Abstract

In order to compare PostgreSQL and MongoDB, a benchmark study was created. A real set of data was used. The data is an event log of an incident management process gathered from an audit system of an instance of the ServiceNowTM platform with about 120,000 tickets. Logging tickets and fixes for software projects is commonplace, and addressing these tickets is essential in creating and maintaining an efficient, useful software product. The hypothetical question here is which database a company should use to handle their incident tickets. Given this question, it is essential that the database used can efficiently support both read and write commands to expedite fixes. Each ticket has a variety of attributes like the status (text), whether or not it is resolved (boolean), id (text), and number of incident changes until that moment (int). A variety of commands were performed. Ultimately, PostgreSQL was the better option but the following discussion clearly describes the methods, results, and conclusion.

Literature Review

A variety of studies have been conducted on comparing PostgreSQL and MongoDB in a variety of ways. These studies have tested performance on data storage usage, time, transactions, and adoption. Overall, PostgreSQL has the highest rate of adoption both in all developers (46.55%) and professional developers (49.09%) whereas MongoDB has adoption rates of 25.52% and 25.66% of all developers and professional developers respectively (Stack Overflow, 2023).

In a test done by the company, OnGres, it was found that Postgres was between 4-15 times faster than MongoDB in a transaction performance test. Their experiment was conducted on online transaction processing, similar to our experiment, functions such as inserting, updating, and reading are tested. 3 different benchmarks were tested: transactions, online transaction processing (OLTP), and online analytical processing(OLAP). Two of the queries run in the experiment used to model query an airline reservation system: 1. Insert into a seats table a user id (random), the schedule relevant data and the date (randomly generated). 2. Upsert (INSERT ... ON CONFLICT DO UPDATE ... for PostgreSQL) the corresponding entry in the audit table that contains schedule, date and number of seats occupied. In all of the benchmarks, it was clear that Postgres was outperforming Mongodb both in faster performance and lower latency.

Methods

For both databases, multiple functions were created to test the performance for time and memory for read, updates, and inserts. For every function, both the time taken and the memory is being tracked and then logged. For memory we calculate the memory before the query and after the query, we apply the same logic to time taken and calculate the time it takes for the function to run. Two experiments were run in our research. The first compared MongoDB and PostgreSQL over reads, inserts, and updates of different sizes: 10, 100, 1000. The second simulates a specified number of days in the company with incident logs. A simulation of 100 incident log database interactions were conducted 100 times.

The second part of the study involves a simulation. The simulation attempts to model how well the databases perform over 100 days. Data is randomly generated and selected. To begin, it is assumed that there are 100 incident ticket operations and that on any given day there are on average 80% read operations, 10% update operations, and 10% insert operations. With these weights, 100 transactions are selected with randomly selected incident ids. The distribution of read, update, and insert operations for each “day” in the simulation is selected such that the number of operations is 100 but the actual number of each varies for each day in the simulation. The number was randomly selected for each 100 times with those as the weights. Then the memory and time components were measured and aggregated over the 100 simulations.

Results

The results from the first study are in Appendix A and B. Further results from the simulation are in Appendix C. Overall, we see that with time taken for both reading and updating, Postgres clearly takes less time especially in the larger sizes. In the inserting function, the two databases are very similar and Mongo is just barely faster for every size. Overall, from both experiments it is clear that PostgreSQL is more time efficient. While in the simulation, it is less clear, from the other benchmark studies it is clear that as the number of transactions increases, it is more time efficient to use PostgreSQL.

An important note is that as the number of transactions increased, the amount of memory required decreased for both database systems. It is unclear to us why that is the case. It seems like it was a way that memory was measured using the `memory_info` from the psutil.process library. It likely has something to do with garbage collection and perhaps as the data increased, RAM was required instead of just disk memory. We researched this and used other ways to measure memory but resulted in this notable trend in the different methods as well.

Conclusions

Ultimately, this research shows that for a type of data that is strongly typed, a strong schema and relational database like PostgreSQL is much better than MongoDB, a document oriented database. It is faster, and in a situation like this, time is essential for the business to timely address complaints. A limit of this study is that both databases were hosted locally. This is unlikely for any enterprise, and a further study would include deploying these databases and services to the cloud to test. This would also bring in additional cost. This study was free because free versions of both databases were used.

