**Independent Study Report**

1. **Knowledge Representations and Reasoning using Graph Databases**
2. Introduction to Graph Databases:
3. **Graph Data Modelling and Techniques**

Graph data modeling is better than relation data modeling because of use of cognitive psychology which improvises big data designs.

For doing graph visualization, we need to translate data's conceptual view to a logical model and that is data modeling.

In Graph Modeling, similar to the relational modeling, we use lo-fi methods such as whiteboard sketches and decide on an initial domain.

The domain is described as the connected graph with nodes and relationships. Nodes are entities with a unique conceptual identity. Labels are assigned to the nodes. Relationships are the interactions between two nodes.

The graph database structures are used by graph databases for storage of information. As opposed to other NoSQL databases that stores data in the form of key-value pairs, document –store.

The graph structures consist of the nodes, edges describing relations and other properties that gives more information about the nodes and edges.

Benefits of Graph Data Modelling:

The various advantages of the graph data structures are as follows:

1) Enables multiple entity representations – Represent entities which are not of the same type and depict relationships between them

2) Supports diversified graph structures – Various Directed, multi-graph, undirected graphs, bipartite graphs.

1. **Graph Querying Language and Algorithms:**

A pattern graph, group matching, graph mining, continuous queries, aggregate queries are some of the terminologies with respect to graph database querying. Pattern queries start from a particular node and matched neighbors are visited with their corresponding neighbors that matched the predicate. The group matching queries determines the frequency of the partial or complete group query in the graph where the nodes and edges are taken into the considerations and conversely, the common and notable patterns in the graphs help in graph mining. Continuous queries are trigger-oriented where the new matching patterns in the updated graphs are informed to the user. Aggregate type of queries is used to for calculations over multiple nodes.

**Query Optimization Techniques in Graph Databases**

Since there exist various types of queries for graph databases, the need for the optimized approach becomes essential to address the issues of graph data like highly interconnected data, changing the schema of graph databases and storage of large data sets. Some of the techniques include decompose and incremental query processing, pre-process aggregate queries, sketch the graph cluster for summarizing graphs. These optimized techniques can be implemented based on the type of query being executed.

The data distribution can be achieved for graph databases by partition large set across servers such that the edges are not broken and there is the equal distribution of data across servers. This should enable to increase the response time of the query and, minimize the cost of storage.

**Graph Algorithms:**

I. Pathfinding & Traversal Algorithms

1. Parallel BFS - Used when the tree is less balanced or target is closer to the starting point. Used to find the shortest path between nodes

2. Parallel DFS - Used for deeply hierarchical data. Used when the tree is balanced and the target is closer to an endpoint.

3. Single Source Shortest Path - Calculates a path between the node and all other nodes whose summed value to other nodes are minimum.

4. All-Pairs Shortest Path - Calculates the shortest path containing all shortest paths between the nodes in the graph

II. Centrality Algorithms

1. PageRank - Current node's influence or importance is known or estimated based on its linked neighbors and neighbor's neighbors.

2. Degree Centrality

III. Community Detection Algorithms

Connected Components - Group of nodes are founded where each node in the same group is reachable to other nodes in the same group.

Legis Graph:

This is a graph generated when we load or import US legislative data into Neo4j database.

The data from GovTrack is incorporated into this data model.

After data is loaded we can find different or interesting patterns about Congress using Cypher queries.

1. **Evaluating the graph queries for big data:**

MapReduce and distributed evaluation technique to be used for the evaluation of graph database application.

1. **Querying Big Graphs within Bounded Resources –**

Localized Queries: These are classes of the query where data is said to be local if other nodes of the graph are within the specific number of hops of querying node.

a. Subgraph Queries

b. Simulation Queries

Non-Localized Queries: In this classes of queries, the data is not local since other nodes are not within specified number of hops

Reachability Queries: For a graph, with a pair of querying nodes, it will return true if there is a path between the querying nodes

1. **Evaluation of graph databases performance through indexing techniques –**

In the graph database, to find a specific node faster, the data structure we use is Indexing.

