BENCHMARKING DATABASES USING APACHE JMETER

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

ABSTRACT

For every business application, an in-depth research is required to find a suitable data storage to serve the business' technology needs within the required cost. In this project, we aim to compare the average latency, average response time and throughput of lookups, insert, update and delete queries for both relational and non-relational databases. We will be utilizing the three major database systems MySQL, SQLite and MongoDB for bench-marking on four types of queries using the Apache JMeter tool. We will then use that analysis to justify which database management systems are faster, more efficient, and provide the best performance for our tested datasets.

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 performance metrics can be appropriately compared.

1.1 Databases

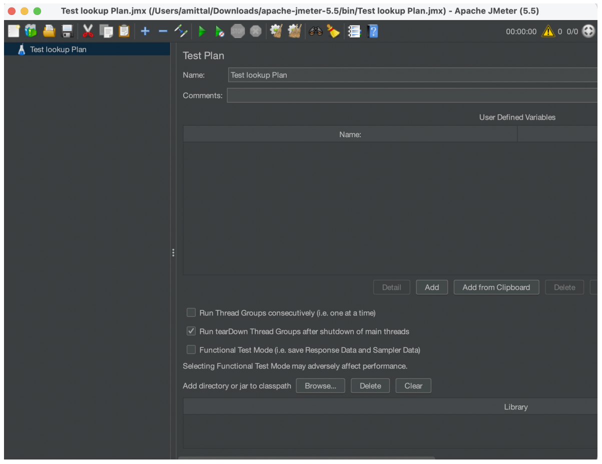
Relational databases are the databases which stores data in the form of tables, rows and columns. The data is structured and can be queried easily. Many relational database systems use Structured Query Language (SQL) for both querying and maintaining the data in the database. Examples of relational database are: SQLite, MySQL, and PostgreSQL.

Non-Relational databases are type of databases that do not use the tabular structure of rows and columns. They generally store data or records in form of documents and are schema free, which allows more complex information storage. They have increasingly become more used in big data and real-time web applications. Examples of non relational databases are MongoDB, Redis, HBase.

1.2 Apache JMeter

Apache JMeter (Halili, 2008) is an Open Source tool that is widely used to measure the performance parameters of an application. It is Java a based application and is highly extensible through its provided API. It provides an interactive user-friendly GUI, as can be seen in Figure 1, that can be used for creating the test plans and running interactions with the database where the user can specify the parameters, such as the number of threads and types of queries, to test. A typical plan involves creating the loop and defining the thread group. The loop helps in simulating sequential requests to the server while a thread group helps in simulating concurrent load. The plan also needs a sampler which is basically a configurable request to the server. It can be a JDBC request, JSR223 Sampler etc where we can specify the SQl query or even a groovy script to query the database. We can also add a listener to the test plan which is used to post process the request data like building the chart of results or creating a summary report of the metrics like throughput and elapsed time. It requires a correct version of the driver to connect to database. After creating the test plan through the GUI, it can be run using the CLI which can generate the report in html format and it is easy to understand. It provide metrics over time such as throughput, threads, response time and latency. It also consists of the error report generated while running the test plan.

Our work focuses on providing more detailed analysis on the specific performance metrics of latency, response time and throughput on two different type of publicly available datasets. The bench-marking and analysis was performed on the three database systems of MySQL, SQLite and MongoDB using the Apache JMeter tool. We will be running lookup, insert, update and delete queries to compare


Test lookup Plan.jmx (/Users/amittal/Downloads/apache-jmeter-5.5/bin/Test lookup Plan.jmx) - Apache JMeter (5.5)

Test Plan

Name: Test lookup Plan

Comments:

[ ] Run Thread Groups consecutively (i.e. one at a time)

[✓] Run tearDown Thread Groups after shutdown of main threads

[ ] Functional Test Mode (i.e. save Response Data and Sampler Data)

Selecting Functional Test Mode may adversely affect performance.

