

Comparison between MongoDB and CouchDB

Naga Sai Prithvi Raj Mamidala
Dept. of Electrical and Computer Engineering
nagasaiprithviraj@unm.edu

Sai Aditya Thalluri
Dept. of Computer Science
aditya369thalluri@unm.edu

ABSTRACT

NoSQL (Not only SQL) is a database used to store large amounts of data. NoSQL databases are non-relational, distributed, open source and are horizontally scalable models. Relational databases are not efficient when it comes to managing huge data. In this paper, we are studying about NoSQL document-oriented databases MongoDB and CouchDB and their comparison along with features like ACID, BASE and CAP theorem. Their comparison with relational shows the various advantages of NoSQL databases.

KEYWORDS

durability, concurrency, redundancy, consistency, fault-tolerant, replication, scalability, JSON, BSON, XML.

1 INTRODUCTION

With the increasing data generated around the world, there developed many ways to efficiently store the data and retrieve fast. Applications over the internet prefer to store the data in databases in such a way that there should be less inter-processing and access must be quick. With the traditional relational databases, data need to modeled according to schema into multiple tables before storing it. This process consumes time as it needs to convert between the database schema and user application pattern. Also, it is difficult for relational databases to handle huge data. This led to the development of NoSQL databases. NoSQL databases in contrast to SQL, follow denormalization and redundancy of data. They don't care about the amount of space and the goal is to make huge data storage and access faster. They won't follow any schema or model while storing the data, uses the dynamic schema of the data instead.

In this paper, it is purported to compare the behavior of NoSQL document-oriented databases. There are many document-based NoSQL databases, but all differ in mechanisms to store data in document format. Two most significant document-oriented databases MongoDB and CouchDB are considered. Throughout the paper starting with an overview of NoSQL databases, we will be first describing the two databases separately about their characteristics and comparing them with the relational database. Then we will compare the two databases in various sections such as the features they provide, performance, security, etc. This will help to deduce the merits and demerits of each database

2 BACKGROUND

NoSQL means not only SQL. It is another data storage, used to store large amounts of data like on Facebook (in which data increases day to day). The greatest advantage of NoSQL databases is they are schema free. It is a non-relational database, quick data recovery and is compact. Here data is stored in different elements and is managed by different elements, so the concept of replication is used.



Fig 2.1: Symbolic representation of NoSQL.

It can handle huge amounts of data, helpful in data warehousing. In SQL, uses Query language to fetch data; for NoSQL, we store huge data entities using documents in XML (extensible markup language) formats. XML language is essentially used to store organized information in a comprehensible structure.

Within NoSQL, databases are classified on how they store the data [1]. They are described below.

Key-Value pair: Entire database is stored as key-value pairs. Some of them are Amazon DynamoDB, Redis, etc.

Document Store: These can be described as a group of key-value stores. Each document is equivalent to a tuple in the relational database. Examples are MongoDB, CouchDB, etc.

Column family: In these databases, data is structured into columns. Data is organized into column, super-column and column family. Examples are HBase, Cassandra, etc.

Graph Database: These are used in network connections to represent graphs such as social networks. Examples are Neo4j used by Facebook, Info Grid, etc.

3 COUCH DB

CouchDB is an open source NoSQL database developed by the Apache software foundation. It is a document-oriented NoSQL database architecture with “documents” as key-value which stores complex data and the unique key is assigned to each document. It mainly focuses on having a scalable architecture and ease of use. Concurrency-oriented language; Erlang is used to implement CouchDB, it uses JSON to store data, JavaScript as query language using MapReduce, and HTTP protocol for an API to access the documents and implement queries in a web browser. It is a multi-master application invented in 2005 and it turned into an apache venture in 2008.

3.1 Data Model

Couch database is a collection of records called documents. Each document keeps up its own information and independent composition.

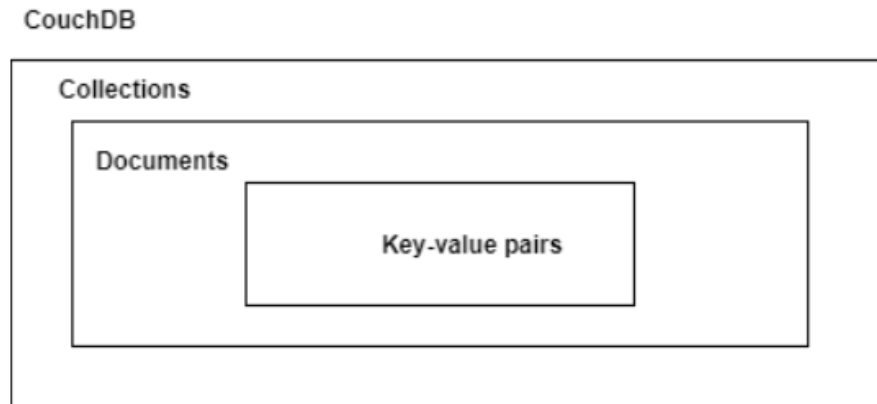


Fig 3.1: Figure showing simple representation of CouchDB

The database is the outermost information structure/container in CouchDB, where each database is a collection of documents. Each document has its own self-contained schema and its own data. Document metadata contains modification data, making it conceivable to combine any distinctions that may have happened while the databases were disconnected. CouchDB implements MVCC (multi-version concurrency control), which helps it to avoid locking of database files during writes[7]. MVCC is a concurrency control method used by database management systems, which provides concurrent access to database and programming languages.

