Indexing Framework For Exploratory Data Analytics

Bachelor Thesis

written by

Khairi Abidi
(born April 25th, 1997 in Jendouba, Tunisia)

under the supervision of **Dr Jhon Doe**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

Bachelor in Computer Science

at the *University Of Jendouba*.

Date of the public defense:	**Members of the Thesis Committee:**
MAY 20, 2020	Dr Jack Smith
	Prof Dr Jane Williams
	Dr Jill Jones
	Dr Albert Heijn

CONTENTS

ABSTRACT

It is very important that you include an abstract in your thesis. This will be used in the online series ILLC Publications.

ACKNOWLEDGEMENT

acknowledgement goes here

CHAPTER 1

GENERAL INTRODUCTION

CHAPTER 2

STATE OF THE ART

2.1 Introduction

In this section, we discuss most popular search and indexing engines and their advantages and disadvantages. and make a short comparison, first we start with an introduction to indexing.

2.2 Indexing

indexing is a mechanism to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed.
It is a data structure technique which is used to quickly locate and access the data in a database.
An Index is a small table having only two columns.

- The first column comprises a copy of the primary or candidate key of a table.
- The second column contains a set of pointers for holding the address of the disk block where that specific key value stored.

2.3 Apache Lucene

2.3.1 What is Lucene ?

Apache Lucene is a high-performance, scalable information retrieval (IR) library. IR refers to the process of searching for documents, information within documents, or metadata about documents.

2.3.2 Apache Lucene Features

Lucene provides search over documents; where a document is essentially a collection of fields. A field consists of a field name that is a string and one or more field values. Lucene does not in any way constrain document structures. Fields are constrained to store only one kind of data, either binary, numeric, or text data. There are two ways to store text data.
Lucene provides many ways to break a piece of text into tokens as well as hooks that allow you to write custom tokenizers. Lucene has a highly expressive search API that takes a search query and returns a set of documents ranked by relevancy with documents most similar to the query having the highest score.
In a nutshell, the features of Lucene can be described as follows:

Scalable and High-Performance Indexing

- Small RAM requirements.
- Incremental indexing as fast as batch indexing.
- Index size roughly 20-30% the size of text indexed.

Powerful, Accurate, and Efficient Search Algorithms

- Provides Ranked search
- Supports many powerful query types: phrase queries, wildcard queries, proximity queries, range queries and more.
- Provides fielded searching.
- Supports multiple-index searching with merged results.
- It allows simultaneous update and searching.
- Supports sorting by any field.

Cross-platform solution

- Available as Open Source software.
- It is 100%-pure Java.

2.3.3 How Does Apache Lucene Work?

In this Section, we will discuss how does Apache Lucene work towards indexing and searching.

Using Inverted Index

Lucene stores it's input into a data structure called inverted index .This data structure makes efficient use of disk space while allowing quick keyword lookups.
What makes this structure inverted is that it uses tokens extracted from input documents as lookup keys instead of treating documents as the central entities. In other words, rather than trying to answer the question "What words are contained in this document?" this struc- ture is optimized for providing quick answers to "Which documents contain word X ?".

Document Indexing

Document indexing consists of first constructing a document that contains the fields to be indexed or stored, then adding that document to the index. Every Lucene Index consists of one or more segments, as depicted in .
Each segment is a standalone index, holding a subset of all indexed documents.

2.4 Search Engines

In this section, we present the most popular open-source search engines and make a comparison of them in order to find the one who fits our need.

