Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

Performance Evaluation of Mongo DB:Native Driver and Mongoose

Saptarshi Samanta
Student, Bachelor of Technology in Computer Science
Kalinga Institute of Industrial Technology
India

***Abstract*—NoSQL databases are an excellent alternative to relational databases and can manage huge volumes of data. One such popular NoSQL document database is MongoDB. In this paper we will try to evaluate the performance of MongoDB native driver and its popular framework mongoose with CRUD operations.**

***Keywords*— MongoDB , Mongoose , CRUD operations , NoSQL , Document database**

I. INTRODUCTION

Few years back, if we looked into software applications like websites, it had few thousand to tens of thousand of users. But the landscape of internet activity has changed and millions of users visit sites 24/7, 365 days a year.
Relational Databases performed really well with those limited number of users.
But with the emergence of big data, it had problem with 3V's : volume , variety and velocity.
"Not Only SQL" or NoSQL databases came out as an alternative to relational database to solve these problems.
One important feature of NoSQL database is that it can handle unstructured data efficiently.
There are primarily four types of NoSQL database:

- Key-Value: Key-Value databases are schema less and are like distributed dictionaries.
- Wide-Column: In wide-column database, data is stored in columns, organized into column families. But unlike relational database, Wide Column database can have variable number of columns to support multiple data types.
- Document: In document database, data is stored as nested key value pairs, known as documents.
- Graph: Graph databases are used to store data which are highly connected.

This paper is separated in sections: Section 2 describes NoSQL databases in detail. Section 3 describes Development experience. Section 4 shows experimental results. Section 5 presents the conclusions.

II. NOSQL DATABASES

NoSQL , aka "Not Only SQL" databases are generally non tabular databases like relational database.
NoSQL database use different data models for managing and fetching the data. NoSQL databases do not have a pre defined schema, and are schema less. This feature makes NoSQL database to be used for applications which require flexibility and can handle large volume and variety of data.
.

A. SQL vs NoSQL database example

Here is an .example of modeling the schema for class grades.

- In relational database , the record are stored in separate tables and relationships are established by primary and foreign keys.In the example a table Student has columns like student id and student name and the Grade table has columns like student id and grade.
- But in NoSQL database like MongoDB, which is a NoSQL Document database it will be written as a documents with key value pairs. The collection name is Student Grade and the keys of key value pair of the documents will be student id, student name and grade.

B. SQL vs MongoDB terminology

Table 1. SQL vs MongoDB terminologies.

SQL	**MongoDB**
Table	Collection
Row	Document
Columns	Field
Primary Key	ObjectId
Index	Index
Nested Table	Embedded Document

C. Types of NoSQL databases

Key-Value database: It is the pimpliest form of NoSQL database. Data is stored in dictionary like key-value pairs. The data can be fetched using a unique key given to each element in the database. The values can range from simple data types like strings to complex like objects.
Examples: Couchbase , Amazon DynamoDB , Riak
Document Database: In document database, instead of storing data in rows and columns, data is stored as collection of key value pairs inside documents. A collection is like a table, which contains multiple documents. A document database stores data in JSON, BSON or XML documents.
Example:MongoDB , DynamoDB , CosmosDB
Column-Oriented database: These databases store all the data in columns. When we want data from small number of columns, we can access them with greater speed and lesser memory usage.
Example: Cassandra
Graph Database: Graph is used when data have many relationships. All data are stored as nodes in database and connections between node is called links.
Example: Nebula Graph

III. DEVELOPER EXPERIENCE

A. MongoDB Native Driver

The MongoDB Native Driver is the official client library for MongoDB. MongoDB [1] stores data in flexible, JSON-like document format, where fields can vary from document to document and data structure can be changed over time.

Features of MongoDB native Driver:
●Provides optimized communication between the application and MongoDB Database.
●BSON aka Binary JSON is binary encoded serialization of JSON like documents, used by MongoDB for efficient data exchange.
●Supports indexing which enhances performance.
●Provides high availability, horizontal scaling and geographic distribution.

B. Mongoose

Mongoose is MongoDB Object Data Modelling library used with Node.js which provides higher-level abstraction over the native driver. Unlike MongoDB native driver, mongoose is flexible schema based where the developer can define fields, data types and validation rules.

Features of MongoDB native Driver:
●Supports Middleware functions which helps developers to make custom logic for events like validation, pre , post operations.
●Supports population where it can automatically replace specified path in a document with documents from other collection.
●Supports hooks like pre , post .

IV. TIME ANALYSIS

Test cases were performed to check and compare the time taken by MongoDB native driver and mongoose for CRUD operations on 100 , 1000 , 10000 , 100000 documents in a collection respectively. The time take in milliseconds is the average of observations of 5 tests conducted.
The tests were conducted in using a machine equipped with an AMD Ryzen 7 5700U processor with Radeon Graphics 1.80GHZ, 16 GB ram, 64-bit Operating System. The test was performed in local machine using MongoDB compass.

A. Time evaluation for Create operation:

The following table and graph shows the comparison between time taken to create the number of documents using MongoDB native driver and Mongoose.

From Table 2 , it is clear that MongoDB native driver is faster than mongoose.