In Neo4j, labels are assigned to nodes and are helpful for creating faster and efficient sub-graphs. Labels primary focus is for indexing. To perform indexing, they do the following:

a) A Cypher Command is used to create the users from social networks like 'Twitter', and a Unique index is based on a unique constraint which will be faster than a standard index

b) A Cypher command is written for all users to follow all other users

After performing experiments on the Twitter dataset, it was found that performance of Neo4j's retrieval queries performs better without using indexing.

1. **Analysis of design techniques for Graph Databases compared with other types of databases:**

Level of Security - Neo4j has improved security foundations for enterprise-level applications.

It provides security functionalities including access levels, security event logging, and securing data. The multiple users are managed by the Neo4j using user-specific names and passwords or LDAP connectors are combined with the Noe4j repository for authentication and verification of the user. The access privileges are based on the users-specific roles. Neo4j provides the facility of aborting a running query and listing the running queries to admin users.

Additionally, admins and other users can fetch the graph database statistics and schema design.

MongoDB Security – Mongo DB provides security at the user level. The access rights and authentication are provided specifically to users.

Thus, Neo4j has better security as compared to MongoDB.

Sharding –

Sharding is a method where data is distributed across the multiple servers. This functionality helps is handling large data sets and improve the throughput of the database operations.

**Sharding in Neo4j:**

Sharding is a key functionality supported by MongoDB. Sharding provides highly scalable solutions to NoSQL database like MongoDB, where sharding for Neo4j high has complexity. Lookup in graph structures is unpredictable because of constantly changing graph structures. The various scenarios like locating most of the connected nodes on one server, locating connected components equally across the servers. Each of the scenarios holds the overhead of heavy load or traversal performance. So the main challenges in sharding for Neo4j graph data are to observe the patterns of relationship, push the data on servers so as to satisfy balanced graph structures and perform optimized query traversal. Below are the sharding instances -

-

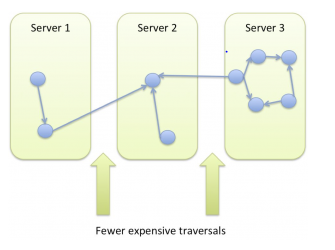
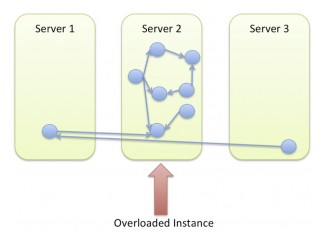


Figure 1. Sharding in Neo4j (Jimwebber.org, 2017)

**Sharding in MongoDB**

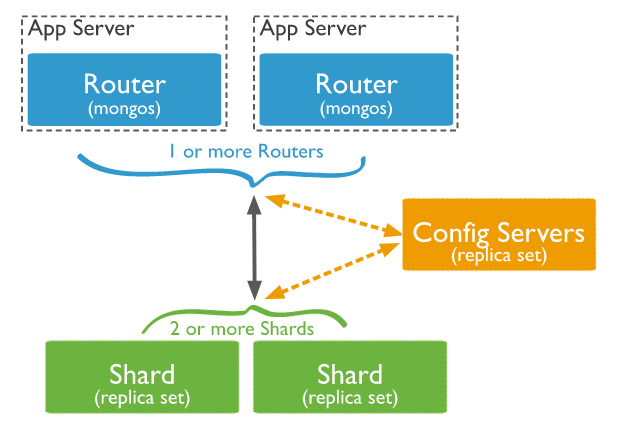


Figure 2. Sharding in MongoDB(Docs.mongodb.com, 2017)

Horizontal scaling can be implemented through sharding for MongoDB. The collection level data are distributed across the systems in the cluster. MongoDB sharding functionality provides the key advantages including distributed read and writes operations for large data sets, increase the storage capacity of the data, enables high availability through the sharded cluster.

Other key differences in MongoDB and Neo4j –

- Neo4j and MongoDB are NoSQL databases. MongoDB is a document-type database whereas Neo4j is a graph database.

- MongoDB provides Aggregate Framework functionality and it is not supported in Neo4j.

- Neo4j allows graph traversals like a tree, however, MongoDB does not provide the visual representation of documents as graphs.

- Triggers are not supported in MongoDB whereas Triggers can be maintained in Neo4j.

