# Twitter Hashtag Traversal

1st Author
Prashant Kumar
Graduate student, Computer science
Iowa State University
(515) 708-5955
pku@iastate.edu

2rd Author
Shubham Agrawal
Graduate student, Computer science
Iowa State University
(515) 708-4807
sagrawal@iastate.edu

## ABSTRACT

*Graph database is constantly getting popularity for storing social interconnected data, because of its high performance and easy to implementable design. We are trying to use Twitter data and using hashtag traversal comparing the performance for different size of database. In this we have designed a graph database model and stored different amount of messages and related information. Different type of queries has been designed and tested on these messages. Cypher query language is used for this, as its very powerful and easy to understand query language for graph database. We have presented performance and compared those in the conclusion. It was seen that no fixed pattern has been observed for measuring the performance of the graph database based on size of the datasets. In last we discussed that in most cases the performance of graph database changed marginally with the increase in the size of datasets.*

## Keywords

Graph databases, Neo4j, cypher query, performance, twitter, hashtag traversal, political data.

## 1. INTRODUCTION

Bing amount of structured data is being stored using Relational databases for decades, as they support ACID properties and help giving high performance. But in the recent years there has been great increase in data size and data complexity because of numerous social services available. To address this many NoSQL database have emerged. Most popular of these is Neo4J. This is a Java-based open source database, which uses graph structure. It has labeled nodes connected by properties, which are key-value pair. Cypher, Neo4j's query language, which is declarative and a bit similar to SQL, is a user-friendly language that is designed to be read and understood easily.

Available data on social networks is highly interconnected, e.g., networks of people, comments, ratings, tags and activities. They are forming acquaintance networks, communication networks and topic networks, just to name a few. Relational database will need high number of many-to-many relations to store this data. We will require complex join operations such data. Graph databases, on the other hand, are specially designed to store such kind of interconnected data and to deliver high performance, traversing them.



Here we are interested in finding how information travels from one entity to another. An entity of interest can be a person, an institution, a state, or a country. Here we will test Neo4j, with cypher query language, for different type of tasks and will check how its performance changes with the size of the database.

We are using twitter as our data source. Twitter, a social networking service, has evolved as information source along with social networking. In this User can read, post and share messages, called tweet. It has 284m monthly active users who send 500m tweets per day. These tweets have a connection property called hashtag, #hashtag, which helps collecting tweets together with same hashtag. Out of tweets related to many fields we have picked tweets related to politics. The reason behind this is that it's a good information source for election results prediction.

We have designed two graph database models to store the data in the database. For both the models some cypher queries has been designed and ran using Neo4j terminal. Each query is defined in a way so that it can use different features of graph database. Some are basic queries and some are complex one. A script file is created to run all the queries together. Time for each query is recorded 30 times and then it is used to plot the performance chart. Bases on this we decide how performance changes with size of graph database for different tasks.

The paper is organized as follows: in next section related work is presented briefly before we explain the test setup and sample data in section 3. Next section, 4, gives a comparison of performance for different task. We present the benchmark results and analyze them in detail in section 5. Finally, we sum up the learning in section 6.

## 2. RELATED WORK

### 2.1 Twitter characterization

[1] Social network communication can be classified mainly in two categories "social" and "informational". But Twitter is different from these two categories, when we consider hashtag on political topics as our criteria. Results show that hashtag-centered reply and retweet networks in this domain do not fall clearly into the social or informational categories. There appears to be a third kind of network associated with political debate.

In this paper for their study they have used retweet, reply and mention networks in political hashtag spaces to explore to what extent they show the characteristics of social or informational networks.

For data, around 2400 tweets, related to 2012 U.S. Senate election in Hawaii, were collected. All of the tweets included the hashtag "HISen".

They analyzed local and national political hashtag spaces, where Twitter users discuss political topics of shared interest by

tweeting. They concluded that Twitter networks exhibit characteristic of social network in some ways but for some cases it don't. So it does not fit into social vs. informational dichotomy. Moreover even a particular type of network – a political topic network on Twitter – can display different social or informational characteristics depending on the topic of interest and the actors involved. Instead of categorizing networks, researchers should consider how various features of a socio-technical system afford both information sharing and socialization.