For the purposes of comparing cost, the Azure pricing models for PostgreSQL and Azure Cosmos DB, which supports MongoDB, was explored. In general, Azure's PostgreSQL may be more cost-effective for traditional relational databases with moderate traffic, while Azure's CosmosDB is better for high-performance NoSQL workloads that require global distribution and scalability. An example monthly cost breakdown is below (Microsoft, n.d):

PostgreSQL (Azure Database for PostgreSQL Flexible Server) Costs:

- Compute: $62.55/month (for 2 vCores)
- Storage: $102.40/month (for 1 TiB)
- Total Cost: $164.95/month

MongoDB (Azure Cosmos DB MongoDB API) Costs:

- Compute: $86.40/month (for 400 RU/s)
- Storage: $256/month (for 1 TiB)
- Total Cost: $342.40/month

Ultimately, given the cost and time constraints, PostgreSQL is recommended for this use case.

Additional Notes

Implementing the benchmark took several hours, but it took longer to run the benchmark as the number of transactions increased. If an enterprise were running the second simulation, they would probably want to use more days in the simulation or even more ticket transactions per day. This would be company specific. For a company like Microsoft, they need many more ticket transactions than 100, but for a smaller company that may suffice. The code can easily be changed to address these needs.It did not take long to bring the data into the systems. After downloading the data, setting up both databases took about 10 minutes. While PostgreSQL is ultimately faster, setting up MongoDB is much easier because it has a more flexible schema set up. Thus, I was able to set up the table and load the data with one command in the command line. For PostgreSQL, a script was required and the whole schema had to be defined. The documentation for both tools is complete and easy to follow on a variety of machines. Because both are widely used, there are also a considerable number of resources for fixing any issues when setting up and using the database.

Depending on the type of transaction, it was easy to access the data via code once the database was set up. They each took a different amount of time, but inserts were fairly quick because nothing needed to be obtained from the database in order to do an insert. When reading and inserting, the record needed to be located first then read or updated.

It was fairly simple to set up a smaller benchmark study like this one. If these databases were deployed to the cloud or utilized Docker, troubleshooting and accessing the databases would have been more difficult. It is difficult to work in these databases across systems. For example, while several members of the group were able to set up and use the database easily, one was not. This shows that while docker and the cloud might be harder to set up, they are necessary when sharing across multiple developers and systems.

Appendix A

Table 1: Time Measurements (in seconds)

	Mongo			**PostgreSQL**		
Records	**Read**	**Update**	**Insert**	**Read**	**Update**	**Insert**
10	1.4174	1.3944	0.0048	0.1016	0.0762	0.0129
100	9.1636	9.4713	0.0066	0.4148	0.4182	0.0194
1,000	89.5847	92.2216	0.02	3.8588	3.8313	0.0747
100,000	(30+ min)	N/A	N/A	349.7355	471.4855	3.6093