1. **Apart from the implementation of Graph Analytics, study involving graph-oriented programming models like Giraph and Graphx.**

Apache Giraph -

Apache Giraph is an open source implementation of Google Pregel. Apache Giraph is a graph based framework topped with Apache Hadoop for analyzing social media data. With the input graph, the connections between the nodes are identified. Apache Giraph is chosen for graphs due to its scalability and speed. The advanced search ability of Giraph has been used by social network websites like Facebook where users’ information posted is time lined. It uses Apache Zookeeper for synchronization with computations is executed in memory.

GraphX **-**

GraphX is the Spark API for graph processing and computations. GraphX extends the spark RDD to form a directed multigraph. GraphX supports various graph computations including subgraph, join Vertices, MapReduce Triplets. GraphX also has arrangements for graph algorithms for graph analytics. With the Property Graph combined with Graph Lab developments, the graph implementations can efficiently be executed. Below is the graph-parallel system implemented in Spark’s GraphX API.

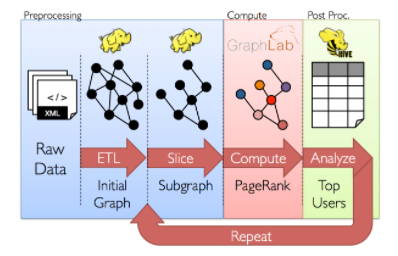
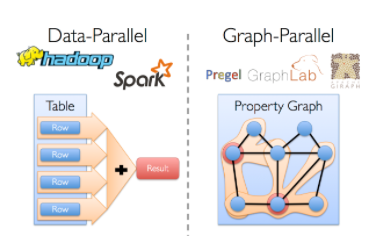


Figure 3: Graph-Parallel systems and GraphX API for graph operations (Ampcamp.berkeley.edu, 2017)

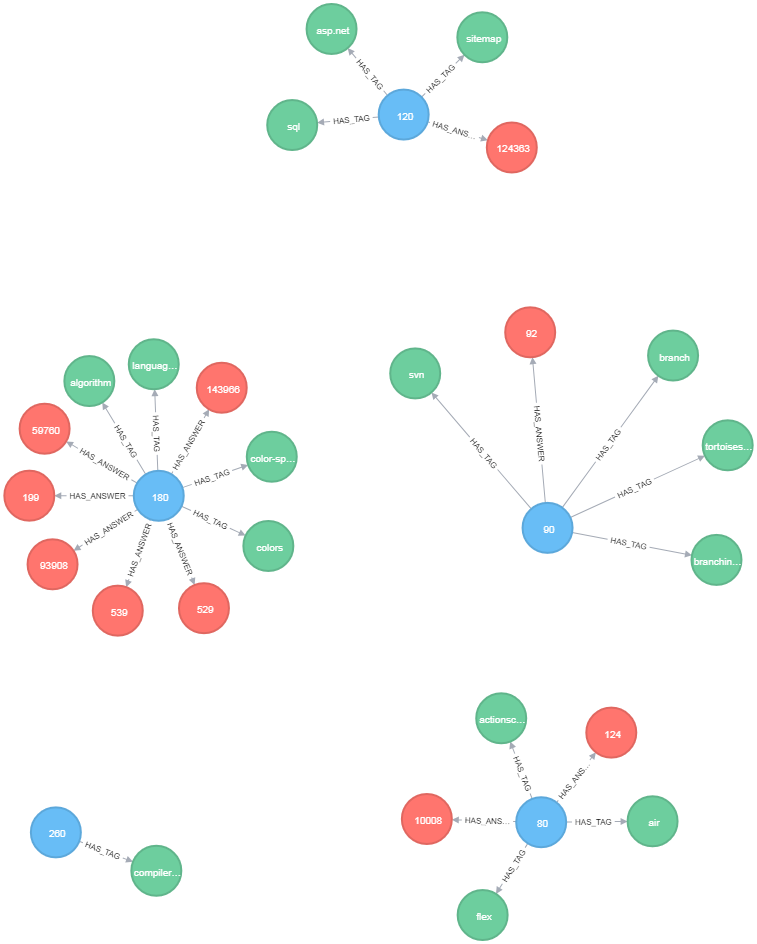
The GraphX provides the programming model that allows graph-parallel systems and data-parallel systems communications and traversing of data between them in an optimized manner. GraphX API is included in Spark packages which can be programmed in Scala programming language.

