MongoDB vs Couchbase Server

Jithin Jacob Benjamin Jacob
Department of Computer Science
New Mexico State University
Las Cruces, New Mexico 88003, USA

Sreeja Matturu
Department of Computer Science
New Mexico State University
Las Cruces, New Mexico 88003, USA

December 1, 2019

Abstract

Here in this report we look at the comparison between MongoDB and CouchBase server. Here both these systems are Open-source, document store and store documents in JSON format. Here these two systems are chosen because they both are used by a wide variety companies where they support applications which generate a high growth in data and considers Fault tolerance and High availability as their highest priority.

Installation of the System

For implementing these systems we chose Amazon Web Services(AWS) as our platform provider. Both the systems were installed on AWS and tested in three setups 1Master, 1Master-1Slave, 1Master-3Slave configurations to compare their performance in a parallel query processing environment using a parallel database setup.

Installation and Setup of MongoDB

The EC2 instances were rented from Amazon Web Services through MongoDB Atlas which is a cloud version of MongoDB. The System was installed with the M30(m5a.large) Tier of MongoDB which is explained in detail in the Hardware Specification. Create a new cluster name with Admin username and password. Then the connection to the database was established from the terminal of the local system through the following code.

mongo "mongodb+srv://< Cluster-Name>.mongodb.net/<Database-Name>"

–username <User-Name>

Then we create the collections folder inside MongoDB Atlas, here in out case it is Business and Tip. For importing the collection of documents into the MongoDB instances after establishing the connection to the system through the terminal we add our local systems IP address to the set of accepted IP address on our MongoDB Atlas portal and then use the following code for importing the dataset to the MongoDB system.

mongoimport –host <Cluster-Name>.mongodb.net:27017 –db <Database-Name> –collection <Collection-Name> –type json –file <Local JSON file Directory> –jsonArray –authenticationDatabase admin –ssl –username <Username> –password <Password>

Once this above step is done then our Data-sets will be inserted into our MongoDB instances. Here in MongoDB when we say a single system it is comprised of a single RAM unit distributed among 3 replica sets(3 Shards is the Default no of shards within each system) of Data in them. The queries were executed from the terminal after implementing the connection between the terminal and MongoDB Atlas.

Installation and Setup of CouchBase Server

The m5a.large AWS EC2 instances were hosted for CouchBase server along with instances for CouchBase sync gateway which is the key factor being authentication and replication among the CouchBase servers. For Good performance the recommended ratio of Sync gateway instances for the CouchBase instances are 1:2 where for every two CouchBase Server instances we need 1 CouchBase Sync gateway instances with the same hardware spec for providing high availability.

Once the system has been installed then they can be accessed by their public IP address on the web browser followed by the port number. Here we can login using the username and password provided and choose create a new cluster option and choose the amount of storage assigned for each attributes like Data, indexes, search etc. Now it is time for us to connect our parallel systems to the network here under servers option we can click add server, type in the public DNS address of the servers under consideration and give the username and password of those servers and they will be connected in a parallel system with the current system being the master node. Now we can click re-balance to replicate all the data among these parallel systems. Then we can create the collection names and then import the data into the database through the terminal by giving the following code

cbimport json -c couchbase://<Couchbase IP Address> -u <Username> -p <Password> -b <Bucket Name> -d file:/<JSON file directory> -f list -g key::%<Key Generator Attribute>%::#UUID# -t 4

Once all the data has been imported into the database system now we have to create the primary index in our database for both of these collections using the following query

CREATE PRIMARY INDEX '<Primary Index Name>' ON '<Bucket Name>' USING GSI;

Then the query's were executed using the query tool within the CouchBase server.

Hardware Specifications of Both the Systems

The instances that were used are m5a.large AWS instances which has 8GB Ram in each, 2-core vCPU from a Intel Xeon Platinum 8000 series (Skylake-SP) processor and has a Network Bandwidth of up to 10 GBPS and a Elastic Block Storage Bandwidth of 2120 Mbps. They were all chosen from the region AWS us-east-1 (North Virginia).

The Data Chosen for the system comparison

This dataset that we have chosen is a subset of Yelp's businesses, reviews, and user data. It was originally put together for the Yelp Dataset Challenge which is a chance for students to conduct research or analysis on Yelp's data and share their discoveries. we have two collection from this dataset which is Business with 192609 documents and the other one is Tip which has 1223094 documents in its collection. Both have a unique value called business_id in them.

