3rd International Conference and Workshops on Recent Advances and Innovations in Engineering, 22-25 November 2018
(IEEE Conference Record # 43534)

Performance Analysis of RDBMS and No SQL Databases: PostgreSQL, MongoDB and Neo4j

Monika Sharma
Department of Computer Engineering
Poornima University, Jaipur, India
smonika15@gmail.com

Vishal Deep Sharma
Software Department
ARRK Solutions Pvt. Ltd. Navi Mumbai
vishaldeepsharma@gmail.com

Mahesh M. Bundele
Department of Computer Engineering
Poornima University, Jaipur, India
mahesh.bundele@poornima.edu.in

Abstract— **In this smart phone era, geospatial data plays the vital role in developing various citizen centric services for sustainable development of the society such as developing the smart cities, disaster management services and locating basic amenities such as schools, hospitals, Railway Stations, banks etc. People are generating geo tagged data on various social media websites like face book, twitter etc. which can be considered as big data due to acquiring major three attribute volume, variety and velocity of big data. Multiple sources are generating voluminous, heterogenous data which can't be fit into a predefined structure. Also, the applications based on these data require higher through put time. Managing Such a huge amount of data is really challenging. This geotagged data needs to be handled using big data management techniques like No SQL rather than traditional Relational Data Base Management System. Selection of appropriate type of No SQL Database is an important activity before developing any Geographical Information System based application. In this paper we tried to find out suitable No SQL data base for GIS application. This paper compares the performance of MongoDB and Neo4j in case of search queries executed on geotagged data.**

Keywords—Relational Data Base Management System, Geotagged, Geographical Information System, NoSQL, Graph Data Base, Document Based Data Base, MongoDB, Neo4j;

I. Introduction

Geo-tagging, as the name suggests stands for geographical tagging of location or a place. Geotagging is an activity of adding metadata to any form of information, be it a text message, an image or a video. It helps us to visualize the message and understand it in a better way. Latitude and longitude are the two main driving forces of geotagging. Other geo-coordinates may include bearing, place names, altitudes distance or even a timestamp. This data is mainly generated via crowdsourcing hence there is no control on the data. Such data is heterogenous and large to manage using traditional Relational Data Base Management System. Geotagged data plays significant role in developing various citizen centric applications like urban planning, public health, smart city applications, disaster management etc. Hence it is important to manage this data efficiently to harness its actual power. Main contribution of this paper is to throw the light on choosing appropriate No SQL data base specifically between the document-based and Graph based Data structure in context of GIS based applications.

Paper is organized in five sections. Section I is about the introduction, similar work done in this field is given in section 2. Brief introduction of No SQL data base was given in section 3. Experimental setup is explained in section 4 and conclusion is presented in section 5.

II. Related Work

Authors compared the performance of MongoDB and Post GIS on indexed and non-indexed data set for spatial data and found that MongoDB performs better for large amount of data however there is no rich set of spatial query functions available in MongoDB as compared to Post GIS [1]. Author proposed a distributed architecture based on three No SQL databases (Apache Cassandra, Apache HBase, MongoDB) and found Cassandra as the most suitable database model for big remote sensing data management approach [2]. Author proposed a data adapter system to convert RDBMS to No SQL DB but not tested on big data [3]. A system to map RDBMS to MongoDB specifically for BLOB data was proposed [4]. Proposed hybrid approach where The RDBMS was used in the GIS Desktop while the NoSQL database was used for data storage and transfer on the basis of JSON document [5]. Hybrid approach gives better performance however it has not been tested for visualization purposes. Comparison between MySQL and MongoDB was done specifically for IOT (internet of things) domain and found in some scenarios where MongoDB performed better but MySQL gives stable response time [6]. After exploring related work in this field, we found that most of the work done using MongoDB as a No SQL Data base and people have not explored graph data bases in their full potential. It seems that for GIS portal, relationship between heterogenous data is important to get full benefit of huge amount of existing data. Graph data bases are more suitable for establishing relationship in fast, easy and accurate manner. Therefore, here in this study, we try to analyse the comparison between RDBMS, Document based No SQL Data base and Graph based Data base for GIS scenario. We have taken MongoDB as document-based data base and Neo4j as a Graph based data base for experiment purpose.

