Analysis and Benchmarking of Threaded Applications on EVA and Neo4j

09.28.2021

Abhay Goel

Pranay Methuku

Prerna Agarwal

Overview

Different types of data exist in the world that are used by Applications. Some is about people and is used by social media applications, while some is more general, and enhances the knowledge of the individuals. The goal of this project is to feed various types of such data into two database systems: Relational Databases and Graph Databases. Then, we would run some transactions on the two databases in single-threaded and multi-threaded fashion, to return a total of four observations:

1. Single-threaded RDBMS load
2. Multi-threaded RDBMS load
3. Single-threaded Graph DB load
4. Multi-threaded Graph DB load

Finally, these observations would be compared and the results would be justified.

Technical Details

This section discusses the platforms used during the project. We will build Python applications that would connect to the two database systems. The Python applications would execute single-threaded and multi-threaded loads on the databases, and would be responsible for timing the responses. The voice of database systems is crucial as they are what the project depends on. We have chosen EVA as the Relational Database and Neo4j as the Graph Database of choice.

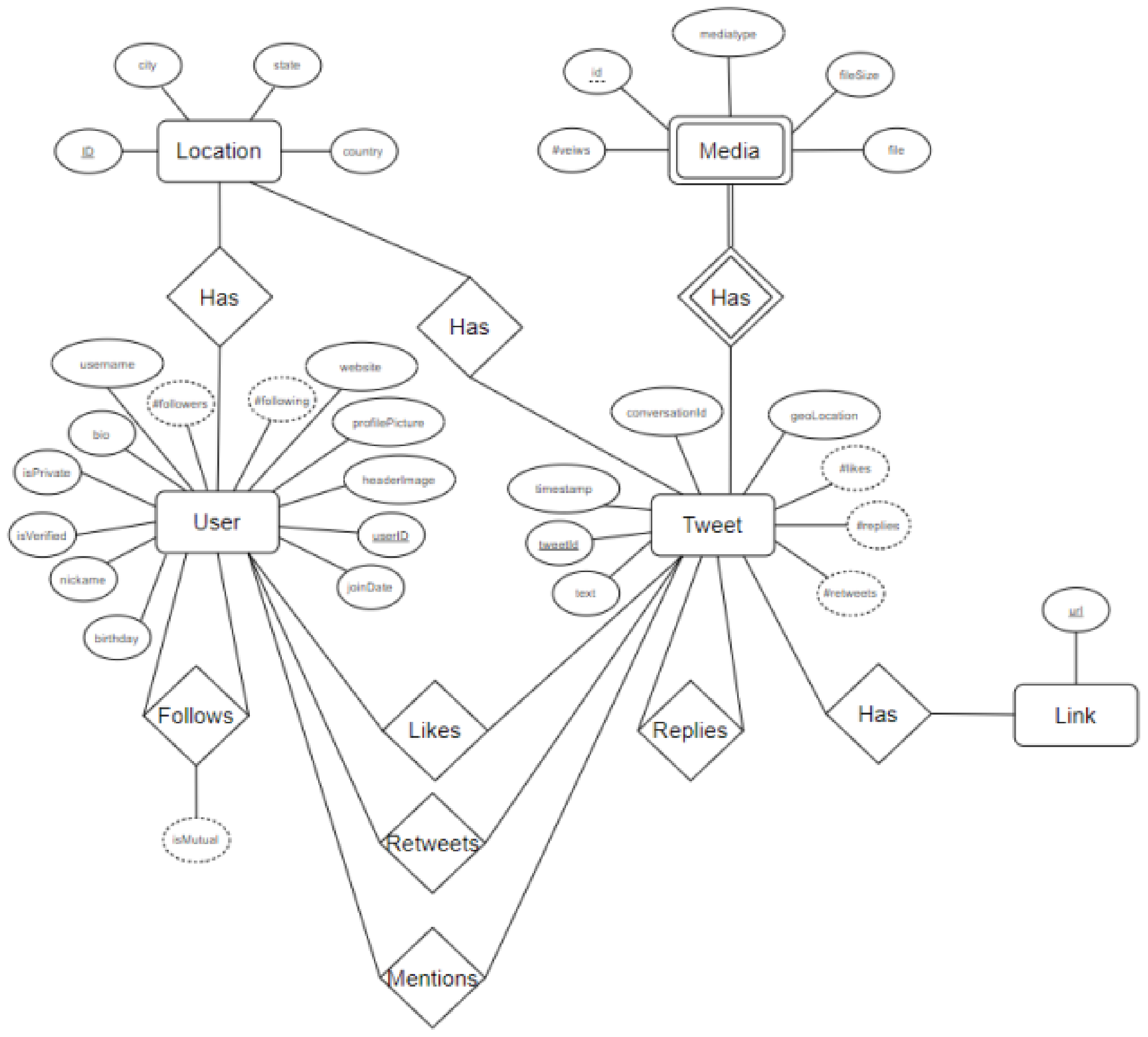
To summarise, these are the platforms of choice:

