Advanced Database Programming – Comparison of CouchDB and MongoDB

Niall Stack T00174406

2016

# Introduction

For my Advanced Database class I used both CouchDB and MongoDB with class exercises and using them for two separate projects where I created databases in them and created a UI to connect to them. These were the first two NoSQL databases that I used and since they are both Document Stores inevitably a choice between the two will be made so I must compare them and although Mongo is currently the most popular by a wide enough margin that does not necessarily mean it is always the best choice. I have six categories to contrast and compare the two databases user friendliness, querying ability, program language integration, CAP theorem and the database, strengths and weaknesses.

# User Friendliness

User friendliness is typically not too much of a consideration to have in a database as people accessing the database through the command line or through a client are usually technical people who do not need a UI for access. Despite this using some databases can be very tedious, especially working with relational databases.

## CouchDB:

CouchDB comes with a web interface called Futon which makes it much easier to perform basic CRUD operations instead of having to work through the command line the whole time. It is nicely laid out and very intuitive meaning you could create an entire database without having to use the command line.

If using the command line though, CRUD and other operations are made through cURL which uses a very well documented list of commands helping to make Couch even more user friendly. Since Couch uses cURL it is also possible to put the commands through a REST interface such as Postman, giving the user even more options.

The only aspect of CouchDB that could be considered non-user friendly would be that when updating a document not only do you need the \_id number but you also have to have the \_rev number but this is only a very minor thing and so shows how user friendly CouchDB is.

## MongoDB:

MongoDB does not come with any interface and since its commands aren’t in cURL a REST client cannot be used, meaning all operations must be done through the command line which is not very user friendly. An intellij plugin can be used but it is not very intuitive and is nowhere near as simple to use as Futon.

Mongo’s commands are written in JSON (technically BSON) which is also very well documented and popular format to use so the commands are simple to use but they are also somewhat similar in structure to SQL commands meaning moving from SQL to Mongo isn’t as great of a change as SQL to Couch is.

When viewing documents, Mongo has a method called “pretty” which displays the results of a query more clearly which is very user friendly. Both Couch and Mongo use JavaScript for things like map reduces but Mongo also allows functions to be created with JavaScript that allow for things such as making a function to insert documents into a collection. This significantly helps to speed up populating a database which adds to the user friendliness of Mongo.

# Querying Ability

The Querying ability of a database is how it can retrieve its data, essentially one of the main features of a database. So the comparison of Mongo and Couch in this category is very important.

## CouchDB:

Being a document store database, Couch stores its data as documents which consists of key-value pairs. Unlike Mongo there is no ad hoc querying, indexed views produced by incremental mapreduce are the principal way to find documents. The map functions allow the user to create a view and emit any data they want as the key and the values so the key does not always have to be the ID number.

The keys and values can be any valid JSON and views created with map and reduce functions can be saved as views which can then be queried later with cURL commands making it very simple to call even a complex mapreduce function. Requesting a view this way will return the whole set but with Couch it is possible to retrieve a subset of a query by using a key parameter providing even more query options.

Most mapreduce systems like in Mongo throw away the output of mappers and reducers after they have completed their initial query but Couch will keep them until they become invalidated. When they become invalidated from new data in the document Couch will incrementally run the mappers and reducers to correct for this, helping Couch to use mapreduce as its primary indexing mechanism.

## MongoDB:

Similarly to Couch, Mongo also stores its data as key-value pair documents but unlike Couch it can ad hoc querying on nested data. An ad hoc query is query that cannot be determined before it is issued, it is dynamically constructed allowing for faster queries initially than non-ad hoc queries like in Couch.

Mongo’s document are also in JSON but instead of using cURL for query commands it uses JavaScript which means you can’t use a key value in the URL to retrieve a subset of a query. Since Mongo uses JavaScript, it is simple to create complex views helped by being able to do things like construct operations like objects.

Mongo is very efficient at querying nested array data and can search a document and return the results of very deeply nested sub documents, it is even possible to match several fields in a deeply nested sub document. Although not recommended due to slow performance Mongo can also run a decision function across your documents but entire queries can fail since it’s possible for the JavaScript to not properly execute.

To increase query performance Mongo features indexing which is not available in Couch and most other NoSQL databases. It provides several data structures for indexing such as the B-tree which can significantly help to populate very large sets of data quickly. Mongo also has aggregated queries such as “count” or “distinct” which help show case the powerful querying ability of MongoDB.

# Program Language Integration

A database by itself isn’t particularly useful so it must be imbedded in a program or website meaning having good program language integration is vital for any database to be successful.

## CouchDB:

Couch is primarily built with the web in mind, so it can be very difficult to integrate it into some programming languages. There are many different drivers available for Java and C# integration meaning there is no official driver that works perfectly for two of the most popular languages showing poor program language integration, especially in object oriented languages.

One of the few non web languages Couch can work well with is Python but typically large scale applications are not built in Python and its UI support is poor. When searching for help and documentation with Couch online for any languages other than JavaScript there are very few useful results adding to the difficulty of integrating it into a programming language.

The one language where CouchDB works best in is JavaScript which has an official driver and many useful applications can be built with JavaScript. Although it makes sense for CouchDB to work best with a web language seeing as it is built primarily for the web it should still be easier to integrate it into other languages.

## MongoDB:

Program language integration is definitely handled better in Mongo than Couch, Mongo offer official drivers for C, C++, C#, Java, Node.js, Perl, PHP, Python, Motor, Ruby and Scala even supplying official courses for some of them. On the MongoDB website there is also help for community supported drivers for less popular languages such as Go and Erlang.

If you were to try to integrate Couch with Java you are often required to download over 7 .jar files to simply perform basic CRUD operations whereas with Mongo a single .jar file is all that is needed for everything. This shows how well Mongo is able to integrate with programming languages especially compared to Couch.

Overall Mongo is far better than Couch at integrating into programming languages, offering much more official drivers, support and documentation than Couch does for any language.

# CAP Theorem and the Database

The CAP theorem states that it is impossible or difficult for a database to simultaneously provide Consistency (Everyone sees the same data), Availability (Everyone can access some version of the database) and Partition Tolerance (The database can be split up over multiple locations). So a database must choose two of the guarantees which help to determine which database is suitable for different types of systems.

## CouchDB:

The two guarantees that Couch has are, Availability and Partition Tolerance:

* Availability – Couch supports master-master replication so data can be written to one node of the database without having to wait for other nodes to come into agreement. This means that the database is highly available and can scale very well but two clients could see different data. In case of conflicts CouchDB uses a deterministic algorithm to choose a winner and then it forces that on any conflicting nodes.
* Partition Tolerance – This master-master replication also helps to guarantee partition tolerance as it supports the replication of data across different servers.

## MongoDB:

The two guarantees that Mongo has are Consistency and Partition Tolerance:

* Consistency – Mongo has replica sets that have one node as master and the others as secondary to it, the data from the master is duplicated to the others this helps to add redundancy in case the master goes offline. When the master goes offline one of the secondary servers becomes the master which is decided by which one has the freshest data.
* Partition Tolerance – Like Couch, Mongo’s partition tolerance comes from its replication as for replication to work the data must be stored across multiple servers.

# Strengths

## CouchDB:

## MongoDB:

# Weaknesses

## CouchDB:

## MongoDB:

# Conclusions