Review Of Nosql Databases And Performance Comparison Between Multimodel And Polyglot Persistence Approach

Neha Singh Rohit Chandra Sunil B. Shambharkar J. J. Kulkarni

Abstract: With advances in recent proliferation of pervasive computing, Social Networking, E-Commerce data are growing with exponential pace. This is also true in scientific experiments such as shown by the recent discovery of Higgs Boson. The data is known as BigData attributed to its sheer volume, diverse variety and lightning velocity, thereby mandating the need for efficient systems for storage and analysis. In this paper, we explore various kinds of NoSQL databases popularly used for storage and processing of BigData and also the limitation of each one of them. We developed an experimental model to compare polyglot persistence approach performance with Multi-model databases for unstructured datasets. We also discuss different scenarios in which these out performs each other.

Keywords: Polyglot Persistence, Multi-model, Big Data, Unstructured Data, Column family, Document database, Graph database, No SQL.

1 INTRODUCTION

Relational databases are the first preference for anyone when we talk about databases. Consistency and Functionality are the prime features of Relational Databases. However, in the recent scenarios where database requirements are constantly changing, Relational Databases have limitations like handling unstructured and massive size of data. To meet up all these requirements NoSQL databases come into picture [1]. Data division and duplication are the basic features of NoSQL databases [16]. NoSQL databases provide more support to semi-structured or unstructured data, including the capability of changing database schema if needed in compare to RDBMs. [11]. As each module having different requirements, it is not feasible to apply one NoSQL for all the modules of complex application. For some module consistency may be major concern and for others it may be availability. There is a need of multiple NoSQL database models. Every NoSQL has its pros and cons. Scalability and processing unstructured data problem can be solved using Polyglot Persistence (coined by Neal Ford, 2006) approach, where different modules can have their own different data processing mechanism [17]. Multi-model systems integrate several data models like polyglot persistence requirement, which is a new class of data store systems. These systems, simplifies the process of application development [2]. In this paper, we have compared different categories of NoSQL databases. We have developed experimental model to study and compare the performance of Multi-model databases and polyglot persistence for unstructured datasets. Our major contributions involve bringing issues of different type of NoSQL database models.

- *Neha Singh Computer Division Bhabha Atomic Research Center, Trombay, Mumbai, India neha@barc.gov.in*
- *Rohit Chandra Computer Division Bhabha Atomic Research Center, Trombay, Mumbai, India crohit@barc.gov.in*
- *Sunil B. Shambharkar Computer Division Bhabha Atomic Research Center, Trombay, Mumbai, India sunils@barc.gov.in*
- *J. J. Kulkarni Computer Division Bhabha Atomic Research Center, Trombay, Mumbai, India jjk@barc.gov.in*

The rest of the paper is organized as follows: Section II gives brief overview of NoSQL database models. Section III gives brief introduction of Polyglot Persistence. Section IV explains Multimodal database. Section V contains experimental setup, Details of Data Set and Server Configuration. Section VI contains Performance Evaluation. Section VII concludes the paper.

2 LITERATURE SURVEY

A. Introduction to Key Value Data Stores

Key value data stores based on hash table implementation where key-value pairs distributed across different remote servers in distributed cluster storage. Keys are mapped to a particular value or set of values. Keys should be indestructible means it should be unique and atomic in nature. Queries are based upon key value to fetch any record from key-value database storage. By this way data can be stored efficiently in schema-less format and can support the extremely fast random read-write access. Foreign key references, complex SQL Joins and Group By can be avoided by using key-value data store since different key value pairs store a set of unrelated data [3].

It is best suited to following conditions:

- Streaming data since it is mostly in memory database.
- Analysis of any web application can be done within a second, to measure the load and popularity [3].
- Content distribution and management, caching and cookie storage [3].
- Unauthenticated clients can also access the TCP/ Unix socket, Key value data store best suited for Highly secure transactional operations on dataset [3].