Appendix B

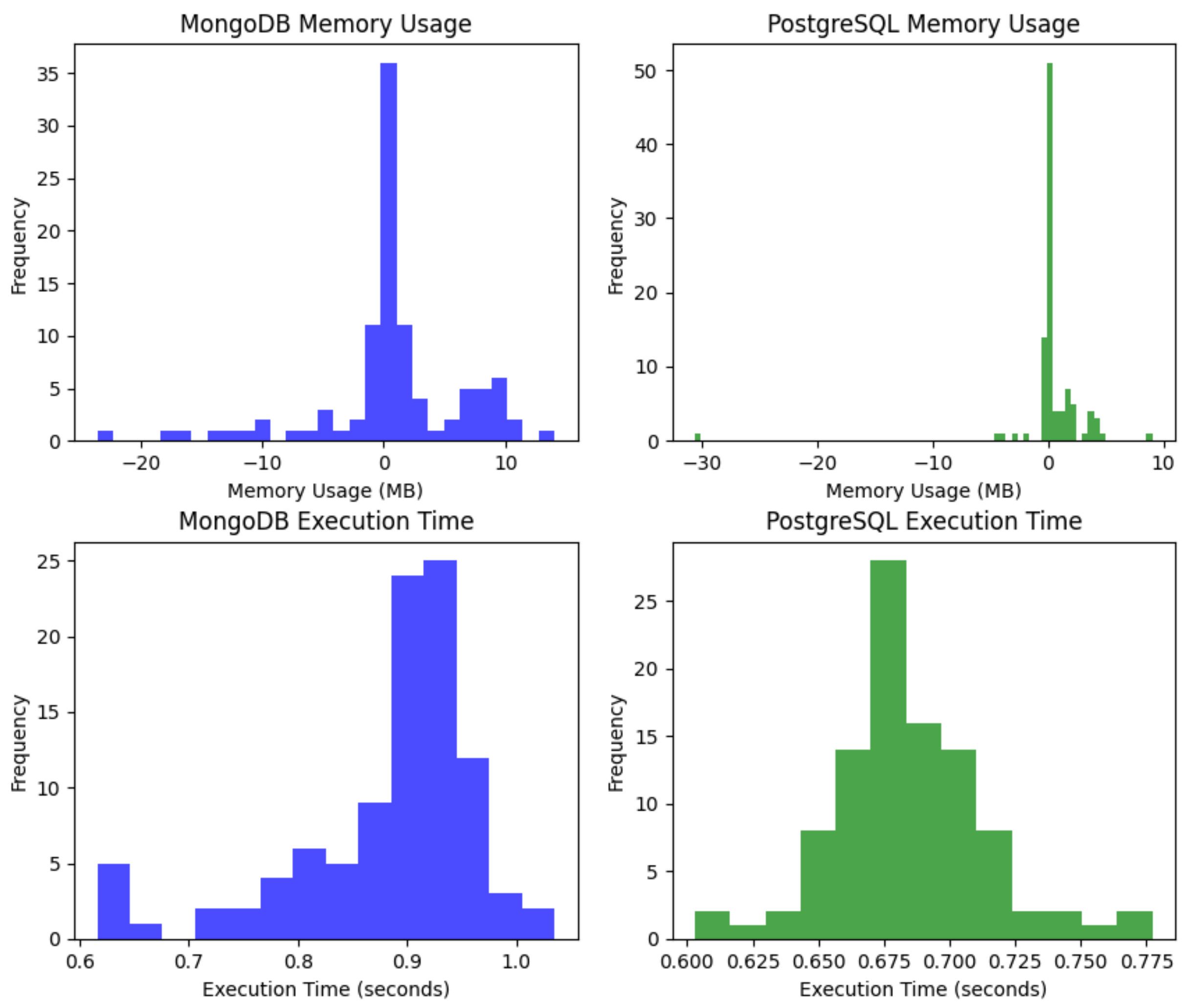
Table 2: Memory Measurements (in MB)

	Mongo			PostgreSQL		
Records	**Read**	**Update**	**Insert**	**Read**	**Update**	**Insert**
10	13.40625	9.890625	0.078125	6.546875	6.046875	0
100	8	9.28125	0.09375	1.46875	2.125	0
1,000	-39.703125	0.515625	0.5	1.09375	2.390625	0.015625
100,000	N/A	N/A	N/A	-30.28125	-25.484375	2.96875

Appendix C

Command Line Output:

```
MongoDB Execution Time - Mean: 0.8821, IQR: 0.0742
PostgreSQL Execution Time - Mean: 0.6844, IQR: 0.0325
MongoDB Memory Usage - Mean: 0.6252, IQR: 3.0703
PostgreSQL Memory Usage - Mean: 0.3237, IQR: 1.1836
```



References

EnterpriseDB. (2019). *New Benchmarks Show Postgres Dominating MongoDB in Varied Workloads*. Retrieved November 17, 2024, from https://www.enterprisedb.com/news/new-benchmarks-show-postgres-dominating-mongodb-varied-workloads

Hernández, A.Cermeno, F., Calvo, E., Herzig, G., Ostapowicz, S., Melli, M., Fernandez, J. (2019). *PERFORMANCE BENCHMARK POSTGRESQL / MONGODB.* Retrieved November 17, 2024, from https://info.enterprisedb.com/white-paper_Get-the-Postgres-and-MongoDB-report.html

Kaggle. (2020). *Incident Response Log Dataset*. Kaggle. Retrieved from https://www.kaggle.com/datasets/vipulshinde/incident-response-log?resource=download.

Microsoft. (n.d.). *Azure Database for PostgreSQL - Azure PostgreSQL pricing*. Retrieved from https://azure.microsoft.com/en-us/pricing/details/postgresql/server/.

Microsoft. (n.d.). *Azure Cosmos DB for NoSQL - Azure Cosmos DB pricing.* Retrieved from https://azure.microsoft.com/en-us/pricing/details/cosmos-db/autoscale-provisioned/.

Stack Overflow. (2023). *Most popular technologies: Database professionals*. Retrieved November 17, 2024, from https://survey.stackoverflow.co/2023/#most-popular-technologies-database-prof