
BENCHMARKING DATABASES USING APACHE JMeter

Aditi Mittal(amittal@umass.edu) Christopher Gomez(cdgozmez@umass.edu)
Mehul Ramaswami(mramaswami@umass.edu)

1 INTRODUCTION

Every business must focus on different metrics to decide the database they should use to fulfil the technical goals within the available cost. For example, if the focus of the system is on the latency then it should choose the database that provides the lowest latency for the amount of data it would be processing along with the computation power it can afford. They should also consider which class of operations will be more frequent. This makes it important to analyze the performance of both relational and non relational databases on the same type of queries and datasets so that the metrics can be appropriately compared.

In one of the existing works, the authors (Rautmare & Bhalerao, 2016) have compared two systems - MySQL and MongoDB - on the IoT sensor data using JMeter. In other work, the author (Keshavarz, 2021) compared the runtime of MySQL and MongoDB by performing set of queries containing read, write, update, and delete operations on the same data. They concluded that MongoDB performed better than MySQL for most operations.

JMeter is an Open Source tool, widely used to measure performance parameters of an application. It provides an interactive user-friendly GUI as can be seen in Figure 1 for creating the test plans and running in interaction with the database where the user can specify the parameters such as the number of threads and the query. It requires a JDBC to connect to the database. Our work focuses on providing more detailed analysis on the specific metrics of latency and throughput on two different type of publicly available datasets. The bench-marking and analysis will be performed on the three database systems of MySQL, SQLite and MongoDB using the Apache JMeter tool. We will be running lookup, insert and update queries to compare their performances with fixing the number of threads.

2 PROBLEM STATEMENT

In this project, we will be focusing on comparing the relational (MySQL, SQLite) databases with non-relational database(MongoDb) in terms of database metrics like

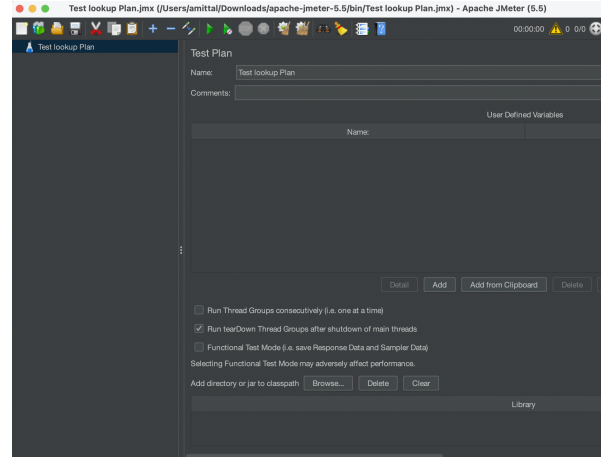


Figure 1. JMeter GUI

throughput and latency. We will be using two datasets: the "Walmart Recruiting - Store Sales Forecasting" dataset available on Kaggle and the Airbnb dataset available on Airbnb's website (<http://insideairbnb.com/get-the-data/>). But for the milestone, we just focused on the Walmart Recruiting - Store Sales Forecasting dataset.

We got our metrics by modifying the number of threads(users) and loop count.

Our expected result is that MongoDB being a non-relational database, will perform better on both the datasets for most of the queries. We also expect MySQL and SQLite to perform somewhat similarly. The evaluation will be based on the metrics mentioned above.

3 TECHNICAL APPROACH

First the dataset should be cleaned and preprocessed. For some records in the dataset, fields such as MarkDown1, MarkDown2, MarkDown3, MarkDown4, MarkDown5 were 'NA' which is not a valid value. These values needed to be removed with a valid value which can then be inserted to the database. For this task a simple python script was written that appropriately changed the 'NA' values based on the type of column. For example, in the MarkDown fields, the values that were present were in the form of integers; so the 'NA' values were thereby updated to '0'.