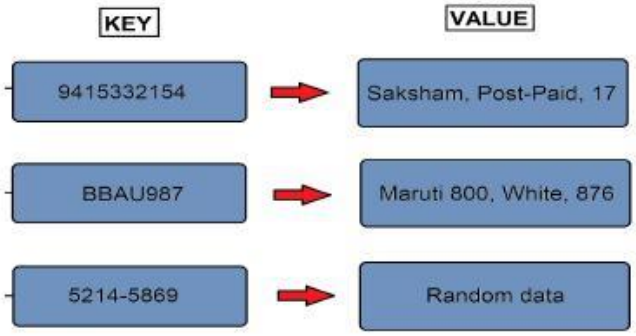

Figure 1: *Key Value Data Store*

B. Introduction to Document Store Database

The Document database is a NoSQL database which store and query data as XML and JSON-like documents, in compare to Relational Database, which store data in rows. Rows can be compared with a document and table with group of documents which is called Collection [4]. In a collection each document can have different schemas and differ on the number and type of data being stored. It is especially optimized to store textual information. Since related data set is stored together it saves the overheads of SQL JOIN operation [3]. Although the database has a schema free design, the stored records are semi-structured and exist in the form of hierarchies. They provide for embedded documents, which are self-describing, hierarchical tree structures that often comprise of maps, collections, and scalar values [18].

It is best suited to following conditions:

- Very well suited for web based application which has semi structured data.
- It is not suited for application requiring excessive joins as joins are done through lookup.
- It is not suited for applications requiring high conformity to ACID properties like financial operations [3].

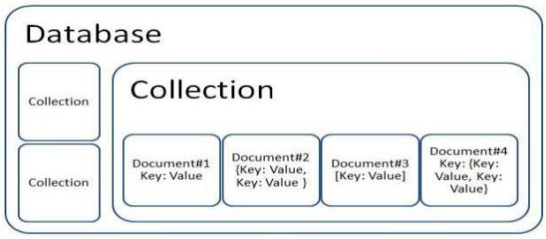

Figure 2: *Document Store Database*

C. Introduction to Column Family Database

Column Family Stores are also known as column-oriented stores, extensible record stores and wide columnar stores [4]. Entire column of a table is stored and mapped to a single key. We can search only a part of table because all the entries in columns have indexes. Super columns can also exist as hierarchies of nested columns inside it [3]. Through this super key records can be easily look-ups and can be easily accessible with avoiding unnecessary overheads to look for individual key of a record. These databases are very well suited for aggregation queries through Map Reduce features.

It is best suited to following conditions:

- Good for automatic load balancing and fault tolerance [3]
- Suitable for performing range-based queries [3]
- Not suitable for application which requires scanning of large dataset row-wise scanning containing all fields because of its column-oriented nature.

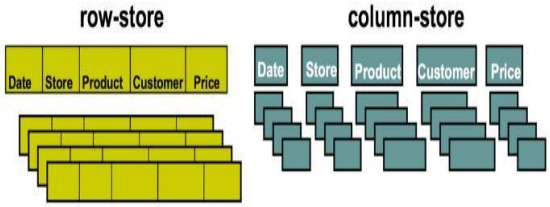

Figure 3: *Column Family Database*

D. Introduction to Graph Database

Graph oriented databases are represented in form of graphs in which data represented as nodes and edges of graphs. Edges connected to each other through relations in a tree like structure [9]. If Graph is directed, nodes stored in ordered form. The common theme among Graph Databases is that data is structured in a mathematical graph. A graph G = (V, E) made of a set of vertices V and a set of edges E. An edge e∈ E is a pair of vertices (v1, v2) ∈ V X V. It is best applicable for traversing and searching applications, such as finding related links on LinkedIn, looking up friends on Facebook [9]. Relationships between data items are preferred over data itself. Graph algorithms are used efficiently for fast traversal and optimization of performance.

It is best suited to following condition:

- It good for concurrent operations and fast real-time analysis of data.