Add directory or jar to classpath

Figure 1. JMeter GUI

their performances while keeping the other configurations the same. The next sections following this will discuss the related works, technical approach, experiments, results and then will finish up with the conclusion and team contributions.

2 Related work

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

In another work, authors (Zaman et al., 2021) compared the loading time, response time, and retrieval time of both Azure SQL and Atlas MongoDB NoSQL databases using the JMeter tool. According to their experiment, it was observed that Azure SQL loading time, response time, and retrieval time was much less than the Atlas MongoDB NoSQL loading time, response time, and retrieval time. In the article, authors (Kholod, 2021) developed a testing application that dynamically visualizes results of conducting CRUD tests on databases such as MongoDB, MySQL and PostgreSQL. They used multiple tools for testing and visualisation such as JMeter, Grafana and InfluxDB. The work concluded that MySQL showed poor performance whereas MongoDB showed good performance and PostgreSQL also gave faster resposnes in few cases.

Authors (Armstrong et al., 2013) presented a new benchmark tool called LinkBench which was developed to evaluate database performance for workloads similar to Facebook's production data. This allows the tuning of different configurations and can be integrated with databases like MySQL, PostgreSQL.

In works related to JMeter for web application, (Nevedrov, 2006) has used JMeter to load-test the web service on BEA WebLogic Server 9.0 and recorded the reponse time with 5 concurrent threads hitting the server with 5ms and 10ms delays. (Tiwari et al., 2016) worked on proposing an approach to optimize performance of the databases using connection pooling method. They incorporated Apache JMeter for load testing to check performance of web application where they created one test plan for connection pooling and other for without connection pooling and then compared the performance. Authors in (Kiran et al., 2015) documented theor ecperiemnce while performing load testing of the web applications with Single sign-on (SSO) mechanism called Unified Authentication platform. It is a challenging task as JMeter does not capture all the dynamic values, such as SAML Request, Relay State, Signature Algorithm, Authorization State, Cookie Time, Persistent ID (PID), JSession ID and Shibboleth, generated using single sign-on mechanism of Unified Authentication Platform.

(Shenoy et al., 2014) developed a new test automation framework for Web services testing by aggregating common and project specific features of generic components and using JMeter. It would not require human intervention. Besides, the proposed system reduces test data dependency, which in turn speeds up execution process.

3 Technical Approach

For the experiments, we will be using two datasets:

1. Walmart Recruiting - Store Sales Forecasting Dataset available on Kaggle (Link)

2. Bike Sharing Demand dataset available on Kaggle (Link)

The benchmarking will be done using three performance metrics-

1. Throughput - Throughput is the term given to the number of requests that are processed within a specific period of time.

2. Average Latency - Latency in terms of database systems indicates how long it takes for the request to be completed.

3. Average response time - Response time measures time taken by an individual transaction or query. It is the elapsed time from the timestamp that a user sends the query until the time that the application indicates that the query has completed.

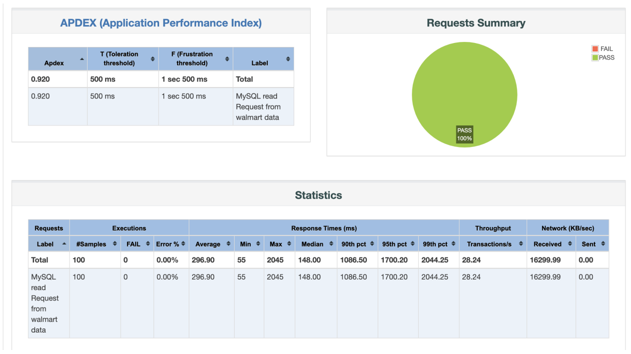
Our expected result from the experiments is that MongoDB being a non-relational database, will perform better on both the datasets for most of the queries. We also expect MySQL and SQLite to perform somewhat similarly. The evaluation will be based on the metrics mentioned above and can vary considering various factors such as database design, size of the dataset inserted in the database etc.