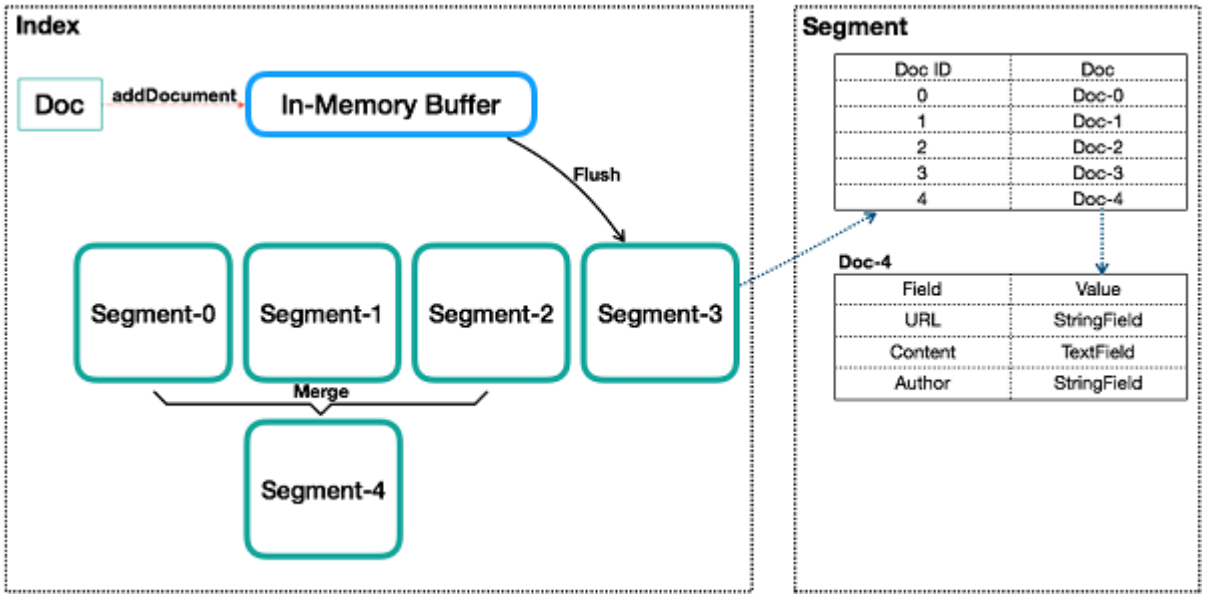


Figure 2.1: The figure shows the basic internal structure of an index. The data in a segment is represented abstractly rather than as a realistic representation of the actual data structure.

2.4.1 Elasticsearch

Elasticsearch is highly scalable open-source search and analytics engine.it provides a distributed system on top of Apache Lucene for indexing and automatic type guessing and utilizes a JSON based REST API which make it easy to our application to communicate with ES. On top of Lucene functionalities ES adds its own:

- Query Caching
- Real-Time Analytics
- Fully Distributed Model

Easy To Use

It is easy to set up out of the box since it ships with sensible defaults and hides complexity from beginners. It has a short learning curve to grasp the basics so anyone with a bit of efforts can become productive very quickly. It is schema-less, using some defaults to index the data.

One Trick JSON Format

Since ES exposes a REST API to index and search data. its not possible to use other data format such as pdf, xml and csv. So, in order to index these format we need to use third-party plugins.

2.4.2 Apache Solr

Like ES, Apache Solr is an open-source search engine built on top of Apache Lucene. It provides a scalable enterprise wide search capability for a diverse set of data types including: NoSQL, rich document (PDF/Binary/MS-Word), relational database, and more. Features include faceted search, hit highlighting, full-text search. It was designed for scale and fault tolerance.

Data Sources

Solr accept data from different sources including XML files, comma-separated value (CSV) files, and data extracted from tables in a database as well as common file formats such as Microsoft Word and PDF.

Searching

With the variety of analyzers and tokenizers implemented in Apache Solr. It is much more oriented towards full-text search.
While ES is often used for analytical querying, filtering, and grouping.

Cloud Environnement

Apache Solr - in its Elasticsearch-like fully distributed SolrCloud deployment mode - depends on Apache ZooKeeper. Although ZooKeeper is mature and widely used, it's ultimately an entirely separate application. SolrCloud is designed to provide a highly available, fault-tolerant environment for distributing indexed content and query requests across multiple servers. With SolrCloud, data is organized into multiple pieces that can be hosted on multiple machines. The replicas will help to achieve redundancy as well as scalability and fault-tolerance.