3 POLYGLOT PERSISTENCE

A. Introduction to Polyglot Persistence

A polyglot persistence means picking the right tool for the right use case. In this approach problems are divided into segments and different database models are applied to solve a complex problem. As in above section we have seen each NoSQL data model have their limitations. It is very difficult to manage Big Data using a single NoSQL database model as each one have their own limitations. For example Document Databases are very good for web application having semi structured data like Content Management [12] but it does not perform well ACID Compliant transactions like banking. For cases such as Transaction Logging Column Family performs very well. But it failed in applications like having mathematical calculations like Sum, Average [13]. If we want to store session information of web application or want to manage shopping cart details in E-Commerce site, Key Value can work well. Key value No SQL is not advisable to use for databases where lot of relationships exists or multi key transactions takes place [15]. Graph stores working fine for processing of customers social graphs on social network [14]. Hence each No SQL database is works well in certain cases and not performs well for others cases. Any E-Commerce application has all kinds of use cases as shown in figure 4. For example application has to store session information of logged in persons. Even when

someone selects an item and added it's to his/her shopping cart, application needs to manage this cart information. Key value works well in both cases. Any enterprise application is a combination of many small use cases. We will discuss this issue by taking a well-known business model of E Commerce. In an E Commerce website user browse through availed catalog, add an item in cart, places order and finally makes payment as shown in figure 4. After placing order user will get a confirmation email or text message. Then order will be dispatched from warehouse through courier service at a given address. Various banks, courier services and shops are involved in this complex workflow. Application needs to manage Product catalogue with all the changes needs to be reflected immediately in the catalogue if any new item added [17]. This involves lots of read and infrequent write. Each product details can be stored very well in Document stores where one document matches to a product id. As shown in figure 4. While payment module should be ACID compliant, NoSQL cannot be uses in this case. Relational database supports ACID properties works best for accounting and financial transactions. Every buyer is also a user of social networking site these days and can recommend product his friend or family. Rapid traversed links between friends, products, purchases and ratings is needed for it. Graph model work very well in this case. For analysis of buying pattern, logging and auditing application maintains activity logs. Since this analysis is also bases on session id we can use Key-value store in this case [17]. So from above analysis we are in conclusion that expecting single No SQL for entire business model will not perform effectively. So we have to consider Polyglot Persistence database approaches in an E-Commerce type application. Polyglot Persistence allows multiple database models to be used within an application where one database model process one part of application and use same data which is used by different database model to process another part of same applications [17].

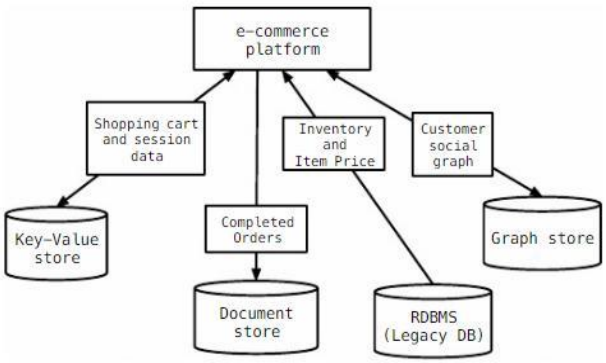

Figure 4: *Polyglot Persistence*

B. Limitations of Polyglot Persistence

- Resulting in complex application development as each database has its own API's and each database has to be learnt separately
- Difficult to manage all databases simultaneously as we need to replicate and take backups of each one of them.
- Consistency of data across different database needs to be handled by application.

4 MULTI-MODEL DATABASE

A. Introduction to Multi-model Database

A multi-model database system combines more than one data model into single database engine thus providing unifying platform for data storage and processing. For example, ArangoDB multi-modal database which integrates document, graph, and key-value models. Transactions, partitioning and replication are key features of it. It works on AQL, which allows joins, operations on graphs, iterations, filters, projections, ordering, grouping, aggregate functions, union, and intersection. It also supports all the ACID properties [19]. Thus it helps to reduce operational friction resulting from using polyglot persistence of learning different query language and solve data consistency and duplication issue of polyglot persistence.

5 EXPERIMENTAL SETUP

A. Dataset Overview

The patent data consists of detailed information on almost 3 million U.S. patents granted between January 1963 and December 1999 along with citations made to these between 1975 and 1999 which extends over 16 million. Patents data have long been recognized and used for research. The dataset though can be modelled through relational databases but analysis on the dataset would not be as efficient as compared using NoSQL database because of the graphical nature of citation data. Citations data is highly interlinked and nested to several depth levels (as high as 17 levels) in graph making queries complex and inefficient in relational database. So graph database is best suited to perform analysis over citation collection using graph properties and algorithms. Patents and Inventor data is well suited for document databases as details (schema) of Inventors at times can be dynamic and there can be several inventors associated to a patent. Modelling the data through document database allows for flexibility of schema and efficient search over dataset.