For the experiments, the datasets should be first cleaned and preprocessed. For some records in the dataset of walmart, fields such as MarkDown1, MarkDown2, MarkDown3, MarkDown4, MarkDown5 were 'NA' which is not a valid value. These values needed to be either removed or replaced 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 to valid 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 8,517 records inserted/updated into the table.

For the bike sharing demand dataset, we didn't have to do any preprocessing as no values were missing or of different datatype. Therefore we inserted it directly to the database. There were 11,346 rows and 12 columns in the complete dataset.

We then integrated Apache JMeter tool with the three databases and ran the test plans for four types of queries - select, insert, update and delete. The test plan was run for 10 threads with a loop count of 10 that means there were 100 requests each for the four operations which was fired on the database and the performance metrics - Average response time, throughput, and average latency - were recorded for comparing the three chosen databases.

JMeter returns the HTML report and relevant files in a folder while using the command line to run the test plan and also write each request metrics to a file which was passed as an argument while executing. Figure 2 shows the report which was saved locally in a folder. This shows all the graphs, overall and with time metrics and statistics and error report as well.


APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.920	**500 ms**	**1 sec 500 ms**	**Total**
0.920	500 ms	1 sec 500 ms	MySQL read Request from walmart data

Requests Summary

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**100**	**0**	**0.00%**	**296.90**	**55**	**2045**	**148.00**	**1086.50**	**1700.20**	**2044.25**	**28.24**	**16299.99**	**0.00**
MySQL read Request from walmart data	100	0	0.00%	296.90	55	2045	148.00	1086.50	1700.20	2044.25	28.24	16299.99	0.00

Figure 2. Report generated by the JMeter CLI

To summarize, the methodology followed for comparing the databases is:

1. Define the configurations
2. Install the databases
3. Decide the datasets to be used
4. Preprocess the dataset so that they can be imported to the test plans properly
5. Write test plans using the Apache JMeter GUI
6. Run the test plans using the CLI
7. Analyze the HTML report obtained using the Apache JMeter and compare the results.

3.1 Configuration used for experiments

For all the experiments, we used the following configurations:

- Max Number of Connections (threads): 10
- Time between eviction runs: 60000 ms
- Max Wait: 10000 ms
- Loop count: 10

3.2 Lookup

For Walmart dataset, we will be using the following select query for loopkup metrics in MySQL: select * from walmart where IsHoliday=FALSE;
The filter criteria would be same for other two databases as well. Here **walmart** is the table in database **cs532** and IsHoliday is the field in the table.

For bike sharing dataset, we will be using the following select query for lookup in MySQL: select * from bike_sharing where season=1;
Here **bike_sharing** is the table in database **cs532** and season is the field in the table. Note, for MongoDB the lookup query was essentially the same, we directly searched for docuents where the IsHoliday field was false.

3.3 Insert, Update, Delete

We will be randomly generating the field values and inserting/updating them with LIMIT as 1. For delete query, we use those same randomly generated values as filters and delete 1 row at a time.

4 Experiment

The project goal was to run the lookup, insert, update and delete query on all the three databases for finding the response time, latency and throughput. In this section, we will be discussing the results from all the three databases. We will be first discussing the benchmarking configuration along with the version of tools used.

4.1 Tools Version and Benchmarking Configuration

Table 1 shows the configuration of the system used for benchmarking.

Entity	Value
Operating System	macOS Ventura v13.0.1
RAM	8GB
Chip	Apple M1

Table 1. Banchmarking configuration

Table 2 shows the configuration of the tools and databases used for benchmarking.

Entity	Version
Apache JMeter	1.8.0
MySQL	8.0.31
SQLite	3.39.3
MongoDB	1.6.1

Table 2. Banchmarking configuration

4.2 MySQL with JMeter

Figure 3 shows the average response time with the number of active threads for lookup query with MySQL.

