DOI: http://dx.doi.org/10.26483/ijarcs.v9i2.5901

Volume 9, No. 2, March-April 2018

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

IMPLEMENTATION OF MAP-REDUCE PARADIGM IN MONGODB AND COUCHDB

Subita Kumari
CSE Deptt., UIET
Maharshi Dayanand University
Rohtak, India

Abstract: The most famous NoSQL document-oriented databases namely CouchDB and MongoDB have been discussed in this paper. CAP theorem is being discussed for MongoDB and CouchDB. Map-Reduce is a parallel and distributive programming paradigm for processing bulk amount of heterogeneous and unstructured data on clusters of computers. Map-Reduce operation has been implemented in MongoDB and CouchDB. MongoDB uses map-reduce to perform aggregation. CouchDB uses map-reduce for querying and implementing views. The paper also presents major differences and use cases of both the databases. It is found that MongoDB is better-suited document-oriented database for today's web applications than CouchDB.

Keywords: NoSQL, Document-Oriented Databases, MongoDB, CouchDB, Map-Reduce

I. INTRODUCTION

NoSQL is the term used to express data stores that do not follow the relational model and do not use SQL (Structured Query Language) as the data query language. These databases scale horizontally and dynamically to support a large number of users and a big amount of data [8]. They are mainly categorized into 4 classes namely - graph store, document store, key-value databases and column-oriented databases. In a document-oriented database a "document" is analogous to "row" of a relational database. These databases use a flexible data model. They use the documents which represent some real life entities and are very similar to constructs of object-oriented languages [1]. These databases allow use of values of keys via its content. Documents of these databases may be in BSON, XML or JSON formats. The documents are made up of complex elements such as scalar values, maps and collections. A document's keys are not of fixed types and values are not of fixed sizes. In a dynamic schema model, adding or removing fields as needed becomes easier [1]. Each document is made up of fields and contains a unique ID which is used for retrieving and indexing the database. Generally, documents oriented-databases have more rich query language than other contemporary NoSQL databases. So they are able to make use of the structure of the objects they store. These databases follow CAP (Consistency, Availability and Partition Tolerance) theorem while relational databases follow ACID (Atomicity, Consistency, Isolation and Durability) properties [9]. The CAP theorem states that little bit inconsistencies are tolerable by distributed databases where partitions are allowed [2]. Most NoSQL databases follow any two of CAP properties out of three properties. The CAP theorem is shown in Fig 1. The main databases that fall in the category of document oriented databases are CouchDB and MongoDB.

II. MAP-REDUCE IN MONGODB

MongoDB is open-source document-oriented database developed by 10gen. It has the ability to scale out along with other features such as aggregations, geospatial indexes, secondary indexes, sorting and range queries [1].

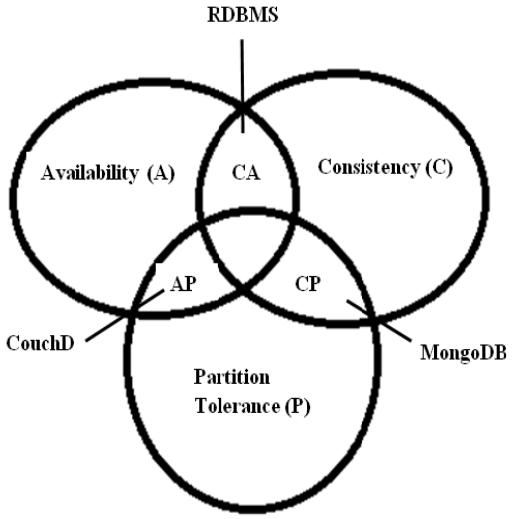

Figure 1. CAP Theorem

MongoDB databases are made up of collections. A collection contains many documents as shown in data model of Fig 2. Each document contains data in the form of key-value pair. Collections can be indexed, which improves lookup and sorting performance.