III. No SQL DataBase

A. Introduction

No SQL databases emerged to handle the voluminous data along with the capabilities like scalability, join free queries and no fixed schema to provide more flexibility. According to data models, No SQL data bases are categorized as "key-value store", "column-oriented store", "document-oriented store" and "graph databases" [10].

- Key value store: It is the simplest data model where data is stored as a key value pair.
- Column-oriented store: Data is stored in columnar manner rather than rows.
- Document store Database: It provides the efficient way to manage document-oriented information in semi structured data format. It has a layer to maintain the relationship between these documents.
- Graph store Database: This type of data stores keep data in graph structure to represent relationship between data. It stores the data in terms of nodes, edges and properties.

In this paper we have taken Neo4j as graph database and MongoDB as document-based database for performance check in case of simple queries. Table I shows the comparison between system properties of PostGre SQL, Neo4j and MongoDB. There are certain features available in MongoDB but not in Neo4j like sharding whereas triggers are not available in MongoDB, but it is possible in Neo4j. Hence choosing the appropriate No SQL data base should be based on typical scenarios and applications in context.

B. *Neo4j*

Neo4j is built from the ground up to be a graph database. The architecture is designed for optimizing fast management, storage, and traversal of nodes and relationships. It has unique features like highly scalable, suitable for rapid development, good for business critical and high-performance operations [7]. Table II shows the common concepts of RDBMS and their equivalent in Neo4j. In neo4j, RDBMS Tables are considered in terms of graph and each record of the table is taken as a node of the graph [8]. Columns and data are considered as the attributes and its values of nodes in Neo4j. Neo4j has Cypher query language apart from other access methods such as Java Application Programming Interface and RESTful API.

C. *MongoDB*

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling [9]. Being a document-based NoSQL data base, it reduced the need of joins and provides the facility to create flexible and dynamic schema. It claims for high performance, high availability and horizontal scalability.

TABLE I. COMPARISON OF SYSTEM PROPERTIES

Property	**Comparison among PostGre SQL, MongoDB and Neo4j based on system properties**		
	PostgreSQL	***Mongo DB***	***Neo4j***
Data base model	Relational DBMS	Document Store	Graph DBMS
License	Open Source	Open Source	Open Source
Triggers	Yes	No	Yes
Foreign Key	Yes	No	Yes
Typing	Yes	Yes	Yes
Secondary indexes	Yes	Yes	yes

TABLE II. NEO4J VS. RDBMS

Sr. No	**Structural mapping in Neo4j w.r.t RDBMS**	
	RDBMS	***NEO4J***
1	Tables	Graphs
2	Rows	Nodes
3	Columns	Properties and its values
4	Join	Traversal

TABLE III. MONGODB VS. RDBMS

Sr. No	**Structural mapping in MongoDB w.r.t RDBMS**	
	RDBMS	***MongoDB***
1	Tables	Collection
2	Rows	Document
3	Columns	Field
4	Join	Embedded document

Table III shows the common concepts of RDBMS and their equivalent in MongoDB. In MongoDB, collections are like tables of RDBMS, documents are similar to rows and fields are similar to columns of RDBMS. It has its own protocol using JSON to access the data however SQL read only queries are possible via MongoDB connector.