Characteristics

- Each patent data contains detailed information about inventor, innovation and technological area, assignee.
- The patent data consists of over 6 million patents and increasing at rate of 1,50,000 patents per year as observed between 1999-2000. Thus the data to be analyzed is huge and well suited for current research.

B. Dataset in Details

PATENT

1. Patent Number
2. Grant year
3. Grant date
4. Application year
5. Country of first inventor
6. State of first inventor
7. Assignee identifier
8. Assignee type (i.e., individual, corporate, or government; foreign or domestic)
9. Main U.S. patent class
10. Number of claims (starting in 1975)
11. Technological category

12. Technological sub-category
13. Number of citations made
14. Number of citations received
15. Percent of citations made by this patent to granted since 1963
16. Measure of "generality"
17. Measure of "originality"
18. Mean forward citation lag
19. Mean backwards citations lag

Percentage of self-citations made –upper and lower bounds

CITATION TABLE

It includes following fields

1. Citing patent number
2. Cited patent number

INVENTOR TABLE

1. Inventor Name
2. Inventor Address

There can be multiple inventors for a patient.

ASSIGNEE TABLE

It includes following fields

1. Assigner Identifier
2. Assignee Name

C. Cardinality of Dataset in Details

The performance of both approaches has to be evaluated against parameter write, read and storage. For measuring the performance, three sizes of Patent dataset has been taken. Small dataset consist of 10000 records of document and nodes and corresponding 17,235 edges. Medium dataset consist of 100000 records of document and nodes and corresponding 2,07,547 edges. Large dataset consist of 29,23,922 records of document and 1,65,22,438 edges. The experiments were performed on the server setup mentioned in 5.4 and dataset is tabulated as below:

Table I
Cardinality of dataset

Dataset	*No of Records*	*No of Edges*
Small	*10,000*	*17,235*
Medium	*10,0000*	*2,07,547*
Large	*29,23,922*	*1,65,22,438*

D. Server Configuration

Server with Intel Xeon Core i3 was setup for evaluating the performances.

Table II
Hardware Configuration

Processor	Intel(R) Core(TM) i3-41500 CPU @3.50GHz
Processor Architecture:	x86-64
RAM	8 GB
OS	Windows 8.1 Pro 64 –bit

6 PERFORMANCE EVALUATION

The performance of multi-model database ArangoDB is evaluated against Polyglot approach (Neo4j for graph + MongoDB for document database) observed for insertion and read operations. Neo4j is a high performance graph database which provides object oriented, flexible network structure. It is based on a Property graph data model which comprises of nodes and relationship along with their properties [10]. MongoDB is a document store database which stores semi-structured data written in XML, JSON or JSON like language into BSON (Binary JSON) format [5]. ArangoDB is a multi-model database that integrates document, graph, and key-value models [19]. It supports transactions, sharding and replication, and the Foxx language to develop components on the server side.

Performance equation:

Performance equation for polyglot approach (With Indexing done Patent_Number field)

Performance Equation

$$T_{write} = N_n * (T_{WM} + T_{WN}) + T_{IN} + N_r * T_{RN}$$
$$T_{read} = N_n * (T_{NR} + T_{MR}) + T_{join} (N \bowtie M)$$

Table III
Terms Details in Performance Equation

T_{write}	Total time taken for insertion operation
T_{read}	Total time taken for read operation
N_n	Number of Nodes or records
N_r	Number of edges
T_{WM}	Time for insertion of each document in MongoDB
T_{WN}	Time for insertion of each node in Neo4j
T_{IN}	Time for creation of Secondary Index in Neo4j for faster search
T_{RN}	Time for creation of each relationship in Neo4j
T_{NR}	Time to Read each Neo4j node
T_{MR}	Time to Read each MongoDB document
T_{join}	Time for join operation between Neo4j and MongoDB N = Neo4j
M	MongoDB
N	Neo4j