Table 2. Time taken to create documents using MongoDB native driver and using Mongoose

No. of documents	MongoDB (in ms)	Mongoose (in ms)
100	150.2	254.4
1000	962	1699.6
10000	7041	12712.8
100000	64958	116988.2

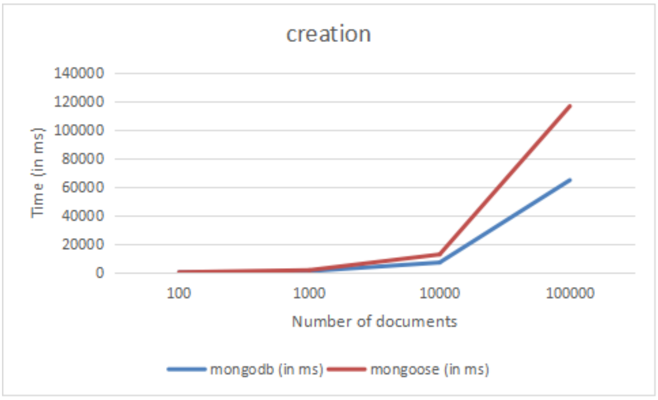

Figure 1. Time taken for creation of documents

B. Time evaluation for Read operation:

The following table and graph shows the comparison between time taken to read the number of documents using MongoDB native driver and Mongoose.

From Table 3 , it is clear that MongoDB native driver performed faster than mongoose.

Table 3. Time taken to create documents using MongoDB native driver and using Mongoose

No. of documents	MongoDB (in ms)	Mongoose (in ms)
100	97	136.2
1000	983.4	1295.6
10000	39837.2	42532
100000	4106458	3739444

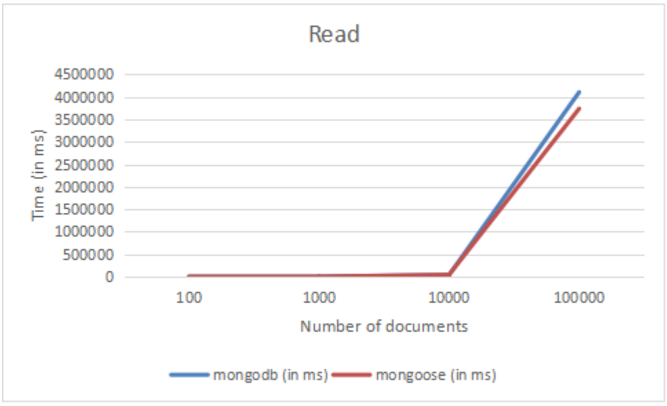

Figure 2. Time taken to read the documents

C. Time evaluation for Update operation:

The following table and graph shows the comparison between time taken to update the number of documents using MongoDB native driver and Mongoose.
From Table 4 , it is clear that MongoDB native driver performed faster than mongoose as number of documents increased.

Table 4. Time taken to create documents using MongoDB native driver and using Mongoose

No. of documents	MongoDB (in ms)	Mongoose (in ms)
100	151.2	147.8
1000	1273.6	989
10000	9631.2	7367.6
100000	41809.8	68875.4

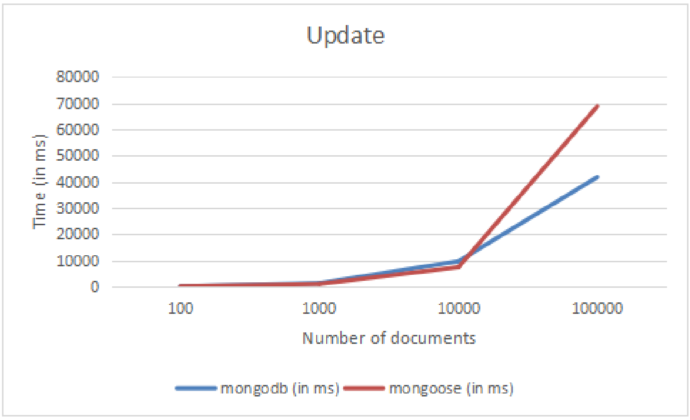

Figure 3. Time taken to update the documents

D. Time evaluation for Delete operation:

The following table and graph shows the comparison between time taken to Delete the number of documents using MongoDB native driver and Mongoose.

From Table 5 , it is clear that MongoDB native driver performed faster than mongoose.

Table 5. Time taken to delete documents using MongoDB native driver and using Mongoose

No. of documents	MongoDB (in ms)	Mongoose (in ms)
100	146.8	191
1000	1212.2	1470.6
10000	9712.2	11307
10000	51131.6	107078

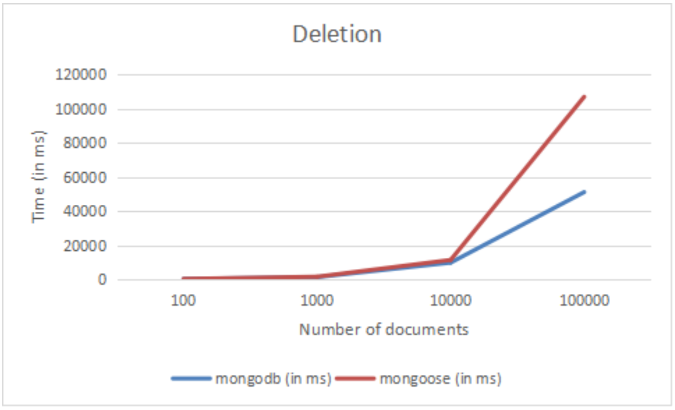

Figure 4. Time taken to delete the documents

E. Conclusions:

This Paper evaluates developer experience and performance for MongoDB native driver and mongoose. This paper can help the readers choose when to use native driver and mongoose respectively based on requirements.
The native driver provides better speeds of larger data during CRUD operations. Mongoose on the other hand has middleware support and hooks which makes it a good choice for developers.

ACKNOWLEDGMENT

I would like to share my gratitude to every reader of this paper.

REFERENCES

[1] Sarita Padhy & G Mayil Muthu Kumaran (2019) "A Quantitative Performance Analysis between Mongodb and Oracle NoSQL", International Conference on Computing for Sustainable Global Development (INDIACom), IEEE