is particularly useful for search engines that back e-commerce websites: faceting allows a user to search on a specific term, then partition results based on a matching attribute to search on a second specific term, etc. Since Lucene is just a library, it cannot be used as a standalone application.

Apache Solr Overview

In its simplest interpretation, Apache Solr is the executable database application that does the things Apache Lucene claims to be capable of. As with Apache Lucene, Solr is on its 7.8 release. It utilizes the Lucene library to index ingested data and return search results when a query is performed and is innately scalable, both vertically (on the same server with more resources) and horizontally (distributed across multiple resources). However, in order to adequately accomplish horizontal scaling, other members of the Apache software family are required (namely, Apache Zookeeper).

As a consequence of its age and open source development and support community, Solr's documentation is rather cumbersome, and its commercial support is nonexistent. In order to use this service adequately for enterprise use cases, administrators must become experts in the database's functionality and features. The dominant features are indexing and sharding.

Elasticsearch Overview

As the commercial counterpart to Apache Solr's open source offering, Elastic describes its Elasticsearch database as "a distributed, open source search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured."[1] As of version 7.10, Elasticsearch is leading the Lucene adoption and usage market with a larger user community, a stronger feature roadmap, and a commercial support offering. Whereas Solr established the market, demonstrated to enterprise developers how to perform indexing and search across large bodies of free text, and built an early user community, Elasticsearch has come from behind to become a more formidable product with a prominent commercial offering.

Andrew Hoeft (arhoeft2)

Solr and Elasticsearch: Comparison

One of the dominant places where Elasticsearch exceeds Solr's narrower usability matrix is in the configuration and operation of horizontal scaling for distributed fault tolerance and high availability. Elasticsearch improves on Solr by incorporating the command-and-control aspect of the distributed storage lifecycle by embedding the Zookeeper service into the normal Elasticsearch runtime.

Typically, given its commercial footprint, Elasticsearch runs on larger, more capable hardware so its higher minimum runtime requirements are not deemed problematic. The primary demand here is for 1GB of heap space for minimal indexing and runtime state management. By comparison, Solr only requires 512MB in the minimum case. Both databases indicate a 20-30% increase in data volume over the baseline data footprint for maintaining indices.

As per the usual architecture concerns, the fact that these two services are Java-oriented have a marked runtime disadvantage compared to other database services such as Postgres for very specific tasks. The ones that come to mind where other databases would outperform Solr or Elasticsearch are either use cases that can support strong typing of relational database field values or use cases that depend on highly traversable, relational data models. Solr and Elasticsearch are more capable when the use cases require support for unspecified data structures (accepting arbitrarily shaped document schemas). As with any data storage and retrieval capability, these tools work best when applied against the problems they naturally work well for and will struggle when evaluated against non-optimal metrics.

At the end of the day, Solr and Elasticsearch will both do an equivalent job for most commercial or open-source use cases. Elasticsearch is increasingly preferred due to its commercial support services and better feature roadmap. When it is preferred, Solr has an advantage in being purely open source, explorable, and extendable. For some use cases, the security aspects of being able to know and evaluate the full source code far outweigh any benefits of providing commercial support. It does not appear that either choice would be "bad" when fast, faceted, free-text search is required.

References

1. https://lucene.apache.org
2. https://lucene.apache.org/core/
3. https://lucene.apache.org/solr/guide/8_7/a-quick-overview.html
4. https://www.elastic.co/what-is/elasticsearch
5. https://logz.io/blog/solr-vs-elasticsearch/
6. https://sematext.com/blog/solr-vs-elasticsearch-differences/