1. Python, EVA and Neo4j environment
2. EVA for Relational Database
3. Neo4j for Graph Database

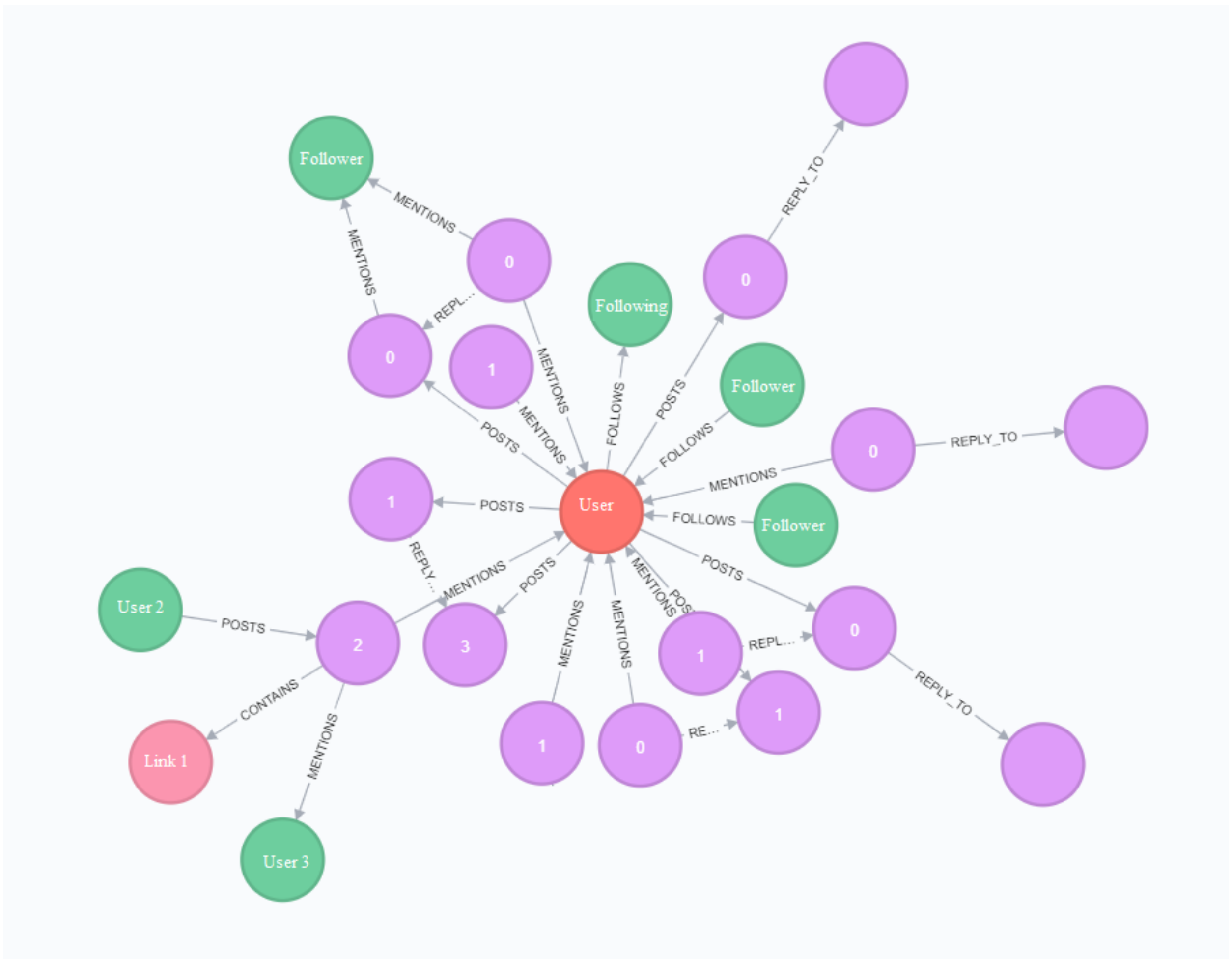
Specifications

The data we would be using would be different for different types of databases. For the graph database, it would be fetched from the Open-Source APIs available at Twitter. This data would be from 2013, and would consist of Users and their Tweets. It would also consist of the likes and comments statistics that each tweet received.

This data would be perfect as it can reflect the performance in the Graph database. The following is the Entity-Relationship diagram of the data that we would use:



And below is how we plan to structure our data in Neo4j:



On the other hand, for the relational EVA database, we are planning to use the UA-DETRAC dataset since it is already available in the EVA repository. According to their website, “UA-DETRAC is a challenging real-world multi-object detection and multi-object tracking benchmark. The dataset consists of 10 hours of videos captured with a Cannon EOS 550D camera at 24 different locations at Beijing and Tianjin in China. The videos are recorded at 25 frames per seconds (fps), with resolution of 960×540 pixels. There are more than 140 thousand frames in the UA-DETRAC dataset and 8250 vehicles that are manually annotated, leading to a total of 1.21 million labeled bounding boxes of objects.” The data at this scale should be good enough to operate on for the scope of this project.

Schedule

For the timeline of the events necessary for this project, we haven't specifically divided the workload between the teammates, but we have an idea of how the division would be over the semester.

During the course of this project, we would be achieving the following:

