Technology Review: Apache Lucene and Elasticsearch

Introduction

Lucene is an open-source java library that affords reliable high performance search. It was originally created in 1999[1] and is still the primary search engine library used by many large companies such as Apple, Comcast, Disney, and IBM[2].

Lucene affords searching for queries in a document or collection of documents, after building an inverted index. *Lucene is an inverted full-text index. This means that it takes all the documents, splits them into words, and then builds an index for each word. Since the index is an exact string-match, unordered, it can be extremely fast*[3]

Elasticsearch is an open-source java search engine built on top of Lucene. *It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. It was originally created in 2010*[4] and is still the primary search engine used by many large companies such as Uber, Shopify, Instacart, and Robinhood.[5]

Elasticsearch runs as a HTTP RESTful cluster, which intelligently handles the distribution of data across its nodes in a methodology very similar to how Lucene organizes data on a single machine. It has clients (REST APIs) in a number of different programming languages including: Java, JavaScipt, Ruby, Go, .NET, PHP, Perl, and Python. This significantly reduces the barrier to entry vs implementing Lucene at scale yourself.

In short, Lucene is a highly optimized single machine search solution, and Elasticsearch is a highly developed distributed search engine cluster built on top of Lucene.

Why use Lucene over a relational database?

The table in a traditional relational database or NoSQL database needs to at least have its scheme defined when it is created. There are some clear constraints on the definition of the primary key, columns and the like. However, there are no such constraints on a Lucene index.[6] A query will be made without forcing the data to fit loose or heavy coupling to database internals. This allows for any document to be indexed and searched with no additional time required of the user to constrain the document to a predefined schema.

---

[1] https://www.google.com/search?q=when+was+lucene+released&oq=when+was+lucene+released

[2] https://cwiki.apache.org/confluence/display/LUCENE/PoweredBy

[3] https://stackoverflow.com/questions/2705670/how-does-lucene-work

[4] https://www.google.com/search?q=when+was+elasticsearch+released&oq=when+was+elasticsearch+released

[5] https://stackshare.io/elasticsearch

[6] https://alibaba-cloud.medium.com/analysis-of-lucene-basic-concepts-5ff5d8b90a53

Why not simply use Lucene?

The primary qualm I would have with using Lucene directly is that I don't want to install java on every single machine I want to search on. Elasticsearch allows for any machine/website to make HTTP method calls to the cluster. Simplification of the application programing interface to RESTful HTTP method calls allows even shell scripts to make queries with curl.

Another reason is that provided a large enough cluster, Elasticsearch allows for scalable in-memory search. Although Lucene has an org.apache.lucene.index.memory class which allows for in-memory-only search, however it is only capable of making a search of one document per instance. *Rather than targeting fulltext search of infrequent queries over huge persistent data archives (historic search), this class targets fulltext search of huge numbers of queries over comparatively small transient realtime data (prospective search). Each instance can hold at most one Lucene "document", with a document containing zero or more "fields", each field having a name and a fulltext value.*[7]

This becomes significant when I want to use something like a website as a source document for Lucene. To do so, I need to download the page and store it as a local document on the machine Lucene is using. As more and more documents are located on the web, this becomes more of an issue for using Lucene directly vs having the Elasticsearch cluster process the url contents directly in memory.

Conclusion

Lucene implements *Powerful, Accurate and Efficient Search Algorithms*[8] that afford *Scalable, High-Performance Indexing*[9] on a single machine. It can *create schema automatically at runtime, like magic.*[10] When wrapped with Elasticsearch, it can be optimized as a fault tolerant distributed cluster of any scale, with a RESTful API to simplify making queries.

Any individual, or organization would be hard pressed to produce a scaleable search solution that was as fast, distributed, reliable, or easy to use as Elasticsearch.

---

[7] https://lucene.apache.org/core/8_10_1/memory/index.html

[8] https://alibaba-cloud.medium.com/analysis-of-lucene-basic-concepts-5ff5d8b90a53

[9] https://alibaba-cloud.medium.com/analysis-of-lucene-basic-concepts-5ff5d8b90a53

[10] https://stackoverflow.com/questions/27793721/what-is-the-difference-between-lucene-and-elasticsearch

References

https://www.google.com/search?q=when+was+lucene+released&oq=when+was+lucene+released

https://cwiki.apache.org/confluence/display/LUCENE/PoweredBy

https://stackoverflow.com/questions/2705670/how-does-lucene-work

https://www.google.com/search?q=when+was+elasticsearch+released&oq=when+was+elasticsearch+released

https://stackshare.io/elasticsearch

https://alibaba-cloud.medium.com/analysis-of-lucene-basic-concepts-5ff5d8b90a53

https://lucene.apache.org/core/8_10_1/memory/index.html

https://stackoverflow.com/questions/27793721/what-is-the-difference-between-lucene-and-elasticsearch