Query Scenario

Here the query scenarios we have chosen for evaluation are Point queries for displaying all the review_count above 1000 in business, sorting and group-by function based on compliment_count from business collection, join operation between both of these collections, Aggregate function queries like Min and Max and range query for returning all documents where the review_count is between 0 and 50 and lastly we have executed the exact range query using Map-Reduce function as well.

Evaluation Chart

1 Master - 3 Slave

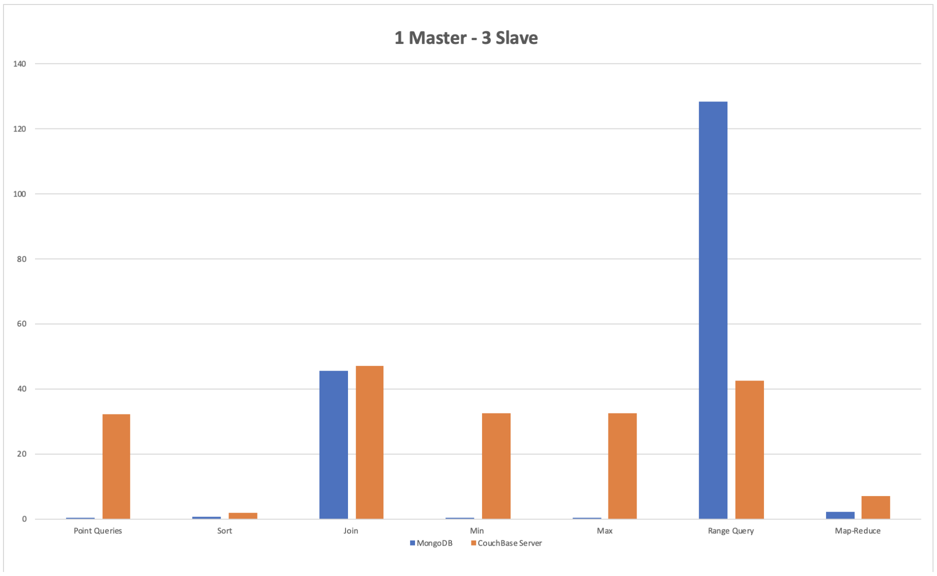

Figure 1: 1 Master - 3 Slaves

Here we have the comparison of MongoDb and Couchbase Server for a 1 master and 3 Slave parallel database system. Here the results that we found are

- The execution time of single point query is much faster in MongoDB compared to Couchbase server this is due to the fact that every MongoDB is a parallel system within and they have a better execution time for point queries compared to couchbase server which has to depend on indexes to do this process, if the index is not present on the searched key then this makes it slow for processing in couchbase server.
- Sort operation is faster in MongoDB than Couchbase because of its in memory sort operation query processing.
- Join operation is slightly faster in MongoDB than Couchbase in this parallel setup as the join operation is distributed among the parallel systems and carried on at the same time in MongoDB.
- The aggregate function min and max is faster in MongoDB due to its replica sets within each system which makes its read operations within a single collection very fast, whereas this kind of replication is not present within each system in Couchbase server.
- For calculating the range query among these systems MongoDB is slower than Couchbase server due to the fact that that due to the returned size of

the data the in-memory query processing has been moved to a temporary file on the disk for doing this range query and also there is no index created on those key value thereby making this slower than Couchbase server.

- Map-reduce was faster in MongoDB compared to Couchbase as they create a new index which is faster for doing this Map-reduce function and returns the value anytime using this index.