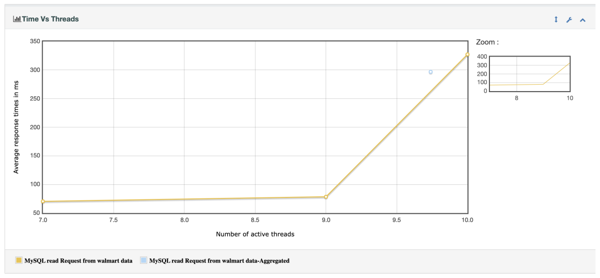


Figure 3. Results for select query with MySQL with the threads

Figure 4 shows the response time of requests for the lookup query in MySQL. Figure 3 and Figure 4 are obtained from the report generated by the JMeter CLI.

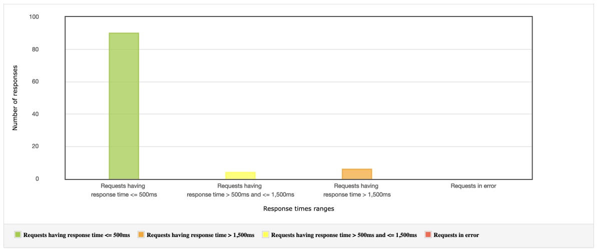
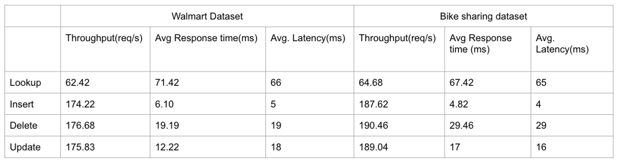


Figure 4. Response time of requests for select query with MySQL

Figure 5 shows the metrics obtained for all the CRUD operations on both of the datasets. We can see the minimum throughput was obtained for lookup queries while insert, delete and update queries gave almost same value of throughput.



	Walmart Dataset			Bike sharing dataset		
	Throughput(req/s)	Avg Response time(ms)	Avg. Latency(ms)	Throughput(req/s)	Avg Response time (ms)	Avg. Latency(ms)
Lookup	62.42	71.42	66	64.68	67.42	65
Insert	174.22	6.10	5	187.62	4.82	4
Delete	176.68	19.19	19	190.46	29.46	29
Update	175.83	12.22	18	189.04	17	16

Figure 5. Results for CRUD queries with MySQL

4.3 MongoDB with JMeter

JSR223 Sampler was used for creating the groovy scripts for read, insert, update and delete. It's imporant to note that the JSR223 sampler does not support the latency metric, and so for MongoDB we can only provide the throughput and average response time for the CRUD operations.
Figure 7 shows the response time of all the CRUD queries fired on MongoDB for walmart dataset.

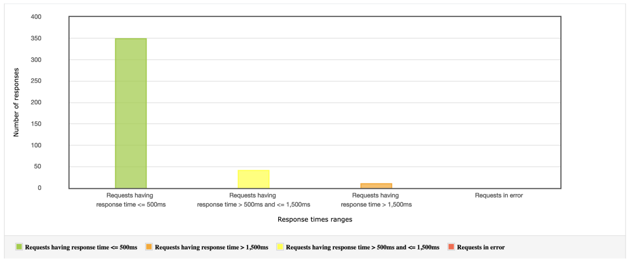
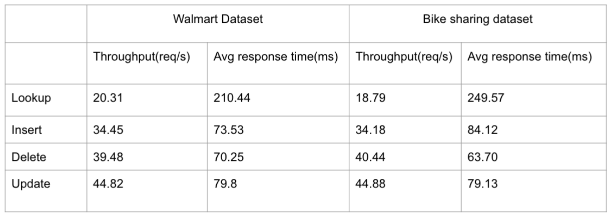


Figure 6. Results for CRUD queries with SQLite

Figure 7 shows the metrics obtained for CRUD operations using MongoDB. We can see from the table that for both the datasets, the least throughput was seen for lookup queries while the maximum throughput was for update queries.