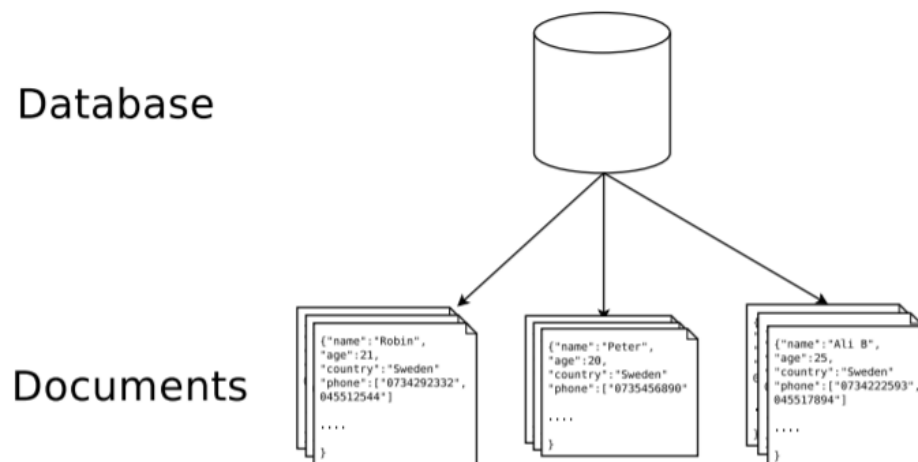


Fig 3.2: CouchDB's data model showing database and documents [7].

3.2 Architecture

It uses HTTP based REST API to communicate with the database easily. It uses HTTP resources and methods like GET, PUT, delete that are easy to understand and to implement also. It facilitates users with powerful data mapping, allowing querying, combining and filtering of data information. It provides synchronization of data between machines and databases, copying of data, easy replication and easy to use. Structure of data is safe as data is stored in a flexible document structure.

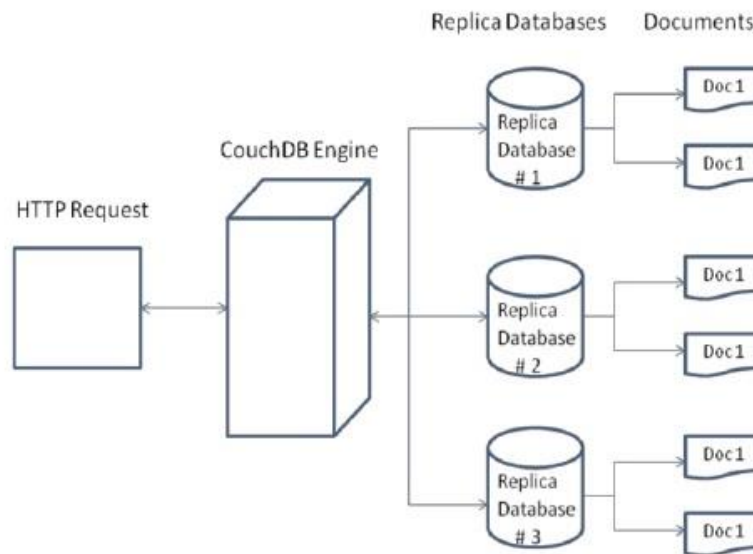


Fig 3.3: Representation of CouchDB Architecture [7].

3.3 Features

CouchDB follows ACID (Atomicity, Consistency, Isolation, Durability), a set of properties of database transactions. It does this from MVCC (multi-version concurrency control), means it can handle dense concurrent readers without a conflict[7].

Atomicity: Document update follow this property, updates like add, edit, delete follow atomicity. This database does not have any partially saves or edited documents, they will either saved completely or not saved at all.

Consistency: In CouchDB, once the data is committed it cannot be overwritten or modified, by ensuring that the document is always in a consistent state. CouchDB rad uses MVCC model, because of which a user can always see a consistent state of database from start to end of a read operation.

Isolation: When a document is updated, CouchDB sends the data onto a disk and updated database header is written in two identical ad consecutive pieces to make up the first 4k of the file, and then synchronously flushes onto disk. So partial updates during flush will be discarded.

Durability: If there is a failure after write operations have completed, it ensures that the effects of the write are visible upon recovery. Databases that support ACID transactions will always provide durable writes. In CouchDB, durable writes are supported. If it fails while committing the header, a copy of the previous identical header will remain. Ensuring coherency of all previous data.

Compaction: If there is wastage in space above a certain limit in a database, all the activity is copied or cloned to a new file. After all the process is done, then the old file is discarded. The database remains online only during compaction and all updates, reads and other operations are completed successfully.

Views: Documents in CouchDB are semi-structured that are flexible with implicit structure. It is a simple way of data storage and sharing. If data must be seen in different ways, it must be filtered, organized and

be split into tables. So, CouchDB provides a view model to tackle this problem, where views are a method of summing up and reporting on documents. These are built on request of aggregate, join.

Document Storage: Documents are the main primary unit of data where each field is uniquely named and has values of various data types like text, int, Boolean, char, etc. There is no limit in documents.

Security: It provides database level security. The permissions are separated based on admin and readers. Readers can do both read and write operations.

Validation: Inserted data can be validated by combining with authentication, ensuring the creator of the document is a logged in user.

Replication: CouchDB follows the simplest method of replication compared to other databases.

Browser-based GUI: It has interface futon, which uses a browser-based GUI to handle data, permission, and configuration.

Authentication and session support: CouchDB keeps authentication open via web application like session cookie.

Built for offline: It can replicate to devices, can be used in offline and handle data sync when the data is back online..

3.4 RESTful API

It is an architecture which describes how services can be provided machine to machine communications. HTTP is used to perform CRUD (create, Read, Update, Delete) operations[7].

POST: Create a resource.

GET: Read a resource.

POST: Update a resource.

DELETE: Delete a resource.

In restful architecture, gets unique identifiers in the form of URI's. CouchDB uses curl application that makes the user to perform raw HTTP.

```
curl -X PUT http://localhost:1234/database
```

REST also makes developers to send information in XML or JSON. Requests made by curl are answered by JSON.

```
{"ok": true}
```

Requests using curl can be used to achieve CRUD operations. Because Web browsers use HTTP, they can also be used to read JSON documents from CouchDB.