1 Master - 1 Slave

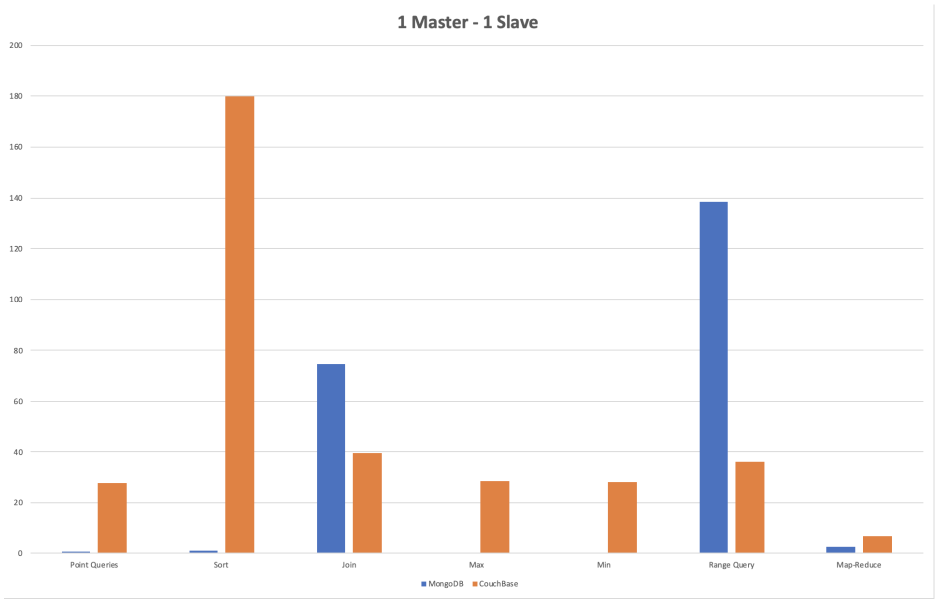

Figure 2: 1 Master - 1 Slaves

For a 1 Master - 1 Slave we got the following finds while comparing MongoDB with couchbase server

- Point query performance was again faster in MongoDB compared to Couchbase server, but the execution time is reduced from the previous setup of couchbase because here when we have less no of I/O communication latency among the couchbase server instances and the couchbase sync gateway instances, this network latency increase the couchbase server execution time by a small margin.

- MongoDB was faster in the sort operation due to in-memory sort algorithm and couchbase server is extremely slow in this case because only one sync gateway instance is used here instead of two in the previous case which made the sorting operation very slow.

- Join between both of the collections were slow in MongoDB and faster in Couchbase server. Here one thing to note is that the performance has been become slightly bad in MongoDB compared to the 1 master - 3 Slave. Whereas in Couchbase server reducing the no of instances reduced the I/O overhead between the instances and the Couchbase sync gateway and thereby increased its performance.
- Due to in-memory query processing and indexes the the aggregate functions of Min and Max were extremely faster in MongoDB than Couchbase server which shows slow performance due to the I/O overhead between the instances and the Couchbase sync gateway but they have improved from the previous setup.
- Range Queries were slower in MongoDB than Couchbase Server just like the 1 master - 3 slave, the time for execution has also in increased in MongoDB as the no of instances are less now. But the Couchbase server shows increase in the performance and also faster for range query than MongoDB.
- The Map-reduce function is still faster in MongoDB than Couchbase server

1 Master

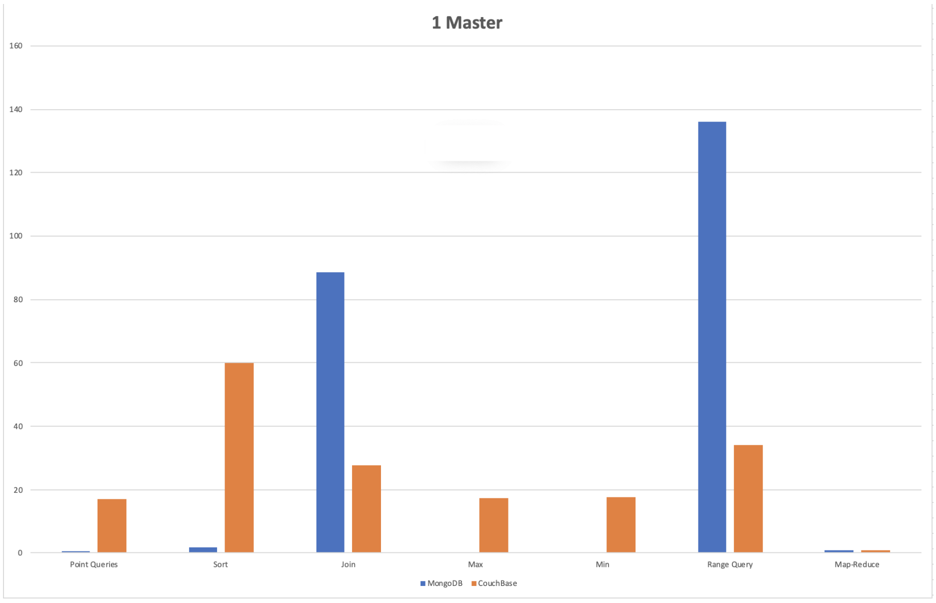

Figure 3: 1 Master

Now we are looking at a single master system performance for the same queries that we have executed before.