	Walmart Dataset		Bike sharing dataset	
	Throughput(req/s)	Avg response time(ms)	Throughput(req/s)	Avg response time(ms)
Lookup	20.31	210.44	18.79	249.57
Insert	34.45	73.53	34.18	84.12
Delete	39.48	70.25	40.44	63.70
Update	44.82	79.8	44.88	79.13

Figure 7. Results for CRUD queries with MongoDB

4.4 SQLite with JMeter

Figure 8 shows the response time of all the requests of CRUD operations in case of Bike sharing dataset in SQLite.

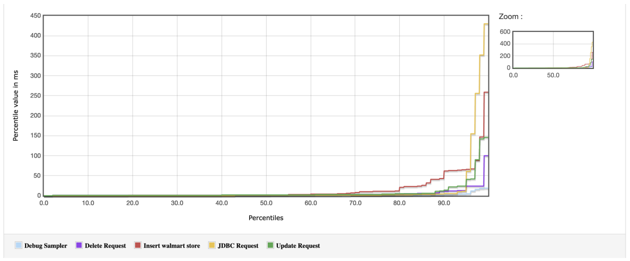
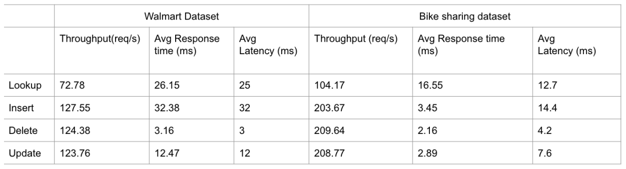


Figure 8. Response time of CRUD queries with SQLite

Figure 9 shows the SQLite metrics obtained for all the CRUD operations on both of the datasets. We can see the minimum throughput was obtained for lookup queries while insert, delete and update queries gave almost the same value of throughput.



	Walmart Dataset			Bike sharing dataset		
	Throughput(req/s)	Avg Response time (ms)	Avg Latency (ms)	Throughput (req/s)	Avg Response time (ms)	Avg Latency (ms)
Lookup	72.78	26.15	25	104.17	16.55	12.7
Insert	127.55	32.38	32	203.67	3.45	14.4
Delete	124.38	3.16	3	209.64	2.16	4.2
Update	123.76	12.47	12	208.77	2.89	7.6

Figure 9. Results for CRUD queries with SQLite

5 CONCLUSION

5.1 Comparison between MySQL and SQLite

SQLite is an transient embedded database that means the data is stored on the application side making it portable and easy to move. No other application on a network has access to the database. SQLite is majorly used to run lightweight and small applications, and so if more data is stored, it will be less efficient and thus it will have worse performance whereas MySQL is better for large dataset as client has to connect to the server to access the data. Thus SQLite will theoretically perform better than MySQL.

5.2 Comparison between MongoDB and SQLite

MongoDB is document based database, making it not rigid with the structure of the data being stored in the collections. It can accept large amounts of unstructured data much faster than relational databases due to slave and master replication.

5.3 Expected Conclusion

MongoDB should perform better than MySQL, and SQLite SQLite should perform better than MySQL in the case of small data size.

5.4 Actual Conclusion

But as the size of datasets was not that big, we can see through the Figure 5, Figure 7, Figure 9 that SQLite performed better than MongoDB in terms of throughput, response time and latency as it can easily store the data in the memory which is faster to access than storing in other application or server.

For future works, more databases should be explored such as ElasticSearch, Cassandra and PostgreSQL. Similarly, more intensive and complex datasets can be used for performance testing of these databases.