The Property graphs that are multigraph with nodes having same sources and shared destinations through edges. Graph Operators like modifying graph structures, joining of subgraphs, iterative graph-parallel computations are exposed by GraphX. MapReduce Triplets operator function of Graph API allows to aggregate the neighborhood nodes and get the simplified results. Thus, GraphX enables end-to-end Graph Analytics solutions for big data.

**Implementation:**

The case study of analysis of large graph for Stack Overflow dataset from Kaggle. The Stack Overflow is a Q&A social and information network. The data files were structured, and large graph networks are formed.

The graphs with Questions, Answers, and Tags as the nodes and HAS\_ANSWER & HAS\_TAG as the edges/relations. Below is the constructed graph:



**Tag Node** has attributes tag ID and tag Name.

**Question node** has properties like questBody, questClosedDate, questCreationDate, questScore, questUserID, questTitle, quest ID.

**Answer node** has properties like answerBody, answerScore, answerUserId.

Relationship:   
(t: Tags) <-[:HAS\_TAG] – (q: Questions)-[:HAS\_ANSWER] ->(a: Answer)

The constructed graph can be used for graph analytics for the following use cases:

1) Trends in stack Overflow data for a technology.

2) The Top Answerers for the JavaScript tag

3) All Answers to each Java Questions according to question score.

4) Where Else Were the Top Answerers of Java also Active?

5) People Who Posted the Most Questions about JavaScript

**Comparison of MongoDB & Neo4j:**

For comparison, MongoDB and Neo4j databases where created for the same data set.

Experimental Setup:

• System Specifications for both database: Windows 7 with 16GB RAM, 2.60 GHz Dual-core processor

• Neo4j Community Edition – 3.2.5 version & MongoDB 3.4 version

• Comparisons were performed without indexing and on single node server for both databases.

Performance Based Evaluation:

The aim of this exercise is to show the comparison between MongoDB and Neo4j queries for various listed use cases based on the response time of the queries. The tabular representation shows the time taken to execute the queries and the disk space taken by the databases Neo4j and MongoDB respectively.

|  |  |  |
| --- | --- | --- |
|  | **Neo4j** | **MongoDB** |
| **Database Disk Space** | **0.164 GB** | **0.031GB** |
| Top 5 tags of all time | **65ms** | **31ms** |
| All Answers to each Java Questions according to question score | **34ms** | **15ms** |
| Where Else Were the Top Answerers of Java also Active? | **13ms** | **25ms** |
| People Who Posted the Most Questions about JavaScript | **25ms** | **36ms** |

The MongoDB queries are executed using Aggregate Framework and Neo4j queries are executed using MATCH operator. The test results show that MongoDB and Neo4j are competitive. However, better results can be expected for larger datasets with better visualizations for Neo4j.

**Cloud Implementation:**

Neo4j Graph Database Community Edition is deployed on AWS EC2 instance. This edition deployed is limited to the single machine. The cloud implementation has the same database setup of Stack Overflow dataset. This cloud setup can be used for further applications with larger data set for graph analytics.

**Big Graph Analytics Algorithms:**

The main aim of this implementation is to construct big graph analytics module for Stack Overflow dataset. The end-to-end analytics module based on GraphX, a Spark’s API for graph operations and algorithm is implemented in Scala Programming language.

In this approach, the Spark libraries and GraphX library are imported into Scala Perspective. Using this, this Neo4j and Spark connecter, that application is linked to Neo4j database for fetching data.

After fetching data from Neo4j database, the GraphX’s graph operations and Spark’s RDD are used in the implementation and Page Rank and connected components algorithms are implemented using GraphX API.

Experimental Setup:

* Programming Language: Scala – Scala SDK – 4.7.0
* Dependencies: Spark-core\_2.11, Spark-sql\_2.11, spark-graphx\_2.11
* Neo4j Database – Neo4j 3.2.4 version

**Algorithms:**

Page Rank – For Graph Analytics, the page rank algorithm is implemented using GraphX API to evaluate the experts’ programming language from Stack Overflow dataset.