3.5 Scalability and Replication

Replication of databases in CouchDB is an easy task. All it takes one simple HTTP request which specifies the source database and target database.

POST / replicate HTTP/1.0 { "source":"employee_database", "target":"http://anything.com/db" }

This can also be done from a remote server to the local server by switching the value of target and source keys. Using web interface futon, replication can be done more easily. Scaling in CouchDB can be done by an application called CouchDB Lounge, in which partitioning, and clustering framework is done. Scaling can also be done using BigCouch.

3.6 Querying

CouchDB uses JSON documents as querying language, as the data is dynamic[7]. Querying in Couch is done using views. There are two kind of views, i) permanent view, in which data is static. ii) temporary view, in which results are reduce functions. Map functions are described by a user and iterate over all the document to check if it matches the criteria specified in the function. If everything matches, result is found. Documents are extracted using emit ().

```
function(doc)
{ if(doc.salary && doc.salary > 15000 && doc.name)
emit(doc.name,doc.salary); }
```

In above example, all documents that have key salary grater than 15000 are emitted. After emitting a list of documents by map function, a reduce function is used to further operate on data.

3.7 Comparing CouchDB and RDBMS

CouchDB has some advantages and key characteristics over Relational[9].

CouchDB	Relational Database (RDBMS)
It contains denormalized data. Data can be structured, semi or unstructured.	Data must be in normalized form.
Built for situations with high speed data networking.	Built while networking still developmental and slow.
Built on reality of inexpensive and huge memory.	Built on assumption, memory is expensive and limited.
Data is flexibly stored as JSON documents or binary data. No need to predefine datatypes.	Data types must be predefined for attributes.
Scalable to millions of users easily.	Scalable to an extent to limited users.
jQuery is used to implement queries.	SQL is used to implement queries.

Highly optimized for retrieve and append operations, can handle large number of documents.	Slowly times for retrieving a committing data compared to NoSQL.
Does not require data modelling, schema designing.	Schema designing is an important task.
Data stored as a key document pair.	Data stored in tables according to the relations, attributes and their data type.
Asynchronous operations and concurrency are responsible for high performance.	Data integrity and normalizations implies slow response and low performance.

Table 3.1: Difference between CouchDB and RDBMS.

4 MONGO DB

MongoDB is the most used NoSQL database of the current time. It is opensource and is developed using C++ programming language in 2007 by an organization “10gen” in New York which is then officially called “MongoDB Inc”. The word “Mongo” is derived from the word “Humongous” which means huge or enormous. It follows a document a document-oriented structure which is similar to JSON (JavaScript Object Notation) or XML [2]. There is no schema or document structure used by MongoDB or any NoSQL databased and it is usually called as Dynamic schema..

4.1 Data Storage Structure

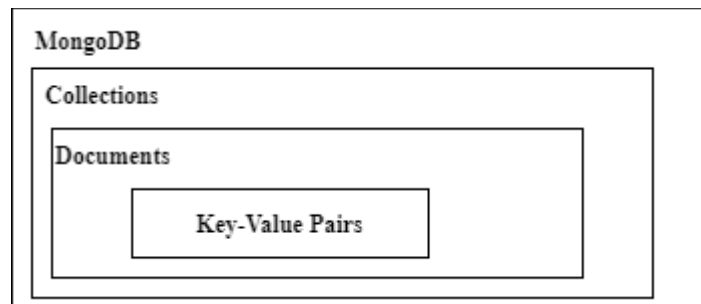


Fig 4.1: Representation of MongoDB Storage Structure

MongoDB arranges data records in the form of Documents. Document store databases can be called as a group of key-value store databases. A document is equivalent to a tuple in the relational database. Fields inside a document are key-value pairs. Document do not need to have the same set of key-value pairs or any standard schema. Having a standard database schema makes it very difficult for the users. With the huge increase in usage of agile development model by software organizations where there will be continuous changes in the process of software development according to customers, it will be inconvenient to software developers to keep on updating the data based on the schema. Instead, having a database which

will take the data from the application and store it as it makes their work simple. There will no pre-processing required while storing or retrieving the data due to the database schema. Also, it is easier to handle large data with the dynamic schema as they are not required to convert to any standard structures.

Each document can have different fields and it depends on the application using the database. Key in key-value pair is the name of the field. Value is the corresponding value of the key field. MongoDB supports various datatypes for the key value such as string, integer, boolean, date, timestamp, Object, Null, etc. So the document structure is dynamic and does not follow any standard schema. The document is stored internally in the form of BSON (Binary JSON) and the size a document is limited to 16 MegaBytes. A group of related documents is stored in a collection. A collection is similar to a table in the relational database. Although it is not mandatory to have all similar documents within the same collection, it is usually followed to keep all the similar document in the same collection.

While inserting or updating data into the documents, no structuring or formatting needs to be followed. Whatever the data can be inserted or updated in documents. Data within a document does not need to be key-value pairs all the time. It can be also key-array pairs or any other depending on the type of value. Value is a complex data structure and can be an array, list, object, etc. Documents can also be inserted inside other documents which are called nested documents. Depending on the usage, the corresponding type for the value and MongoDB supports many kinds of structures for the value.

Every database will have its own way to identify documents or tuples. When it comes to MongoDB, there will be a field `_id` in each document which will be always unique. It will be inserted along all other fields while creating a document. If users don't provide the `_id` field or its value while inserting a new document, then MongoDB itself will include the `_id` field with a unique ObjectId for each document. ObjectId is a 12-byte hexadecimal number unique for each document and is divided in such a way that 4 bytes represent the timestamp, 3 bytes for machine id, 2 bytes for process id and the remaining 3 bytes will be incrementor.