There were 8517 records inserted in the table. We will integrate JMeter with the three databases and run the test plan for three types of queries like select, insert and delete. The test plan was constantly run for 10 threads with a loop count of 10. Minimum elapsed time, maximum elapsed time, throughput, min latency, and max latency was recorded for comparing the three databases. For MySQL, we will be using the following select query for loopkup metrics: select Store, Unemployment from walmart where IsHoliday=FALSE; Here **walmart** is the table in database **cs532** and Store, Unemployment and IsHoliday are the fields/columns in the table.

Configuration Used in JMeter

Max Number of Connections (threads): 10

Time between eviction runs: 60000 ms

Max Wait: 10000 ms

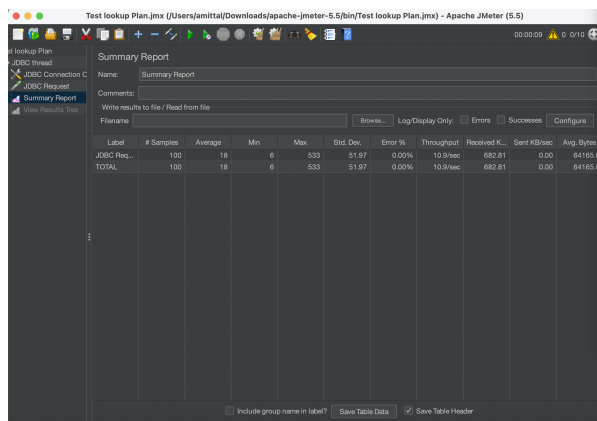
Loop count:10

4 INTERMEDIATE RESULTS

For intermediate results, our goal was to run the lookup query on all the three databases for finding the latency and throughput, and the only database with fully-complete querying so far is MySQL. We are still trying to finish querying with MongoDB, and only afterwards will we continue onto SQLite.

4.1 MySQL with JMeter

Figure 2 shows the results of JMeter for lookup query with MySQL. We can see the lowest elapsed time was 6 milliseconds and maximum was 533 milliseconds. Throughput was found out to be 56.1 requests/hour. Among all the requests, maximum Latency was found out as 351ms and minimum latency was found out to be 6ms.



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
JDBC Req...	100	18	6	533	51.97	0.00%	10.9/sec	682.81	0.00	64165.0
TOTAL	100	18	6	533	51.97	0.00%	10.9/sec	682.81	0.00	64165.0

Figure 2. Results for select query with MySQL

4.2 MongoDB with JMeter

For integrating JMeter with MongoDB we are able to make it connect to the database but are encountering a time-out error on querying. There were many complications when integrating JMeter with MongoDB as it had to be done using Groovy after updating many incompatible jar files built into JMeter. We are still looking into the cause of the timeout error.

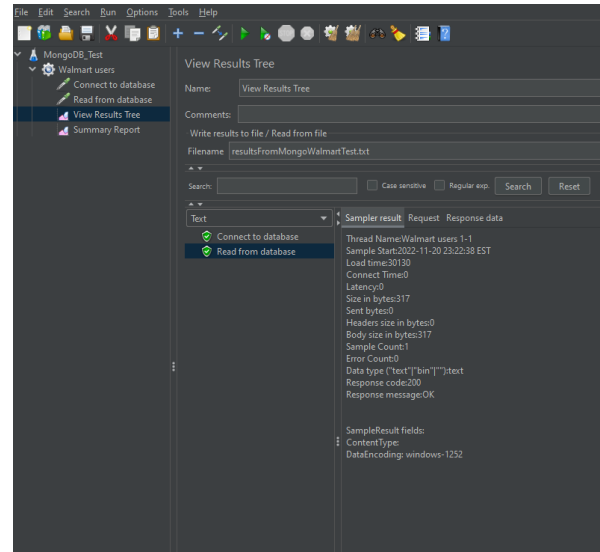


Figure 3. Current progress on querying with MongoDB

REFERENCES

- Keshavarz, S. Analyzing performance differences between mysql and mongodb. 2021.
- Rautmare, S. and Bhalerao, D. M. Mysql and nosql database comparison for iot application. In *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*, pp. 235–238, 2016. doi: 10.1109/ICACA.2016.7887957.