IV. EXPERIMENTAL RESULTS

A. *Experimental Data Setup*

For experimentation, OpenStreetMap data was taken in "osm" format and migrated into PostGre SQL data base using a java program. Mumbai region was selected which gave more than 2 GB data. Around 600 users had tagged various Point of Interest data for Mumbai and Navi Mumbai region. Geographical data is described in terms of nodes, ways, relations and tags. Tags are attributes associated with any node, way or relation. Nodes are smallest object which can be considered as point where as ways are like edges or connection between nodes. Relations are the collection of points (nodes) and ways which represent the larger set of ways or nodes. For better understanding of a node, ways and relations, Fig 1 can be referred. A point tagged as "Apollo hospital Navi Mumbai" represents a node with attribute value name as Apollo hospital Navi Mumbai, a highway NH348A can be considered as way which connects multiple point on highway. Similarly, relation can be a collection of many highways in this region. During the data scanning, it was observed that some nodes were not related to any way or relation hence we discarded such nodes. Specifically, we found 580 users who have tagged the nodes having connection with ways and relation. Finally, 5,20,689 nodes were selected which had some association either with ways or relationship which can be considered as point where as ways are like edges or connection between nodes. Relations are the collection of points (nodes) and ways which represent the larger set of ways or nodes. To better understand the node, ways and relations, fig 1 can be referred. A point tagged as "Apollo hospital Navi Mumbai" represents a node with attribute value name as Apollo hospital Navi Mumbai, a highway NH348A can be considered as way which connects multiple point on highway. Similarly, relation can be a collection of many highways in this region. During the data scanning, it was observed that some nodes were not related to any way or relation hence we discarded such nodes. Specifically, we found 580 users who have tagged the nodes having connection with ways and relation. Finally, 5,20,689 nodes were selected which had some association either with ways or relationship.

Data setup was done for all three data bases in following ways:

- For PostgreSQL total nine tables node, node_tag, relation, relation_tag, relation_way_mapping, user_details, way, node_way_relation, way_tag was created as shown in fig 5 and OSM data was migrated into appropriate tables for all 5,20,689 nodes.

- In Neo4j only those tables were created first which have only primary key, no foreign key was involved and directly these were imported through csv from PostGre SQL DB using below command

  COPY (SELECT * FROM user_details) TO '/tmp/user_detail.csv' WITH CSV header;

  Once the csv is available it was loaded directly in Neo4j using below mentioned commands.

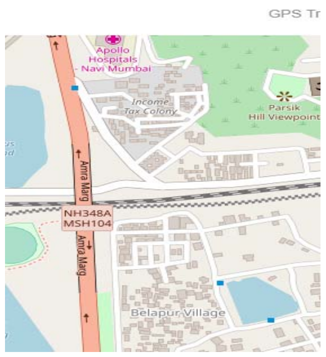
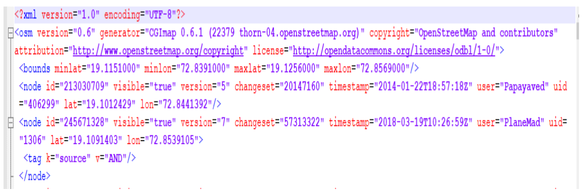
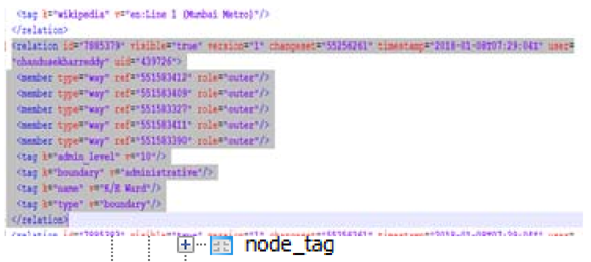


Fig. 1. Example of a node, way and relation

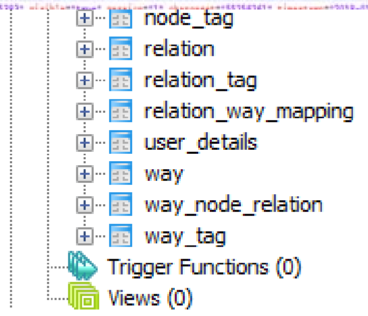


```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.6.1 (22379 thorn-04.openstreetmap.org)" copyright="OpenStreetMap and contributors"
attribution="http://www.openstreetmap.org/copyright" license="http://opendatacommons.org/licenses/odbl/1-0/">
 <bounds minlat="19.1151000" minlon="72.8391000" maxlat="19.1256000" maxlon="72.8569000"/>
 <node id="213030709" visible="true" version="5" changeset="20147160" timestamp="2014-01-22T18:57:18Z" user="Papayaved" uid
 ="406299" lat="19.1012429" lon="72.8441392"/>
 <node id="245671328" visible="true" version="7" changeset="57313322" timestamp="2018-03-19T10:26:59Z" user="PlaneMad" uid=
 "1306" lat="19.1091403" lon="72.8539105">
  <tag k="source" v="AND"/>
 </node>
```