Table I gives the details of dataset selected for both Polyglot and Multi-model approach to calculate write, read and storage performance. The timings thus noted are presented in tables as followed:

A. Write Performance

Table IV
Write Performance of Multi-model and Polyglot

	Muli-model (ArangoDB)	Polyglot (Neo4j+ MongoDB)
	Timing in Seconds	
Small Dataset	4.933	9.847
Medium Dataset	38.78	43.034
Large Dataset	2287.02	1185.22

B Read Performance

Table V

Read Performance of Multi-model and Polyglot

	Mulimodel (ArangoDB)	Polyglot (Neo4j+ MongoDB)
	Timing in Seconds	
Query with depth = 3 No. of records = 1593	0.6	3.023
Query with depth = 5 No. of records = 23129	9.95	9.8
Query with depth = 7 No. of records = 23129	224.7	25.9
Query with depth = 10 No. of records = 157908	3120.36	250.665

7 RESULTS AND CONCLUSIONS

A. Results

Table I gives the details of dataset selected for both Polyglot and Multi-model approach to calculate write, read and storage performance and Table IV, V and VI gives the details of The timings thus noted are plotted in the graphs as followed:

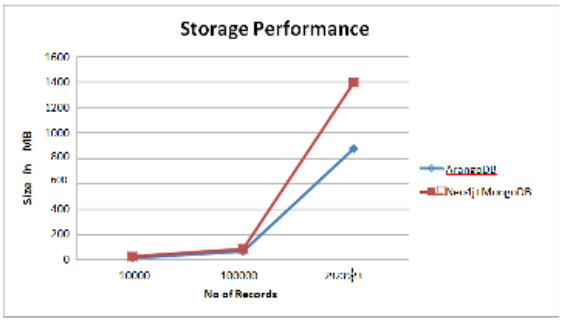

Figure 5: *Insertion Performance*

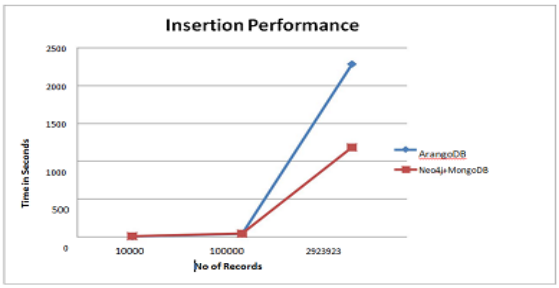

Figure 6: *Storage Performance*

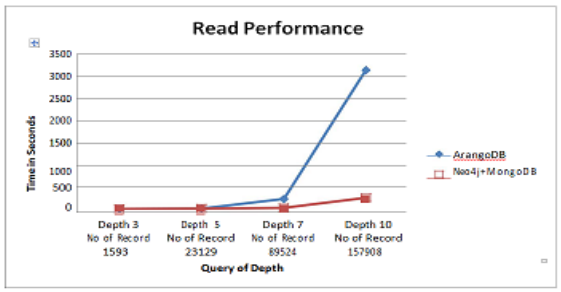

Figure 7: *Read Performance*

B. Conclusions

It was worth noting that for small and medium insertion load ArangoDB performs slightly better than Polyglot approach (MongoDB+Neo4j), since some of the overhead is involved for join operation which is carried out in application layer in case of polyglot approach. For large dataset Polyglot approach performs comparatively better than ArangoDB. The storage requirements for Polyglot approach (Neo4j+MongoDB) is considerably higher than ArangoDB as Neo4j stores lot of meta data to preserve the graph structure in the disk in form of adjacency relation. The adjacency relation is contained in the node itself. ArangoDB stores the graph structure in form of another collection which includes all the edges information of the graph. Read performance involving depth query using graph traversal below 5 level shows that ArangoDB is much faster than Polyglot approach (Neo4j+MongoDB) but as the depth level of query increases Polyglot approach (Neo4j+MongoDB) performs better than Multimodel Database ArangoDB. It is evident from the fact that ArangoDB use a separate collection to store the edges of the graph and the graph traversal is thereby done by using the index lookup which makes it slow for greater depth queries whereas Neo4j uses the adjacency information stored in the node to perform the graph traversal resulting better read performance graph queries. ArangoDB is better choice for lesser depth queries since it gets the record directly using index lookup eliminating overhead incurred in joining two different databases under polyglot approach.