When it comes to reading operations indexing is important in database. Since `_id` is a unique field in each document, MongoDB creates a default index on it. We can also choose multiple fields as indexes while inserting data. It is called as "Compound Index". Indexes in MongoDB follow B tree data structure to organize data [2]. Even though there are multiple indexes, the query optimizer uses only one index which is efficient compared to all other while processing the queries to retrieve the data.

4.2 Architecture

MongoDB uses Master-Slave architecture to maintain replicas of data. A slave will have a copy of data contained in the master. While master and slave both are Mongo database servers, a master can have multiple slaves. The master node is always primary is used for both read and write operations. Slaves are secondary nodes and are usually used for backup. When the master is down then the slave with most recent data will serve as the master. Also, when there are large data on the master, then it will be shared on to the slaves and can be used for read operations.

There can be various types in which the secondary nodes can be used. One of them is where secondary can only be a member used for backup and cannot become primary in any conditions. Also, the secondary nodes set election among themselves whenever a primary node is down. Based on the proper availability of data, one of the secondary nodes is elected as primary by others. The election between the secondary nodes is

also called a heartbeat. It will be difficult to elect when there is even number of secondary nodes. In this case

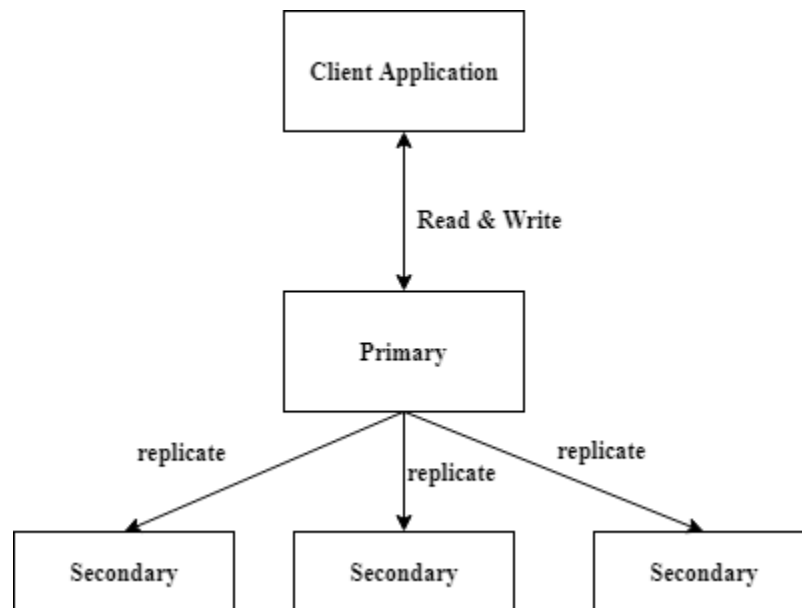


Fig 4.2: Representation of MongoDB Architecture

a node named Arbiter can be added which is no purpose other than voting. Arbiter does not any hardware and it will not serve as any backup. There can be also nodes which are secondary but cannot vote during an election for primary. Based on the need and purpose, database administrators will choose the types of secondary nodes from various kinds available [2].

There can be various types in which the secondary nodes can be used. One of them is where secondary can only be a member used for backup and cannot become primary in any conditions. Also, the secondary nodes set election among themselves whenever a primary node is down. Based on the proper availability of data, one of the secondary nodes is elected as primary by others. The election between the secondary nodes is also called a heartbeat. It will be difficult to elect when there is even a number of secondary nodes. In this case, a node named Arbiter can be added which is no purpose other than voting. Arbiter does not any hardware and it will not serve as any backup. There can be also nodes which are secondary but cannot vote during an election for primary. Based on the need and purpose, database administrators will choose the types of secondary nodes from various kinds available.

Secondary nodes also keep performing the operations from the primary node asynchronously. It will ensure the availability of the database even in case of multiple node failures. There will be some threshold time span where the primary node has to connect to the secondary nodes at least once. If it doesn't, then it is considered as failure and the secondary nodes will elect node from among themselves as primary. Secondary nodes can be distributed in various geographical locations to overcome regional failure and to maintain data localization.

4.3 Features

Durability and Concurrency are the most important characteristics of MongoDB [5]. Durability is the availability of the database in stages of failure and also the ability to save operations permanently after they

are committed. Durability is achieved through replication of nodes through master-slave configuration as explained above. To maintain concurrency, MongoDB uses multi-granularity locking. Based on the requirement, an operation can lock a database or collection or a document. It also uses the reader and writer latch. A latch is multi-reader, single writer and is writer greedy. Multi-reader specifies there can be any number of read operations at a time. Single writer specifies that there can be only one write operation on a document at any time. Write greedy means that when they're read and write operations to be performed at the same time, write operation will be given priority to maintaining consistency. This way MongoDB maintains concurrency.

To overcome the problem of handling huge data, NoSQL databases follow horizontal scaling of data known as “Sharding”. It is basically spreading the data across various servers which reduces the burden of maintaining on the same server. MongoDB provides Auto-Sharding of data which makes sure that data is evenly spread across servers on its own [2]. It also distributes the query load to the multiple shards and even in case of failures, replication happens very quickly to maintain the availability.

Sharding is made very easy using cloud computing tools such as Amazon web services. It is very easy to maintain server instances through cloud tools. Below diagram shows various shards of a mongo database on which the data is spread across. All these are configured using a configuration server which is generally a cloud software. Client applications will connect to these servers.