So this is the reason behind choosing political data for our project. We have taken some political hashtags like "Obama", "ELECTION2014" etc. As for future we can predict about election results based on the tweets related to the hashtags of that time. Also we can find a relation between, activeness of a party on Twitter and result of the election. [2] As previous record shows that for 2012 election the party who won the election has shown better presence on Twitter compared to competitor party.

## 2.2 Performance of graph query languages

[3] In this paper author describes his experiences with a different back-end based on the graph database Neo4j and compare the alternatives for querying data with each other and the JPA-based sample back-end running on MySQL. Moreover, why the different approaches often may yield such diverging results concerning throughput, is analyzed. The results show that the graph-based back-end can match and even outperform the traditional JPA implementation and that Cypher is a promising candidate for a standard graph query language.

They stated that Cypher supports 5 out of six types of graph queries. The first type is adjacency queries, which test whether two nodes are connected or are in the k-neighborhood of each other. The second type is reachability queries that test whether a node is reachable from another one and further more, which is the shortest path between them. The third type is pattern-matching queries and especially the sub-graph isomorphism problem, as only this one can be solved in finite time. The fourth type is summarization queries that allow some kind of grouping and aggregation and fifth is summarization queries, which has been fixed in Neo4j version 1.8, which supports count, sum, max, min and avg aggregation functions.

To get the data for this project, they have used realistic sets of sample data in terms of complexity and size. For this they wrote a generator that is able to create data sets with random person, organization, message and activity data based on anonymized friendship graphs and store it in an XML file. This resulted dataset that contains 2011 people, 26,982 messages, 25,365 activities, 2000 addresses, 200 groups and 100 organizations.

Cypher queries are constant strings, so they can be cached by the database as compiled queries. Depending on the query, results can be nodes with all attributes, individual attributes or aggregated data. Like SQL, Cypher is not only a query language but also allow data manipulation like updates and deletes.

In their paper, with Neo4j and Cypher, they were able to achieve performance improvements of one order of magnitude for queries that span multiple tables. Although Cypher didn't perform very well itself in these tests compared to Gremlin and native object access. However, expectation is that this problem will be addressed in Neo4j version 1.9 which will be optimized for providing high performance Cypher query results.

Overall they have concluded that Neo4j can be used as a high performance replacement for relational databases, especially when handling highly interconnected data as in our social Web portal. Using an embedded Neo4j instance can yield very good performance when using data sets of limited size.

# 3. PROPOSED WORK

In this project, twitter hashtag traversal, we have developed an application to collect twitter messages related to President Barack Obama on twitter using twitter search API. We have fetched twitter messages based on geo locations of the US cities (source: http://sujee.net/tech/articles/geocoded/). In our data model, (see Figure 1) for each twitter message a new node is created in graph database and it's tweet id, text, created_at, retweeted and retweet_count information is stored as the property of the node of type Tweet. Also, for each node type of Tweet, 4 new nodes of type Hashtag, Source, User and State are created. In node type Hashtag, the name of the hashtag used in the tweet is stored. In source type node, information about the device from which the tweet was sent is stored. For User node, the name of the user, user location, followers count, count of the profile the user is following and whether this user is verified or not is stored. In state type node, we have stored the name of the state from where the tweet was created.

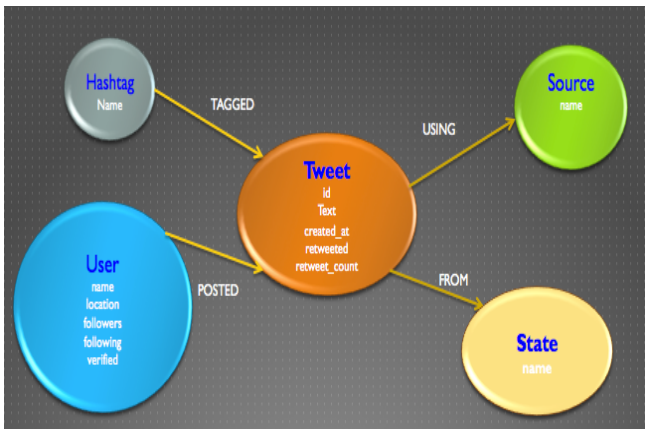


**Figure 1. Graph Database Model**

Read performance of the model using cypher queries was recorded for the following tasks:

1. Return top K most retweeted tweets.
2. Return top K hash tags that travel the most number of states.
3. Return the longest path of states that a particular hashtag travels within a given time period.
4. Show contents of the most retweeted tweet for a hashtag Y and the location X where it was posted originally.
5. Return users and their state, which have used hash tag A and also used hash tag B and hashtag C in their tweets.
6. Return max k platform used for tweeting of a certain hash tag.
7. List the top tweeting user for location X for a hashtag Y.
8. Return the maximum count of tweet using certain hash tag, by a user who has maximum followers.
9. Return the verified users and their followers and friends count for a particular hashtag.