C. Future Work

In future we will evaluate some more Multimodel Databases like OrientDB, DataSTax in above comparison model. These databases uses different storage engine which needs to be studied and explored further. Also we will evaluate the databases performance based on complex graph algorithm execution like PageRank, Vertex Centrality, Community Detection, etc.

8 ACKNOWLEDGEMENT

The author would like to thank Computer Division, Bhabha Atomic Research Center, Mumbai, India for supporting the work.

9 REFERENCES

[1] A Critical Comparison of NOSQL Databases in the Context of Acid and Base Deepak GC St Cloud State University

[2] Ingo Friepoertner. Polyglot Persistence and Multi Model Databases, Open Source Data Center Conference, Berlin, Germany (Apr 21-23 2015).

[3] Analysis of various NoSQL database Pragati Prakash Srivastava; Saumya Goyal; Anil Kumar 2015 International Conference on Green Computing and Internet of Things (ICGCIoT)

[4] Ganesh Chandra Deka, “Fine A Survey of Cloud Database Systems”,“IT Professional” vol.16, Issue: 2, 2014, pp. 50-57, 03 January 2013

[5] MongoDB Labs “JSON and BSON” https://www.mongodb.com/jsonand-bson.

[6] Jagdev Bhogal and Imran Choksi, “Handling Big Data using NoSQL”,“IEEE Conference Publications on Advanced Information Networking and Applications Workshop (WAINA)”.

[7] International Journal of Advanced Research in Computer Science RESEARCH PAPER Available Online at www.ijarcs.info © 2010-14, IJARCS All Rights Reserved 170 ISSN No. 0976-5697
[8] A Comparative Study of NoSQL Databases.
[9] St. Cloud State University the Repository at St. Cloud State Culminating Projects in Information Assurance Department of Information Systems 5-2016 A Critical Comparison of NOSQL Databases in the Context of Acid and Base
[10] International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA Volume 5– No.4, March 2013 – www.ijais.org 16 Type of NOSQL Databases and its Comparison with Relational Databases
[11] Big Data: The NoSQL and RDBMS review Rashid Zafar; Eiad Yafi; Megat F. Zuhairi; Hassan Dao 2016 International Conference on Information and Communication Technology.
[12] Chaouki, Li., Yang, Wu., “The Distributed Storage Research of Remote Sensing Image Based on Mongo DB,”, Third International Workshop on Earth Observation and Remote Sensing, 2014.
[13] Yang, F., Milosevic, D., Cao, J., “An Evolutionary Algorithm for Column Family schema Optimization in Hbase,”, IEEE First International Conference on Big Data Computing Service and Applications (BigDataService),pp. 439-445,2015.
[14] Zarrinkamal, F., Kahani, M., Paydhar, S.,”Using Graph Database for for file Recommendation in PAD Social Network,”, 7th International Symposium on Telecommunications (IST), pp-470-475,2014.
[15] Leavitt, N.,”Will No SQL Live Up to Their Promise?”, white paper on IEEEcomputer society, vol. 10, no.9162, pp. 12-14, 2010.
[16] Rashid Zafar, Eiad Yafi, Megat F. Zuhairi, Hassan Dao. "Big Data: The NoSQL and RDBMS review", 2016 International Conference on Information and Communication Technology (ICICTM), 2016
[17] Kriti Srivastava, Narendra Shekokar. "A Polyglot Persistence approach for E-Commerce business model", 2016 International Conference on Information Science (ICIS),2016
[18] Benymol Jose, Sajimon Abraham. "Exploringthe merits of NoSQL: A study based on mongodb", 2017 International Conference on Networks & Advances in Computational Technologies (NetACT), 2017
[19] Fábio Roberto Oliveira, Luis del Val Cura. "Performance Evaluation of NoSQL Multi-Model Data Stores in Polyglot Persistence Applications", Proceedings of the 20th International Database Engineering & Applications Symposium on - IDEAS '16,2016