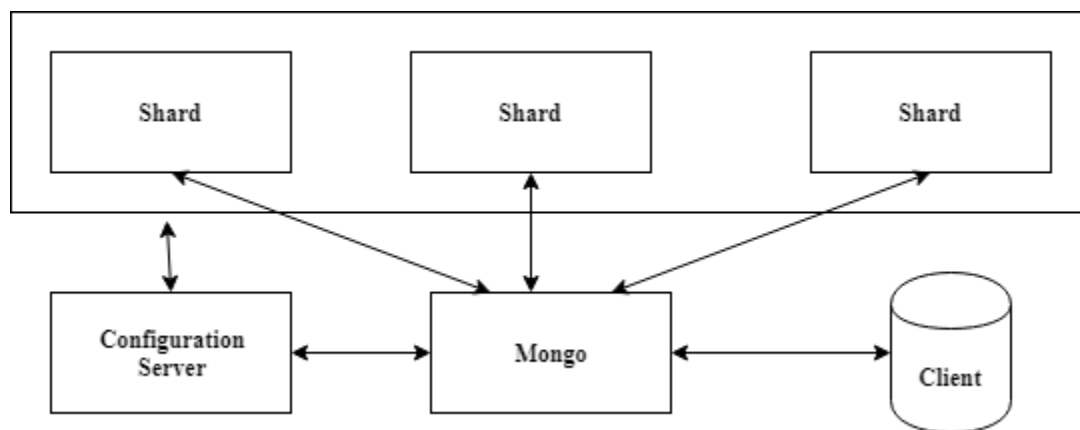


Fig 4.3: MongoDB Server distribution using Shards

MongoDB, like many other NoSQL databases, maintains integrated caching. Caching in databases is very useful when there are a lot of read operations. Even in the case of distributed servers, most used data from all the servers will be collected and stored in the system cache. This is what makes integrated caching. It reduces a lot of time delay for read operations. But when there are more write operations, caching is not going to help as expected due to frequent change in data.

Since it stores data in the form of objects, Mongo also provides validation mechanisms at the collection level. While creating a collection, we can specify the validation using the validator option. A validator has additional attributes such as validationLevel and validationAction. Option validationLevel determines on which operations the validation should as read, insert and update. Its value can be strict or moderate.

ValidationAction is the corresponding action needs to be performed when the validation fails in any of the cases.

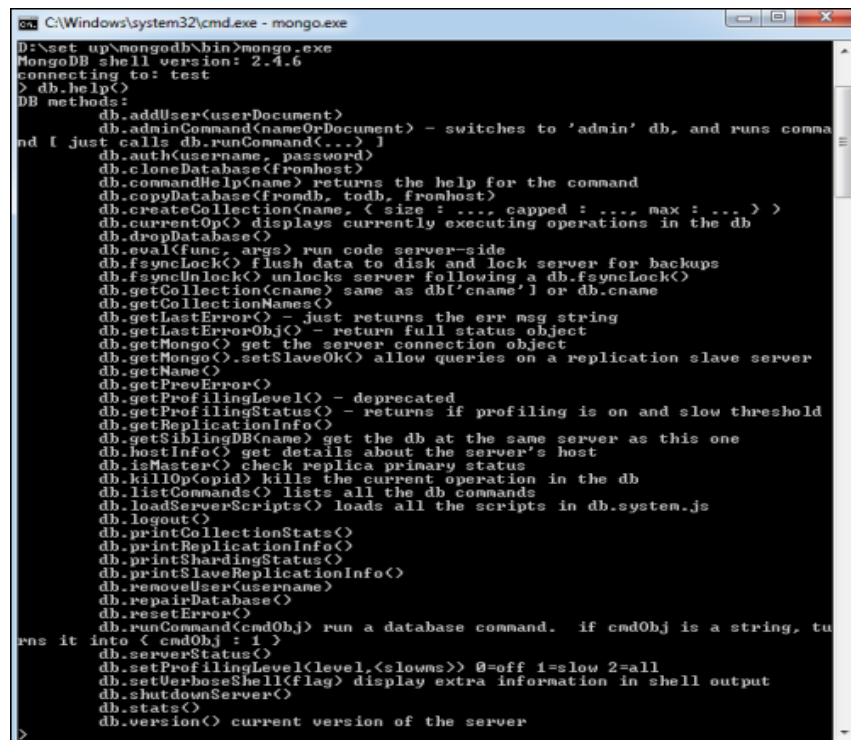
Apart from these, there are many other features if MongoDB. It provides an aggregation framework through which the Map-reduce model can be applied for batch processing. It also allows to search data using regular expressions which are called Adhoc queries. Capped collections are the collections of fixed size. Mongo Management Service (MMS) is a tool which helps to track all the database along with their machines and also to back up the data during recovery times.

4.4 Client and Console:

“mongod” is the primary process and client for the MongoDB system. It will handle data requests, access operations and also management process in the background. The four major operations of database CRUD which re Create, Read, Update and delete can be executed over MongoDB. It also supports usage of cursors with many options through users can iterate over documents or collections. The most recent stable release of MongoDB is version 4.0. It includes features for hosting database clusters and their automatic upgrade. From version 3.4, it no longer supports machines with 32-bit operating system configuration. The major limitation for 32-bit versions of Mongo is that the maximum size of the database cannot exceed 2GB.

After proper install and configuration of the path run the mongo shell. We can also change the default hostname and port using “- -host” option. MongoDB by default creates a database named “test” after installation in which collections and documents can be inserted directly. Since it is developed in C++, it can run on any operating system and is cross-platform.