- Point Queries are still faster in MongoDB due to its in built parallel replica set(shards) than Couchbase server
- The sort operation is faster in MongoDB than Couchbase but now it is slightly slower than its performance from its previous setup. But Couchbase server has increased slightly in performance in this setup.
- MongoDB is slower for join operation than Couchbase and its slightly slower than the previous setup as well due to the fact that a single system is doing all the join operation.
- Aggregate functions Min and Max is still faster in MongoDB than Couchbase server due to the use of indexes and In-memory query processing.
- Range query is slower in MongoDB compared to Couchbase server. Here this can be overcome if a index had been created over the key attribute in MongoDB.
- Map-reduce performance is almost the same for MongoDB and Couchbase server in a single Master system.

Cost

Comparing the cost of MongoDB with the Couchbase server we find that the cost is high for Couchbase server as in terms of physical hardware as well as software licenses for a parallel system. For the use of a two system configuration MongoDb needs only two instances to implement this but whereas in Couchbase it needs 3 instances where 1 instances is dedicated for the Couchbase sync gateway server with a high cost of its licensing. In our case for every Couchbase sync gateway server the cost was $2.709 just for its licensing use whereas non of this drawbacks are found in MongoDB.

How are Multiple Queries Executed Parallelly?

MongoDB allows to run simultaneous queries in parallel, it allows multiple clients to read and write the same data. In order to ensure consistency, it uses locking and other concurrency control measures to prevent multiple clients from modifying the same piece of data simultaneously. MongoDB uses multi-granularity locking operations to lock at the global, database or collection levels and also allows for individual storage engines to implement their own concurrency control below the collection level using WiredTiger storage engine which provides document-level concurrency model, check pointing and compression in MongoDB. It uses a shared locking mode for reads and exclusive locking mode for write operations and for higher levels it uses intent exclusive modes for any intent to read or write a resource using a finer granularity lock.

How are Single Query Parallelized?

For a single query parallelism, MongoDB uses Intra-operation parallelism for executing its queries in a parallel system. Here a single query is decomposed into smaller tasks that execute concurrently on multiple processors.

Sort function in MongoDB? Is it in-memory sort algorithm, or external sort algorithm?

The sorting algorithm used by MongoDB is a top k-sort algorithm which is an in-memory algorithm. The sort is the fast operation in MongoDB and sometimes, the sort gets satisfied by scanning for an index in any order. In this case, if the query execution plan use the index that is provided by the requested sort order, then MongoDB uses that to perform the in-memory sorting for the result set. In MongoDB the in-memory sort has a limit of 100MB if it exceeds that limit then we have to enable allowDiskUse option to allow the database to create a temporary file on the disk to perform this sort operation.

How are Joins parallelized?

Joins are supported in MongoDB. Here we use $lookup operator for performing the join operation and it performs left outer join between two or more collections in the database using a aggregate pipeline structure. MongoDB in general is a parallelly replicated system where it will store a set of 3 default replica set in each system. Here during the join operation the mongos connects with the config server present within each system and retrieve the metadata of all the chunks in each shard system and send a request to all the systems to return a partial set of the documents from those collections each, which when returned contains the result of the entire join operation. Generally the join operation is slow in MongoDB.

How are Query Plans evaluated Parallelly?

During a query plan evaluation the MongoDB database uses the MongoDB query optimizer to evaluate the best query plan. If a query can be satisfied by multiple indexes in the system then MongoDB will test all those application indexes in parallel, the first index with the query plan that can return 101 result will be chosen by the query optimizer.

Conclusion

MongoDB and CouchBase server had their strength's and drawback's. MongoDB Proved to be a better performer in Point queries, Aggregate functions and Map-reduce compared to Couchbase server. But Slower in performance in executing join's and range. This drawbacks can be overcome to a point by implementing new indexes based on a user's application requirement and that would make the MongoDb system perform faster for these operations. Coming to the infrastructure cost MongoDB is the cheapest compared to Couchbase Server, on average the cost difference for implementing MongoDb and Couchbase server is in the ratio of 1:5. This makes MongoDB a great chose when we need a NoSQL database which can handle rapid data growth.

Team Members Contributions

Jithin Jacob Benjamin Jacob

- System Installation and parallel connection and Data import
- Performance metrics comparison
- Answers on Single Query, Multiple Query, Join and Query Plans in Parallel
- Report

Sreeja Matturu

- Query Scenarios
- Performance Graph Charts
- Answers on Sort functions in MongoDB
- Powerpoint Presentation