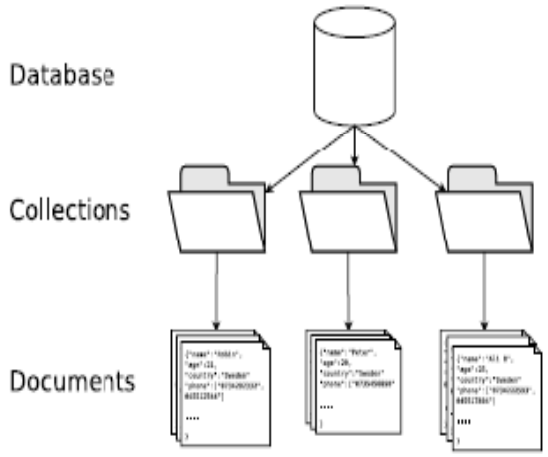

Figure 2. Data Model of MongoDB

MongoDB use map-reduce concept to carry out aggregation operation. Map-reduce concept involves two stages: a *map* phase processes each document of collection one by one and *emits* one or more objects as intermediate output. The *reduce* phase then aggregates the intermediate output of the *map* stage. An optional *finalize* stage may be used to make some changes to the final result. MongoDB provides the *mapReduce* database command to carry out aggregation operations over a collection [5]. Custom Java Script functions are used by MongoDB *"mapReduce"* command to perform the map, reduce and finalize operations. These functions provide immense flexibility but they are very complex. The syntax of this command is shown as below -

```
db.collection_name.mapReduce(
{
    map: <function>,
    reduce: <function>,
    finalize: <function>,
    out: <output>,
    query: <document>,
    sort: <document>,
    limit: <number>,
    scope: <document>,
    verbose: <boolean>
} )
```

The description of various mandatory and optional parameters used by *mapReduce* command is shown in table1.

Now, the example below shows the implementation of *mapReduce* command in the MongoDB. Suppose we have a test database "testdb" and collection "classes". We have inserted 6 documents in this collection one by one. The code below shows the insertion of one document in the collection "classes".

```
db.classes.insert ({
class : "Computers 100",
startdate : new Date(2018, 1, 06),
students : [ {fname : "Amit", lname: "Gupta", age: 25}
{fname : "Veer", lname: "Rai", age: 35}
{fname : "Samar", lname: "Kashayap", age: 30}],
cost : 1400,
professor : "Pradeep Kumar"
})
```

Now, suppose we want to find out how many classes are being taken by professor "Pradeep Kumar". The following code of mapReduce command will provide the required results. Firstly we need to write code for map and reduce functions and then these functions are used in *mapReduce* command.

```
var mapFunc = function( ) {
emit(this.professor, 1);     }
var reduceFunc = function(professor, count) {
return Array.sum(count);   }
db.classes.mapReduce(
mapFunc,
reduceFunc,
{ query : {professor : "Pradeep Kumar"},
out : "mapexample"
} )
```

The result of the above *mapReduce* command is shown as below –

```
{
"results" : "mapexample",
"counts" : {   "input" : 3,
               "emit" : 3,
               "reduce" : 1,
               "output" : 1
           }
"ok" : 1
}
```

Table 1: Description of Parameters of "mapReduce" command

Sr. No.	*Name of Parameter*	*Optional/Mandatory*	*Functionality*
1	Map	Mandatory	A JavaScript function that associates or “maps” a value with a key and emits the key and value pair.
2	Reduce	Mandatory	A JavaScript function that “reduces” to a single object all the values associated with a particular key
3	Finalize	Optional	It follows the reduce method and modifies the output.
4	Out	Mandatory	It specifies where to output the result of the map-reduce operation. We can either output to a collection or return the result inline.
5	Query	Optional	It specifies the selection criteria using query operators for determining the documents input to the map function.
6	Sort	Optional	It sorts the input documents. It is useful for optimization.
7	Limit	Optional	It specifies a maximum number of documents for the input into the map function.
8	Scope	Optional	It specifies global variables that are accessible in the map, reduce and finalize functions.
9	Verbose	Optional	It specifies whether to include the timing information in the result information.

The aggregated result is also available in collection "mapexample" as shown below –

db.mapexample.find()

{"_id" : "Pradeep Kumar", "value" : 3}

The above example shows the use of map-reduce functions in MongoDB to get aggregated results.

III. MAP-REDUCE IN COUCHDB

One word to describe CouchDB is "relax" [3]. CouchDB is an open source document-oriented database developed by Apache. It is written in Erlang language. Data in CouchDB is organized in the form of documents. Documents consist of

fields as key-value pair, which are stored and accessed as JSON objects. Various documents are grouped together in a database. So we can say that databases in CouchDB are collection of variable-schema documents as shown in data model of Fig 3.

