Comparison between Cassandra, CouchDB and MongoDB

Colin Pieper
Minka Firth
Rob Lavrijsen

Fontys Hogescholen
Semester 4 – Data Processing – Module 3
Jan 2022

Introduction

Each group member was tasked with researching and prototyping a big data solution by choosing a different NoSQL database and experimenting with data processing and analytics. In this document, we compare various characteristics and features of our chosen database and exchange our personal experiences from building our prototypes.

Each of us chose a different database: Colin used Cassandra, Minka used MongoDB, and Rob used CouchDB.

Comparison

CAP theorem

According to whatshisface a choice between availability and consistency must be made in the case of a network failure. That means that databases are either Available and Partition Tolerant, or Consistency and partition tolerant.

Cassandra and CouchDB have both chosen for availability over Consistency, while MongoDB is a consistency and partition tolerant database.

Type of database

CouchDB an open-source document-oriented NoSQL database written in Erlang and uses multiple formats and protocols to store, transfer and process its data with JSON files. It uses JavaScript as its query language and features a REST API.

MongoDB is also a document-oriented database. That means that each record is considered a document. They are often either JSON formats, or JSON-like formats. These documents are retrieved using unique keys.

Cassandra is a partitioned row store database. Rows have columns and are stored in tables. Rows are partitioned by the value of one or more columns. The partitions are distributed across the nodes in the cluster.

Use cases

CouchDB

CouchDB is mainly used for its scalability, because of its architectural design. This makes CouchDB really adaptable when partitioning databases and scaling. CouchDB supports horizontal partitioning and replication to create a solution for balancing both read and write loads during a deployment.

MongoDB

Especially suitable when dealing with real-time analytics and high-speed logging, caching and high scalability. It is a useful tool when trying to store many diverse types of data. It is easily horizontally scaled and comes with valuable tool support. Hardly any learning curve necessary for the use of MongoDB, due to easy query language and intuitive UI.

While recently securing more atomicity withing their system, it might still not be a fitting system when dealing with transactional systems, especially when these are multi-document transactions.

Cassandra

Cassandra is mainly used when high performance with large volumes of data is required. It's linearly scalable and features a masterless architecture, which is great for use cases that require high availability. Cassandra features excellent performance for writes and great performance with reads.

However, when frequent data updates and deletions are required, Cassandra is less suited for the job.

In summary, Cassandra is most suitable for applications that require a heavy write workload with high availability. Practical examples are storing sensor data and storing data about user interactions for analytics, such as to feed a recommendations engine and fraud detection.

Personal experiences

CouchDB

My experience with CouchDB is equally good and bad. At start it's very user-friendly and the documentation is on point. But as soon as you installed the database and want to do something more than replicating and store simple JSON files, you need to learn a lot. The learning curve is very steep and the problem if you are a Windows user is that most of the forums are for Linux users. This makes it very difficult for a beginner to start using the database to its full potential. So, in short, CouchDB is equally beginner friendly as not, it depends on what you want to do with it. It has a very nice UI and for a beginner that can be immensely helpful, but if you want to unlock its full potential you have to use command line and preferable Linux OS.

MongoDB

Due to a lot of available documentation and tutorials, installing MongoDB did not cause any problems. The MongoDB UI was a bit rigid in the variety documents of documents that could be imported, but with a combination of the command line, it worked smoothly.

The query language is quite simple and not really verbose. Adding indexes was remarkably easy to do and felt intuitive in the UI. This could not be said for adding Aggregation Pipelines though, or the explanation tab. I prefer to use the command line for the explanation behind the query. Overall, very easy to use, when looking for a shallow look into the basic features of NoSQL, but not as intuitive when looking for a more in-depth experimentation with NoSQL.

Cassandra

Cassandra was easy to install with Docker. Scaling horizontally by adding more nodes was just as easy. Due to its similarities to relational databases, both in table structure and its SQL-like query language, it felt familiar and easy to understand. However, this feeling of familiarity was misleading at times; it made me expect similar behavior as conventional relational databases, but this often resulted in low performance, or more commonly, error messages or warnings.

The documentation is fairly complete and easily accessible. However, I initially had trouble finding documentation and guides at the right level; many were either very high-level overviews with barely any details, many were very technical and detail-oriented, but not much in-between. This however became less and less of an issue the more experience I gained with Cassandra.

Cassandra is focused on what it does best: storing high volumes of data in a distributed fashion and handling high amounts of writes. Features such as aggregations and data pipelines are not present in Cassandra itself but require external software such as Apache Spark. From what I gathered from discussing the features with my group members, CouchDB and MongoDB support more of these additional features natively. However, my experience with Apache Spark was positive; it was easy to connect with Cassandra and offered powerful features.