Below diagram shows the console of MongoDB after running it and default database “test” available on installation. Using the command help, all the available operations on the database can be listed and selected according to requirement and usage.



```
C:\Windows\system32\cmd.exe - mongo.exe
D:\set up\mongodb\bin>mongo.exe
MongoDB shell version: 2.4.6
connecting to: test
> db.help()
DB methods:
  db.addUser(username, password)
  db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command I just calls db.runCommand(...)
  db.auth(username, password)
  db.cloneDatabase(fromhost)
  db.commandHelp(name) returns the help for the command
  db.copyDatabase(fromdb, todb, fromhost)
  db.createCollection(name, { size : ..., capped : ..., max : ... })
  db.currentOp() displays currently executing operations in the db
  db.dropDatabase()
  db.eval(func, args) run code server-side
  db.fsyncLock() flush data to disk and lock server for backups
  db.fsyncUnlock() unlocks server following a db.fsyncLock()
  db.getCollectionName() same as db['cname'] or db.cname
  db.getCollectionNames()
  db.getLastErrorMessage() - just returns the err msg string
  db.getLastStatusObj() - return full status object
  db.getMongo() get the server connection object
  db.getMongo().setSlaveOk() allow queries on a replication slave server
  db.getName()
  db.getPrevError()
  db.getProfilingLevel() - deprecated
  db.getProfilingStatus() - returns if profiling is on and slow threshold
  db.getReplicationInfo()
  db.getSiblingDB(name) get the db at the same server as this one
  db.hostInfo() get details about the server's host
  db.isMaster() check replica primary status
  db.killOp(opid) kills the current operation in the db
  db.listCommands() lists all the db commands
  db.loadServerScripts() loads all the scripts in db.system.js
  db.logout()
  db.printCollectionStats()
  db.printReplicationInfo()
  db.printShardingStatus()
  db.printSlaveReplicationInfo()
  db.removeUser(username)
  db.repairDatabase()
  db.resetError()
  db.runCommand(cmdObj) run a database command. if cmdObj is a string, turn it into { cmdObj : 1 }
  db.serverStatus()
  db.setProfilingLevel(level, {slowms}) 0=off 1=slow 2=all
  db.setVerboseShell(flag) display extra information in shell output
  db.shutdownServer()
  db.stats()
  db.version() current version of the server
>
```

Fig 4.4: MongoDB Console and Client

4.5 Querying MongoDB

MongoDB has a powerful query language developed which can support almost all of the functions of single table in relational database [2]. It also has the map function which can specified in the query to reduce the search time. Let's say there is a collection named students and we need to fetch all the student from computer science having more than 3.5 gpa.

```
db.Student.find({department:'CS',grade:{$gt 3.5}})
```

As we can see above, query is made so simple in mongo DB almost all the features in relational. Since mongo uses indexing on the data, its access speed is very fast compared to other NoSQL databases.

4.6 Comparing MongoDB with Relational Database

MongoDB differs a lot with relational databases in many ways. Below table shows the differences between two databases [3,6].

MongoDB	Relational Database (RDBMS)
It is contrast to normalization and follows denormalization of data.	Follows normalization of data.
Generally used for high speed access of data	Generally used to maintain consistent data and it flexible SQL even though it is slow.
Doesn't care about redundancy of data and memory space usage.	Maintains as much redundancy for data and tries to reduce the memory usage
Data internally stored as Binary JSON which BSON format. Supports many complex data structures of the key values.	Support some standard datatypes for the field values and
Access is unlimited to number of users at a time.	Has limitation on the maximum number of users on database at a time.
It has powerful query language which can handle almost kinds of operations like in relational.	Relational is most used for the querying and any requirement can be achieved with queries.
Latency is low for read/write operations even for huge number of documents.	Latency is high as the data operation involve multiple tables and joins.
No schema to be pre-defined and everything is dynamic.	Each relation will have its own schema and records in it need to follow that to be inserted.

Data stored as a key value pair in documents.	Data stored as records in tables according to the relation schema, attributes and their data type.
Durability and concurrency are responsible for high performance.	Consistent and reliable due to ACID properties.

Table 4.1: Difference between MongoDB and RDBMS.

5 COMPARING MongoDB AND CouchDB

5.1 Feature Comparison

A better way to compare both the databases is to differentiate them among the major features of NoSQL databases. Features such as Replication, Concurrency, CAP, etc of NoSQL databases can be considered to this and figure out which database suits for which feature. Below table shows the clear distinction between MongoDB and CouchDB for the selected features[8].

	MongoDB	CouchDB
Development language	C++	Erlang and C
Storage Type	BSON	JSON
Protocol	TCP/IP	HTTP
Transactions	No	Yes, but only on a single document
Concurrency	Multi granularity	MVCC
Locks	Yes	No
Triggers	No	No
Replication	Master-Slave pattern	Master-Master pattern
CAP theorem	Consistent, Available and Partition tolerant	Eventually Consistent, Available and Partition tolerant.

Operating Systems	Linux/MacOS/Windows	Linux/MacOS/Windows
Data storage	Disc	Disc
Characteristics	Indexing, replication, Adhoc-queries, validation, etc	ACID, replication, validation, etc.
Areas of use	CMS and Web applications	Common applications

Table 5.1: Feature comparison between MongoDB and CouchDB

Even though some of the features are common, there many differences between both the databases. Both are document databases which are similar to JSON, but mongo stores the documents in Binary JSON format. MongoDB uses multi granularity to maintain concurrency whereas CouchDB uses multi-version concurrency control (MVCC) where maintains multiple versions of the same document during write operations and doesn't any locks to perform operations. None of the read operations in CouchDB need to wait when there is write operation going on the same document. The protocol used to connect to MongoDB is TCP/IP which is connecting to a server with a mongo database installed on it. Whereas when it connection for CouchDB, HTTP protocol is used. Restful API's are used to connect to CouchDB which requires the protocol to be HTTP[8].

Replication of data in MongoDB is achieved through master-slave pattern, whereas in CouchDB it is a multi-master pattern. All the nodes in the CouchDB are primary and they all share the data among themselves to reduce the burden on a single server. MongoDB is consistent from the beginning. In contrast, CouchDB will not be at the beginning but will be eventually consistent. Even though most of the NoSQL databases don't support the acid properties of the relational databases, CouchDB supports them. Both the applications are used rigorously in web applications due to their JSON document storage structure which makes reduces the load on application developers.