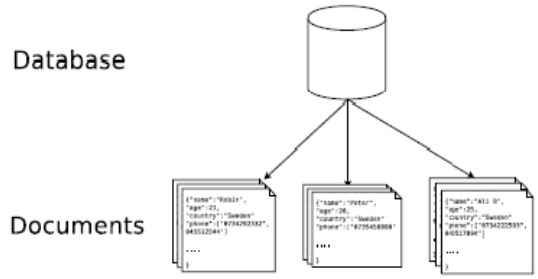

Figure 3. Data Model of CouchDB

CouchDB's RESTful APIs make it easy to use database because with the help of these APIs database can be accessed through http requests [4]. CouchDB can be accessed via both the interfaces i.e. GUI as well as CLI. The name of GUI utility of CouchDB is Futon and name of CLI utility is Curl [6]. APIs of CouchDB are run on the Curl. In CouchDB, views are used to query the database and for implementing views, map-reduce functions are used. These map-reduce functions provide great flexibility because they are capable of dealing with alterations in the structure of documents. These functions generate indexes parallelly and automatically. Documents are passed to map function as arguments one by one and whole database is traversed this way. Documents with matching criteria are given as output in the form of key/value pairs. CouchDB has two types of views namely temporary and permanent. Permanent views (static views) are formed with the help of unique documents called design document. Design document contains map function inside it and specify the criteria to get the view [7]. A single design document may contain many views. A JSON file named "design2.json" as shown in Fig 4 has been created. This file contains the id of the design document “phd_course" and map functions of two views namely “supervisor_detail” and “marks_detail".

design2.json - Notepad

File Edit Format View Help

```
{
 "_id" : "_design/phd_course",
 "views" : {
 "supervisor_detail" : {
 "map" : "function(doc){ if(doc.name && doc.supervisor){emit(doc.name, doc.supervisor);}}"
  },
 "marks_detail" : {
 "map" : "function(doc) {
var subject, marks, key;
 if (doc.name && doc.Marks) {
 for (subject in doc.Marks) {
 marks = doc.Marks[subject];
 key = [doc.name, marks];
 emit(key, subject);}}}"
  }
}
}
```

Figure 4. Design Document for Creating Views

Design document "phd_course" is created using file "design2.json" with the help of below command.

curl-X PUT http://127.0.0.1:5984/university/_design/phd_course-d @desktop/design2.json

Curl responds to the above command by showing following JSON document.

{"ok":true,"id":"_design/phd_course","rev":"1ddde17668d269f5cfeb2aa3259e26b1"}

In GUI interface "Futon" our design document looks like as shown in Fig 5 containing two views.

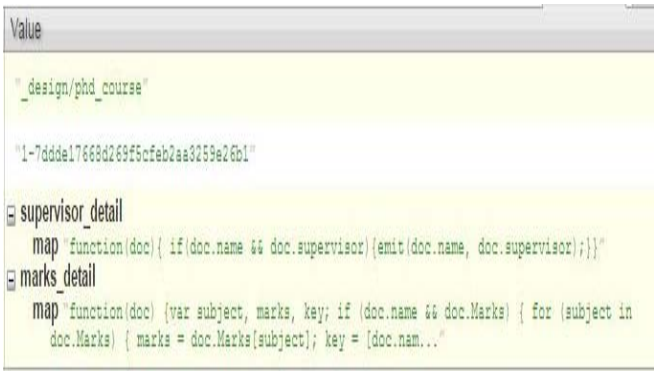

Figure 5. Design document in Futon

This result of the view is shown in Fig 6. The result shows the every subject's marks of each student individually.