Fig. 2. Example of a xml structure of a node in osm file

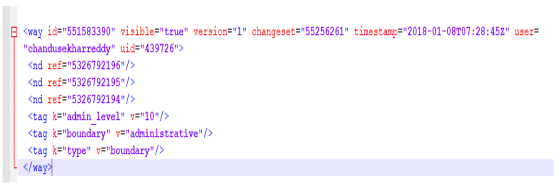


```
  <tag k="wikipedia" v="en:Line 1 (Mumbai Metro)"/>
 </relation>
 <relation id="7885379" visible="true" version="1" changeset="55256261" timestamp="2018-01-08T07:29:04Z" user=
 "chandusekharreddy" uid="439726">
  <member type="way" ref="551583412" role="outer"/>
  <member type="way" ref="551583409" role="outer"/>
  <member type="way" ref="551583327" role="outer"/>
  <member type="way" ref="551583411" role="outer"/>
  <member type="way" ref="551583390" role="outer"/>
  <tag k="admin_level" v="10"/>
  <tag k="boundary" v="administrative"/>
  <tag k="name" v="K/E Ward"/>
  <tag k="type" v="boundary"/>
 </relation>
```



```
node_tag
relation
relation_tag
relation_way_mapping
user_details
way
way_node_relation
way_tag
Trigger Functions (0)
Views (0)
```

Fig. 4. Example of a xml structure of a Relation in osm file



```
<way id="551583390" visible="true" version="1" changeset="55256261" timestamp="2018-01-08T07:28:45Z" user=
"chandusekharreddy" uid="439726">
 <nd ref="5326792196"/>
 <nd ref="5326792195"/>
 <nd ref="5326792194"/>
 <tag k="admin_level" v="10"/>
 <tag k="boundary" v="administrative"/>
 <tag k="type" v="boundary"/>
</way>
```

Fig. 5. osm data converted in to tables in PostgreSQL

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:user_detail.csv" AS row

CREATE (:User_Detail {id: row.Id, username: row.UserName});

Similarly, data of node table was also migrated in Neo4j using below commands.

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:node.csv" AS row

CREATE (:node {id: row.Id, user_id: row.user_id,latitude: toFloat(row.latitude), longitude: toFloat(row.longitude) });

Once all the tables along with data were migrated, then relationship was created on User_details table and Node table so that all the nodes created by a specific user can be identified.

MATCH (a:user_details), (b: node)

WHERE a.id=b.user_id

CREATE (a)-[:CREATED]->(node)

RETURN a,b

Similarly, all the other relationships were created and data is ready to use in Neo4j.

- For Mongo DB, creating a table in not required but the data can be inserted directly through below command.

  mongoimport -d TestMumbai -c user_details – -type CSV – -file user.csv –headerline

  mongoimport -d TestMumbai -c node – -type CSV – -file node.csv –headerline

  Example of one record in node document is given below:

  {

  “user_id”:”45708”

  “version_no”:3

  “changeset”:28799

  “latitude”:”19.2561”

  “longitude”:”72.854”

  }

In above example, value of Latitude and longitude would be converted as numeric values using function NumberDecimal for calculation.

B. *Environment setup*

PostGre SQL, MongoDB and Neo4j were installed on 64-bit windows environment on Intel® core ™ i7-7700 HQ CPU @ 2.80 GHz and 16 GB RAM machine.

C. *Queries*

Following two simple queries were executed to analyses the performance of all three data bases.

S1- Find all users who have tagged any point
S2- Find all nodes created by given user

D. *Results*

Query S1 and S2 were executed using SQL interface provided by PostGre SQL, MongoDB and Neo4j. In case of query S1, execution time in milliseconds was noted for different no of users and Fig 6. Shows the performance of all three data bases in each case. It shows that MongoDB performs quite well whereas Neo4j has least performance. Query S2 has a filter criterion where all the nodes created by a specific user were fetched and time taken to get the records was measured. Fig 7. shows the performance of three data bases in case of query S2. Once again, the performance of Mongo DB was best in terms of less time taken to execute the query irrespective of the no of records but neo4j was least performer.