The total time taken for the execution of these queries was recorded on 3 different datasets: D1 with 1293 nodes and 4699 edges, D2 with 10718 nodes and 34041 edges and D3 with 19805 nodes and 70653 edges (see Table 1). Each query was run 35 times on these datasets and average time was calculated using the 30 recordings and first 5 running time of the queries was not used for calculating the average performance.

| Dataset | No. Of Nodes | No. Of Edges |
|---|---|---|
| D1 | 1,293 | 4,699 |
| D2 | 10,718 | 34,041 |
| D3 | 19,805 | 70,653 |

**Table 1. Size of Datasets used**

## 3.1 Technologies used

The application was developed using python 2.7.6 and Neo4j 2.0.4, which is the most commonly deployed graph database worldwide. Py2neo 1.6.3 and Tweepy python libraries were used. Py2Neo is a library to interact with Neo4j whereas Tweepy is a library for interacting with the twitter search API.

## 3.2 Testing environment

All the queries were executed on Mac machine with 4 GB 1333 MHz DDR3 memory and 2.3 GHz Intel Core i5 processor on OS X Yosemite 10.10.1 version.

# 4. PERFORMANCE EVALUATION

Some interesting observations can be made from the average time taken for each query on different sets. For return the maximum count of tweet using certain hashtag, by a user who has a maximum followers and return top k hashtags that travel the most number of states, the average time taken by the query kept on increasing as the size of the dataset was increased (see Figure 2). For return top K most retweeted tweets, return the longest path of states that a particular hashtag travels within a given time period and Show contents of the most retweeted tweet for a hashtag Y and the location X where it was posted originally, no pattern was observed for the average time taken by the queries on increasing the size of the dataset (see Figure 3). Whereas for rest of the 4 queries, there was no significant effect of changing the size of dataset on the average time taken to run those queries (see Figure 4). The return top K hashtags that travel most number of states query took highest time to run on all 3 datasets and the query to Return users and their state, which have used hash tag A and also used hash tag B and hashtag C in their tweets, took the least time to run on all the 3 datasets (see Figure 5).