Key	Value
["Jitender", 69] ID: ae288eecf814efef177e0428387002f8d	"Software Engineering"
["Jitender", 78] ID: ae288eecf814efef177e0428387002f8d	"Database"
["Jitender", 78] ID: ae288eecf814efef177e0428387002f8d	"Language - C"
["Ramesh Chander", 29] ID: ae288eecf814efef177e0428387000eb7	"Software Engineering"
["Ramesh Chander", 31] ID: ae288eecf814efef177e0428387000eb7	"Database"
["Ramesh Chander", 35] ID: ae288eecf814efef177e0428387000eb7	"Language - C"
["Reet", 62] ID: ae288eecf814efef177e0428387031e7	"Analog"
["Reet", 79] ID: ae288eecf814efef177e0428387031e7	"MicroProcessor"

Figure 6. Result of Permanent View

IV. DIFFERENCES BETWEEN COUCHDB AND MONGODB

MongoDB is more suited for online web applications, while CouchDB is better to use for applications that do not have internet assured. CouchDB and MongoDB have a lot in common, but also have a lot of differences. The major difference between both databases have been explained below and also depicted in table 2.

Table 2: Differences between MongoDB and CouchDB

Sr. No.	*Feature*	*CouchDB*	*MongoDB*
1	Developer	CouchDB is an Apache product.	MongoDB is developed by 10gen.
2	Language in which written	It is written in Erlang.	It is written in C++.
3	Data Format	It follows the document-oriented model and data is presented in JSON format.	It follows the document-oriented model but data is presented in BSON format.
4	Interface	CouchDB uses HTTP/REST-based interface. It is very intuitive and very well designed.	MongoDB uses binary protocol and custom protocol over TCP/IP.
5	Data Model	In CouchDB, database contains documents.	In MongoDB, database contains collections and collection contains documents.
6	Query Method	CouchDB follows map-reduce query method.	MongoDB follows map-reduce and object-based query language.
7	Replication	CouchDB supports master-master replication with custom conflict resolution functions.	MongoDB supports master-slave replication.
8	Concurrency	It follows MVCC (Multi Version Concurrency Control).	It follows update-in-place.
9	Preferences	CouchDB favors availability.	MongoDB favors consistency.
10	Performance	CouchDB is safer than MongoDB.	MongoDB is faster than CouchDB.
11	Consistency	CouchDB is eventually consistent.	MongoDB follows strong consistency.
12	Data Awareness	It does not provide data awareness inherently but possible with the help of GeoCouch.	It provides data awareness inherently.
13	Sharding	It does not provide auto-sharding. Sharding can be done with the help of third-party software CouchDB Lounge.	It has auto-shading capabilities.

V. CONCLUSION

MongoDB supports multiple collections, stores data in BSON (faster access), does better querying, does aggregation through map-reduce and is capable of executing millions of queries per second. CouchDB stores data in JSON, querying through map-reduce. CouchDB does not support distributed map-reduce, sharding and data awareness in its standard implementation. For these features, third-party software is required, like Big Couch and Lounge. So, MongoDB is faster and CouchDB is safer. Based on the above arguments we can

realize that MongoDB is better-suited document-oriented database for today's web applications than CouchDB.

VI. REFERENCES

[1] Chodorow, K. (2013). MongoDB: The Definitive Guide: Powerful and Scalable Data Storage. " O'Reilly Media, Inc.".

[2] Frank, L., Pedersen, R. U., Frank, C. H., & Larsson, N. J. (2014, January). The CAP theorem versus databases with

[3] relaxed ACID properties. In Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication (p. 78). ACM.

[4] Anderson, J. C., Lehnardt, J., & Slater, N. (2010). CouchDB: The Definitive Guide: Time to Relax. " O'Reilly Media, Inc.".

[5] Henricsson, R., & Gustafsson, G. (2011). A comparison of performance in MongoDB and CouchDB using a Python interface. Bachelor thesis BTH.

[6] Kumari, S., & Gupta, P. (2017a). Proposed Architecture of MongoDB-Hive Integration. International Journal of Applied Engineering Research, 12(15), 5000-5004.

[7] Kumari, S., & Gupta, P. (2018). Implementation of CouchDBViews. In Big Data Analytics (pp. 241-251). Springer, Singapore.

[8] Gulia, P. & Hemlata (2017). Novel Algorithm for PPDM of Vertically Partitioned Data. International Journal of Applied Engineering Research, 12(12), 3090-3096

[9] Hemlata, Gulia, P. (2018). DCI3 Model for Privacy Preserving in Big Data. In Big Data Analytics (pp. 351-362). Springer, Singapore.