Weeks	Task	Task Owner (tentative)
1-2	Set up the development environment, get the data, feed it to the systems.	Abhay, Pranay, Prerna
3-4	Write all the queries, make a test-bench, connect the databases to Python applications.	Abhay, Pranay, Prerna
5-6	Run the single-threaded and multi-threaded loads on both the databases, obtain the results, and form justification.	Abhay, Pranay, Prerna
7-8	Prepare for presentations and demos, add new functionalities at the lower level, if time permits.	Abhay, Pranay, Prerna

Deliverables

1. Python application(s) that acts as the base of the project
2. A file of all the data fed to the systems (.csv, .sql, and .mp4)
3. Neo4j downloaded files
4. Instructions to set up all these platforms
5. Project Presentation and proposal

Future Work

If time permits, we would be changing the internal buffer management policy of both the database management systems. Also, we would be interested in looking into the operators like insert, update, delete used in these database systems, and adding new functionalities for them, at a lower level. On top of all this, we are interested in including some query optimization. Lastly, on the DBMS side, we can test these systems on how they handle fault-tolerance.

References

1. Holzschuher, F., & Peinl, R. (2014). Performance optimization for querying social network data. In EDBT/ICDT Workshops (pp. 232-239).
2. Fernandes, D., & Bernardino, J. (2018, July). Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB. In Data (pp. 373-380).
3. Gupta, S. (2015). Building Web Applications with Python and Neo4j. Packt Publishing Ltd.
4. L. Wen, D. et al. (2020). UA-DETRAC: A new benchmark and protocol for multi-object tracking. CVIU, 2020.
5. Georgia Tech Database Group - EVA. https://evagatech.readthedocs.io/