2.4.3 Sphinx

Sphinx is open source search engine. Sphinx allow full-text searches. Sphinx is very efficient to perform search over large data. Sphinx can index data from various different sources like: MySQL, HTML, Text Files etc.

Data Sources

From Sphinx point of view, the data it indexes is a set of structured documents, each of which has the same set of fields and attributes. This is similar to SQL, where each row would correspond to a document, and each column to either a field or an attribute.

Distributed Mode

In Order to make Sphinx run in distributed mode. We need to do partitioning by hand, then create special distributed index on some of the Sphinx instances. In distributed mode we can't achieve replication with Sphinx.

2.5 Conclusion

The main aim of this project is to:

- Index Investment Data
- Provide Descriptive Analysis

Then, we must consider analytics and visualizations a core component of our use cases. So we choose Elasticsearch as search and indexing engine for our framework because of his simplicity, quering features and capability to scale.

CHAPTER 3

PROPOSED SOLUTION

3.1 Introduction

This chapter is devoted to present our proposed solution to index data and make an **EDA** as well as implementing the solution in a distributed way.

3.2 Architecture of the solution

3.2.1 Distributed Architecture

Definition

A distributed system is a system whose components are located on different networked computers, which communicate and coordinate their actions by passing messages to one another.
The components interact with one another in order to achieve a common goal.

Benefits

Using a distributed architecture can make our life much easier, because it provides:

- **Scaling**: Distributed Systems are easily scalable.
- **Prallelism**: Distributed Systems are designed for parallelism.
- **Reliability**: Distributed Systems can tolerate hardware failures.

3.2.2 Logical architecture

In this section we present, the logical architecture of our framework to index investment data provided by MAJESTEYE.

- **Zookeeper Ensemble**: 3 Zookeeper instances to synchronize *Solr* nodes.
- **Apache Solr**: 5 Solr nodes used to index data.
- **Hadoop (HDFS)**: 1 NameNode and 2 DataNode used to store index and provide easy scaling.

3.2.3 Docker (Physical Architecture)

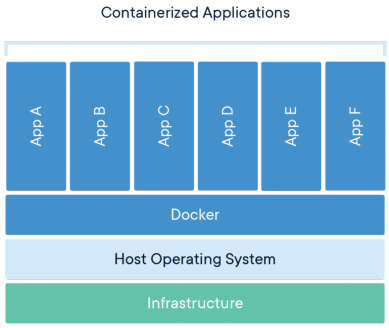
all instances are running under **Docker Engine** to simulate a Mini-Cloud Environnement.

Definition

Docker is a set of platform as a service (PaaS) products that uses OS-level virtualization to deliver software in packages called containers.

Docker Conatiner

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.
A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.



Benefits

Docker Containers are:

- **Standard**: Docker created the industry standard for containers, so they could be portable anywhere.
- **Lightweight**: Containers share the machine's OS system kernel and therefore do not require an OS per application.
- **Secure**: Applications are safer in containers and Docker provides the strongest default isolation capabilities in the industry.

3.3 Software Environnement

3.3.1 Apache Zookeeper

ZooKeeper: Because Coordinating Distributed Systems is a Zoo

Hadoop, Nov. 19, 2012

ZooKeeper is a coordination service for distributed Systems. **SolrCloud** uses ZooKeeper for three critical operations:

- Centralized configuration and storage.
- Detection and notification when the cluster state changes.
- Shard-Leader election.

ZooKeeper is also a distributed system that can be deployed to be fault tolerant and highly scalable. For production, we set up a separate ZooKeeper ensemble made up of at least three nodes, 3 as ZooKeeper relies on a majority of nodes agreeing on the state of a znode at all times.

3.3.2 Apache Solr

As mentioned below we will be using Apache Solr for indexing our investment dataset.

3.3.3 Apache Hadoop

CHAPTER 4

IMPLEMENTATION

CHAPTER 5

EXPLORATORY DATA ANALYSIS