5.2 Performance Comparison

Another better method to compare MongoDB and CouchDB is to compare the performance and latency of both the databases in Read/Write operations among different sizes of records. Along with type operation, latency also depends on the number of documents accessed by the operation.

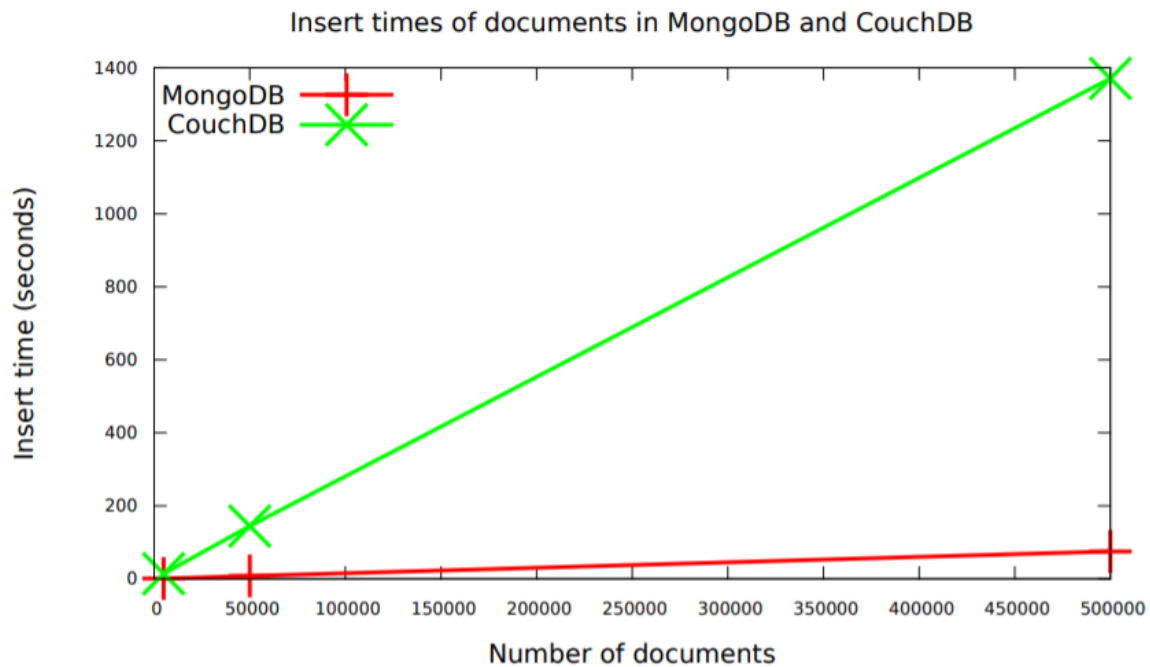


Fig 5.1: Insert time comparison MongoDB and CouchDB [4]

Above image shows the time is taken by MongoDB and CouchDB while inserting documents into their databases. Multiple documents are inserted generally using batch inserts. MongoDB takes very less time for insertion till the first 50000 records. After that, the time increases, but still, it's very less compared to the multiplication ratio of a number of documents. Whereas in CouchDB, the time for inserting documents increases continuously with documents. So, MongoDB is very quick due in write operations. The reason for this fast operation is sharding and replication architecture in the mongo database. When there are huge write operations, they will be shared among the shards which are horizontal scaling of data. Each shard will complete the requests shared and data will get updated synchronously.

The graph plotted above shows the read operation times with respect to the number of documents between MongoDB and CouchDB. Due to the master-slave replication pattern, all the read operations in MongoDB can be carried through secondary nodes. All the secondary will have the same data as the primary node. They can be used for backup or read operations as required. It reduces the overload on the master node, and it takes very less time for the read operations. The ratio of increase in time for read operations with respect to the number of documents is very less. Since CouchDB uses the master-master architecture, both the write and read operations has to be performed on the master servers which makes it slow even for read operations[8]. So, the ratio of time taken for read operations with respect to the number of documents keeps increasing continuously.

