MongoDB

Background

MongoDB is yet another immensely popular NoSQL database widely used in industry. Compared to an RDBMS, or something with a much different intended use case such as Cassandra, it seems that MongoDB is a very flexible and middle of the road solution, such that developers have more freedom to use both some relational concepts in their data patterns, while still generally trying to keep denormalized data when possible.

Data Format

MongoDB is a document database - documents are stored within collections, and in turn each document itself can have other nested collections.

Famous Actresses

```
{
        Name: Riley Reid
        Age: 30
        Videos:
        [SUBCOLLECTION]
}
```

```
{
        Name: Mia Khalifa
        Age: 29
        Videos:
        [SUBCOLLECTION]
}
```

Note that documents do not follow a set schema: within a collection it is advisable to have generally similar documents but there is nothing enforcing a format, which allows for adjustable schemas over time!

Architectural Overview

- Within a collection, database partitions the documents based off of a shard key per record
- Within each shard, availability is ensured via single leader replication
- MongoDB uses a B-tree based storage engine
  - Prioritizes read speeds over write speeds

Cross Partition Operations

- Mongo enables multi document transactions (either over the same or different shards)
- Mongo enables parallelized queries over the entire data set (over the network), can be sped up via the use of secondary indexes
  - Secondary indexes can be applied even on deeply nested data

MongoDB vs. Relational Databases

In comparison to a relational database, MongoDB exhibits the following properties:

- Flexible schema
- The ability to design an application with data locality in mind via a highly nested document structure
- Lack of normalized data may require having to change data in multiple places

MongoDB vs. Cassandra

In comparison to a relational database, MongoDB exhibits the following properties:

- More expressive document data model
  - Although with a bit of tinkering on a clustering key you can effectively organize Cassandra rows in a subcollection type of ordering
- The ability to query data using complicated secondary indexes, as well as making operations (both writes and queries) spanning multiple shards
- B-Tree architecture is better for read throughput
- Use of single leader replication and B-Tree as opposed to LSM trees and leaderless replication means that write throughput does not scale linearly to the number of nodes in the cluster

Conclusion

Ultimately, it seems like MongoDB is a middle of the road solution - we use it when we want the ability to make complicated queries on data, but have a more flexible and expressive document model. Additionally, MongoDB is useful if we would like to use Cassandra for high write throughput, but ultimately our data access patterns do not allow us to keep all of our operations on just one shard.

In this sense, the architecture of MongoDB does not make it a “specialized” database for any specific type of task, but rather just an extremely flexible and intuitive one that has a large feature set while avoiding the pitfalls of relational data (compared to Cassandra and MySQL which are restrictive in their own ways).