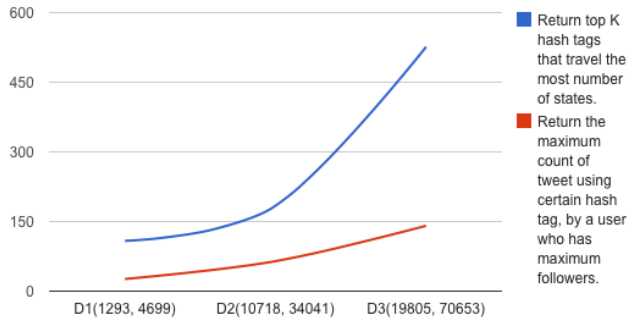


**Figure 2. Increase in average time taken**

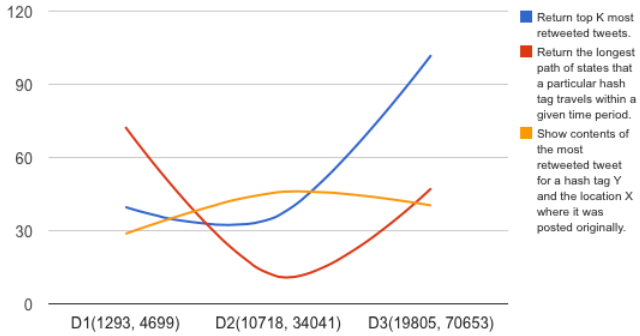


**Figure 3. No fixed pattern for average time taken**

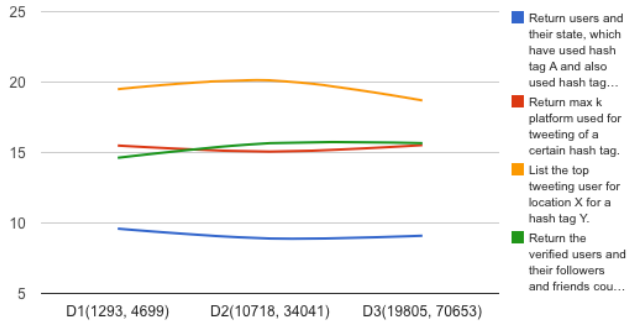


**Figure 4. No significant change in average time taken**

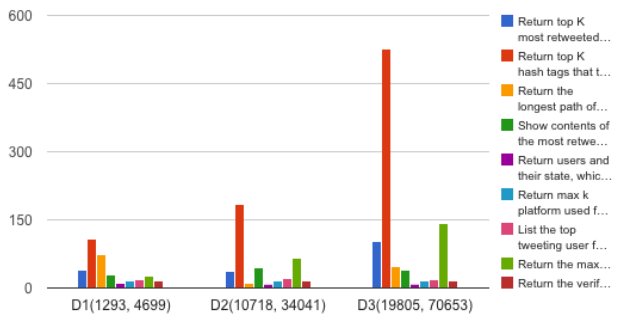


**Figure 5. Average time taken on different datasets**

| Tasks | D1 | D2 | D3 | No. Of output rows |
|---|---|---|---|---|
| Return top K most retweeted tweets. | 39.7ms | 36.17ms | 101.97ms | 10 |
| Return top K hash tags that travel the most number of states. | 108.4ms | 184.2ms | 526.23ms | 10 |
| Return the longest path of states that a particular hashtag travels within a given time period. | 72.5ms | 11.1ms | 47.33ms | 1 |
| Show contents of the most retweeted tweet for a hashtag Y and the location X where it was posted originally. | 28.7ms | 45.8ms | 40.33ms | 1 |
| Return users and their state, which have used hash tag A and also used hash tag B and hashtag C in their tweets. | 9.6ms | 8.9ms | 9.1ms | 1 |
| Return max k platform used for tweeting of a certain hash tag. | 15.5ms | 15.07ms | 15.53ms | 5 |
| List the top tweeting user for location X for a hashtag Y. | 19.5ms | 20.13ms | 18.7ms | 10 |
| Return the maximum count of tweet using certain hash tag, by a user who has maximum followers. | 26.27ms | 64.6ms | 140.87ms | 1 |
| Return the verified users and their followers and friends count for a particular hashtag. | 14.63ms | 15.67ms | 15.67ms | 1 |

**Table 2. Average time taken by the cypher queries**

## 5. CONCLUSION AND FUTURE WORK

As observed from the given charts no fixed pattern was found for measuring the performance of the graph database based on size of the datasets. For nearly half of the queries, time taken to run the query is more or less the same. For some queries, time taken increased and for some it decreased. So, bases on this for now we can draw a conclusion, that in most cases the performance of graph database changed marginally with the increase in the size of datasets and for future work, the queries must be tested on more large datasets to check, if we can find a clear cut pattern for the performance of the graph database.

## 6. REFERENCE

[1] Misa Maruyama, Daniel D. Suthers, Scott 2014. *Characterizing Communication Networks Associated with Political Hashtags* P. Robertson. 47th Hawaii International Conference on System Science.

[2] https://storify.com/Jonzey08/is-involvement-in-social-media-the-keys-to-success

[3] Florian Holzschuher, Prof. Dr. René Peinl. *Performance of Graph Query Languages.* Institute for Information Systems

## 7. APPENDIX

**Project contribution**

| Name of team member | Percentage of contribution | Description of work |
|---|---|---|
| Prashant Kumar | 50 | Contributed in designing and coding |
| Shubham Agrawal | 50 | Contributed in background study and coding |

**Report contribution**

| Name of team member | Percentage of contribution | Description of work |
|---|---|---|
| Prashant Kumar | 50 | Initial part till related work, and remaining report formatting |
| Shubham Agrawal | 50 | Remaining report till conclusion |

**Coding contribution**

| Name of team member | Percentage of contribution | Description of work |
|---|---|---|
| Prashant Kumar | 50 | Database insertion and python mapping |
| Shubham Agrawal | 50 | Implementing database and query design |