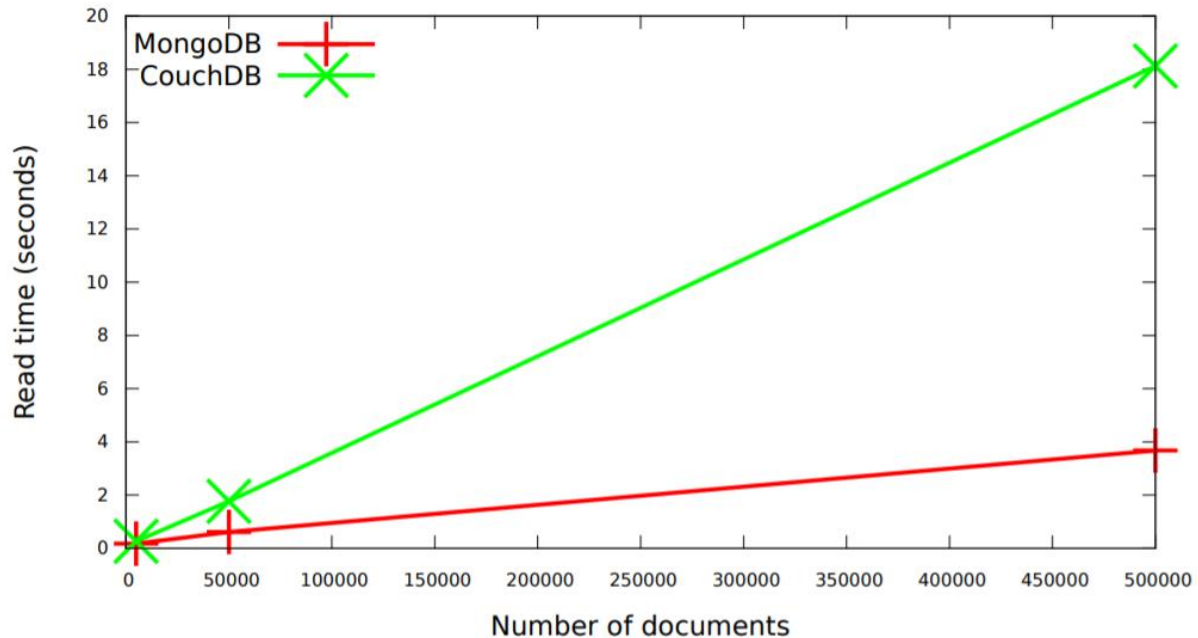


Fig 5.2: Insert time comparison MongoDB and CouchDB [4]

5.3 Querying and Connectivity Comparison

MongoDB has a very powerful query language. It provides all the functionalities like on a single table in a relational database. It also uses indexing with the B tree data structure which makes the data retrieval very easy and faster. Since the operations on CouchDB has to be done through API, it has limitations on the query language and not as much powerful as mongo. CouchDB doesn't use any indexes on data too. It uses views which are written in JavaScript to index and search data. Views will be sent along with the API request in HTTP protocol to get the required data. Like all other NoSQL databases, MongoDB and CouchDB use the Map-reduce algorithms to search for data efficiently.

Connection to MongoDB is simple with TCP/IP Protocol. CouchDB provides interface only based on HTTP-REST and it is another limitation on connectivity. Additional securities need to be implemented when using HTTP services which is another overload for the application developers. Concurrency Read and write operations is not ideal as it does not use any locking system[8].

5.4 Security Comparison

With the increasing security incidents around the globe, one should follow proper security measures on the software they use or develop. It applies for databases too. Let's compare the security and features of MongoDB and CouchDB[10].

	MongoDB	CouchDB
Authentication	Provides authentication using the usernames and passwords	Maintains a special database named authentication database which the stores users
Authorization	Uses role bases access control to give appropriate access to users	Users are divided into two categories member and admins. They will be given the corresponding access
API level security	Provides security for REST API operations but only for enterprise customers.	Since it relies completely on the Rest API, uses cookie and many other authentications by default.
Additional	It also provides TLS/SSL (Transport Layer Security) security to encode and decode the communications.	As it is based on HTTP, provides security for various internet attacks by hashing the passwords, etc

Table 5.2: Security comparison MongoDB and CouchDB

5.5 Cost Comparison:

Any NoSQL database has its community editions available for an individual to work on. But not all of them are free of cost for business applications. MongoDB is more costly than any other NoSQL databases. It cores costs \$6500 per server per year whereas the advanced version costs \$10000 per server in a year. When it comes to CouchDB, since it is built on top of Apache it. Users whoever has Apache licenses can use all the products of Apache without any further cost. So CouchDB is free of cost for the Apache users.

6 CONCLUSION

MongoDB is a lot better than CouchDB in many aspects. According to comparisons done above, it provides many features than CouchDB. It's read/write operations are so quick and take very less time even for a large number of documents. All the aspects of CAP theorem are followed by MongoDB. The query language of MongoDB is very simple to learn but also very powerful. Very strong security measures are also provided by MongoDB to its users. The replication architecture and backup mechanisms of MongoDB are strong and reliable compared to CouchDB.

MongoDB is more preferred when operations are supposed to be very fast. CouchDB even though a NoSQL database, supports the ACID properties. It is preferable for Restful API architecture. CouchDB uses Views which are written in JavaScript to query the database. CouchDB is suitable for a more robust and fault-tolerant database requirements. The reason for the slowness of CouchDB is that it performs operations on the documents one by one, whereas MongoDB does batch processing which makes it a lot faster. With all these factors, we can consider MongoDB as a better NoSQL database then CouchDB..

REFERENCES

- [1] Han, Jing, et al. "Survey on NoSQL database." *2011 6th international conference on pervasive computing and applications*. IEEE, 2011.
- [2] Membrey, Peter, Eelco Plugge, and DUPTim Hawkins. *The definitive guide to MongoDB: the noSQL database for cloud and desktop computing*. Apress, 2011.
- [3] Boicea, Alexandru, Florin Radulescu, and Laura Ioana Agapin. "MongoDB vs Oracle--database comparison." *2012 third international conference on emerging intelligent data and web technologies*. IEEE, 2012.
- [4] Henricsson, Robin. "Document Oriented NoSQL Databases: A comparison of performance in MongoDB and CouchDB using a Python interface." (2011).
- [5] Abramova, Veronika, and Jorge Bernardino. "NoSQL databases: MongoDB vs cassandra." *Proceedings of the international C* conference on computer science and software engineering*. ACM, 2013.
- [6] Parker, Zachary, Scott Poe, and Susan V. Vrbsky. "Comparing nosql mongodb to an sql db." *Proceedings of the 51st ACM Southeast Conference*. ACM, 2013.
- [7] Anderson, J. Chris, Jan Lehnardt, and Noah Slater. CouchDB: The Definitive Guide: Time to Relax. "O'Reilly Media, Inc.", 2010.
- [8] Bhardwaj, Niteshwar Datt. "Comparative study of couchdb and mongodb--nosql document oriented databases." *International Journal of Computer Applications* 136.3 (2016): 24-26.
- [9] Padhy, Rabi Prasad, Manas Ranjan Patra, and Suresh Chandra Satapathy. "RDBMS to NoSQL: reviewing some next-generation non-relational database's." *International Journal of Advanced Engineering Science and Technologies* 11.1 (2011): 15-30.
- [10] Okman, Lior, et al. "Security issues in nosql databases." *2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2011.