6 TEAM CONTRIBUTION

6.1 Aditi Mittal

6.1.1 Tasks Responsibilities

1. Integrate JMeter with MySQL and document the steps to reproduce the results
2. Record results on both datasets
3. Contribute in report

6.1.2 Tasks Completed

1. Contributed in creating the presentation slides.
2. Integrated Apache JMeter with MySQL and recorded results for all CRUD operations on both the datasets and documented the steps to reproduce the results.
3. Integrated Apache JMeter with SQLite and recorded results for all CRUD operations on both the datasets and documented the steps to reproduce the results.
4. Integrated Apache JMeter with MongoDB for read query on walmart dataset as Christopher was facing issue and helped him in debugging the same. This included getting familiar with groovy and bulk insertion in Mongo. Ran all the test plans created by Christopher for both the datasets on my system to get the visuals and statistics for the final report results.
5. Written most of the final report.

6.2 Christopher Gomez

6.2.1 Tasks Responsibilities

1. Integrate JMeter with MongoDB and document important steps
2. Record results for both bench-marking datasets
3. Contribute to the reports/presentations

6.2.2 Tasks Completed

1. Wrote majority of presentation slides and acquired many of the utilized images.
2. Recorded the entire/complete presentation recording. Then edited/uploaded to YouTube.
3. Integrated JMeter with MongoDB for all CRUD operations for both the walmart and bike-sharing dataset. This required learning Groovy (for the JSRR23 Sampler) and the Mongo-Java interface.
4. Recorded results (without aesthetic visuals) for both datasets.
5. Included description and required files for running JMeter with MongoDB (BSON, MongoDBJavaDriver). Rewrote JMeter test plan to be more descriptive/easily-understandable.
6. Contributed to this final report. Added significantly to the intro, technical approach, MongoDB performance, MongoDB/SQLite/MySQL comparison, and the ”actual conclusion” sections of the final report. Followed by extensive editing/proofreading of other sections.

6.3 Mehul Ramaswami

6.3.1 Tasks Responsibilities

1. Integrate JMeter with SQLite
2. Pre-process the Datasets
3. Contribute in report

6.3.2 Tasks Completed

1. Pre-processed the Walmart Recruiting Dataset by removing cells with ”NA” values and replacing with appropriate data-types to result in clean data.
2. Added/edited presentation slides.
3. Tried to integrate Apache JMeter with SQLite with limited success due to repeated laptop system failures.
4. Edited contents and added some technical aspects to the report.

REFERENCES

Armstrong, T. G., Ponnekanti, V., Borthakur, D., and Callaghan, M. Linkbench: a database benchmark based on the facebook social graph. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 1185–1196, 2013.

Halili, E. H. *Apache JMeter*. Packt Publishing Birmingham, 2008.

Keshavarz, S. Analyzing performance differences between mysql and mongodb. 2021.

Kholod, S.-V. Performance comparison for different types of databases. 2021.

Kiran, S., Mohapatra, A., and Swamy, R. Experiences in performance testing of web applications with unified authentication platform using jmeter. In *2015 international symposium on technology management and emerging technologies (ISTMET)*, pp. 74–78. IEEE, 2015.

Nevedrov, D. Using jmeter to performance test web services. *Published on dev2dev*, pp. 1–11, 2006.

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.

Shenoy, S., Abu Bakar, N. A., and Swamy, R. An adaptive framework for web services testing automation using jmeter. In *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, pp. 314–318, 2014. doi: 10.1109/SOCA.2014.36.

Tiwari, A. K., Yadav, S., Mathuria, M., Sharma, M., and Chaudhary, H. Performance optimization of web applications using connection pooling. In *Proceedings of International Conference on Innovations in information Embedded and Communication Systems (ICIIECS'16)*, 2016.

Zaman, F. U., Khuhro, M. A., Kumar, K., Mirbahar, N., Khan, M. Z., and Kalhoro, A. Comparative case study difference between azure cloud sql and atlas mongodb nosql database. *International Journal*, 9(7), 2021.