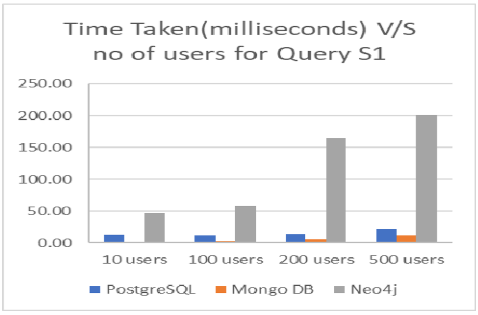


Fig.6. Graph between no of users and time taken to execute the query S1 in different data bases.

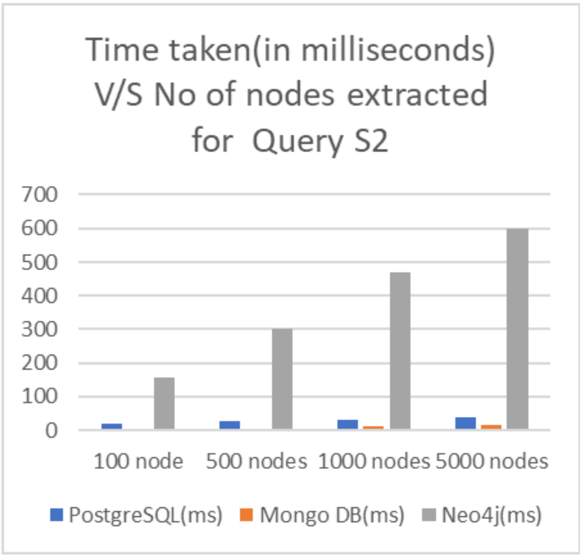


Fig. 7. Graph between no of nodes and time taken to execute the query S2 in different data bases.

V. CONCLUSION AND FUTURE RESEARCH SCOPE

In this research paper, we studied the performance of RDBMS, Document based No SQL data base (MongoDB) and Graph based No SQL Data base (Neo4j). We got unexpected results in case of neo4j as it took longest time as compared to MongoDB and PostGre SQL. Further investigation was done and found that it happened due to cypher Query platform. In real scenario, end user will access the Graph DB through some application and then response time will change in case of Accessing Neo4j using Java Application. In current scenario where we have directly used the available query access platform, these results recommend that MongoDB Performed well specifically in case of large number of records. This experiment needs to be further explored in case of insert and update operations and through a common Java Application to analyze the performance more accurately.

REFERENCES

[1] S. Agarwal and K. Rajan,(2017) “Analyzing the performance of NoSQL vs. SQL databases for Spatial and Aggregate queries ” Free Open Source Softw. Geospatial, 17(1). 6-14.

[2] Y. Hajjaji,(2018) “Performance investigation of selected NoSQL databases for massive remote sensing image data storage,” 4th

International conference on advanced technologies for signal and Image Processing-ATSIP,2018.

[3] S. Ghule and R. Vadali(2017), “Transformation of SQL system to NoSQL system and performing data analytics using SVM,” 2017 Int. Conf. Trends Electron. Informatics, pp. 883–887.

[4] A. E. Ibrahim, M. Youssef, and E. El Fakharany,(2017) “Transforming RDB with BLOB fields to MongoDB,” 8th Int. Conf. Inf. Commun. Syst. ICICS 2017, pp. 296–301.

[5] W. Tampubolon,(2016) “Hybrid Concept of NoSQL and Relational Database for GIS Operation,” 19th Agil. Int. Conf. Geogr. Inf. Sci. Helsinki, Finl., p. 2.

[6] S. Rautmare and D. M. Bhalerao,(2016) “MySQL and NoSQL database comparison for IoT application,” 2016 IEEE Int. Conf. Adv. Comput. Appl. ICACA 2016, pp. 235–238.

[7] https://neo4j.com/docs/developer-manual/current/introduction

[8] https://www.tutorialspoint.com/neo4j/neo4j_overview.htm

[9] https://docs.mongodb.com/manual/introduction/

[10] https://en.wikipedia.org/wiki/NoSQL