The graphs are evaluated to fetch user id of the users who asked the questions tagged Java and the user id of the user with maximum scored answer for the corresponding question. As the relationship between Question user ID and answer user id is identified from Neo4j data set, the page rank algorithm is implemented to rank the users and the top experts for Java Programming language are identified.

Results of Page Rank algorithms:

|  |  |  |  |
| --- | --- | --- | --- |
| **User Rank** | **User ID** | **Ranking** | **Language** |
| 1 | 1247 | 9.7702765 | java |
| 2 | 6282 | 9.21478 | java |
| 3 | 3087 | 7.5259477 | java |
| 4 | 3335 | 2.2549554 | java |
| 5 | 6044 | 2.2549554 | java |
| 6 | 12815 | 2.2549554 | java |
| 7 | 14047 | 2.2549554 | java |
| 8 | 304 | 2.2532542 | java |
| 9 | 16414 | 1.7838051 | java |
| 10 | 9636 | 1.6416784 | java |

Connected Components –

The Connected Components algorithm is implemented to evaluate the connected and non-connected tag from the Stack Overflow data set.

The results from GraphX API shows the tags with the same ID assigned are connected tags. Below is the sample result –

|  |  |
| --- | --- |
| Tags | Connected |
| pcm | 3576 |
| yuv | 3576 |
| firewire | 3538 |
| video-camera | 3538 |
| sony | 3538 |
| powerbuilder-build-deploy | 3381 |
| powerbuilder | 3381 |
| dashboard-designer | 3333 |
| performancepoint | 3333 |
| bitcount | 3086 |
| chess | 3086 |
| nntp | 2744 |
| usenet | 2744 |
| use-case | 2630 |
| coroutine | 2630 |

**Conclusion:**

This study of Graph database has allowed studying the graph theory concepts, graph algorithms and various real-world implementations of the graph. Because of this study, the big graph analytics implementations using Apache Spark has enabled handling large data sets and gain insights from Stack Overflow data set.

The performance evaluation of MongoDB and Neo4j dataset implies that Neo4j database has ease of querying and better visualizations as compared to MongoDB. Thus, with the combined experiment of performance evaluation and graph analytics, Graph databases and analytics is an exciting new area to explore its potentials.

As the implementation in this study was taken up on the single server, the future scope can include setting up distributed instances of the database server. The clustering setup for Neo4j database can be implemented using Neo4j Enterprise Edition only. With the perspective of graph analytics, advancements in this can be obtained with better visualizations and develop the graph analytics application for the distributed cluster.

Acknowledgement: Supported and guided by Prof. Holden for the study and provision of the machine. Sincere gratitude for the aid to the Professor and IST Department.

1. **References:**

Ammar, A. (2016). Query Optimization Techniques in Graph Databases. *International Journal of Database Management Systems*, 8(4), pp.01-14.

Ampcamp.berkeley.edu. (2017). *Graph Analytics With GraphX*. [online] Available at: http://ampcamp.berkeley.edu/big-data-mini-course/graph-analytics-with-graphx.html [Accessed 15 Dec. 2017].

Disney, A. (2017). *Graph Data Modeling 101*. [online] Cambridge Intelligence. Available at: https://cambridge-intelligence.com/graph-data-modeling-101/ [Accessed 15 Dec. 2017].

Docs.mongodb.com. (2017). *Sharding — MongoDB Manual 3.6*. [online] Available at: https://docs.mongodb.com/manual/sharding/ [Accessed 15 Dec. 2017].

Fan, W., Wang, X. and Wu, Y. (2017). *Querying big graphs within bounded resources*.

Graphdatamodeling.com. (2017). *Graph Data Modeling*. [online] Available at: http://www.graphdatamodeling.com/ [Accessed 15 Dec. 2017].

Jimwebber.org. (2017). *On Sharding Graph Databases – World Wide Webber*. [online] Available at: http://jimwebber.org/2011/02/on-sharding-graph-databases/ [Accessed 15 Dec. 2017].

Tsham Mpinda, S., Ferreira, L., Ribeiro, M. and Prado Santos, M. (2015). Evaluation of Graph Databases Performance through Indexing Techniques. *International Journal of Artificial Intelligence & Applications*, 6(5), pp.87-98.