NoSQL Databases: Critical Analysis, Storage Efficiency, and Performance Benchmarking

Supriya Shrestha

Patan College for Professional Studies, Nepal

supriya.shrestha@patancollege.edu.np

Abstract—The exponential growth of *global data has rendered traditional Relational Database Management Systems (RDBMS) impractical for colossal workloads. NoSQL databases have emerged as the preferred solution for handling large-scale, unstructured data due to their horizontal scalability and flexible schema. This paper provides a critical analysis of NoSQL architectures, categorized into key-value, document, wide-column, and graph stores. We examine storage efficiency techniques (SETINS) using deduplication and compression to reduce cloud storage costs. Furthermore, we evaluate benchmarking results for OLAP workloads using Koalabench and YCSB. Finally, we discuss the integration of "SQL over NoSQL" frameworks (UniqueNOSD) and the development of "Text-to-NoSQL" translation models (SMART) to improve user accessibility. The findings suggest that while NoSQL offers superior performance for Big Data, selection must be based on specific consistency and analytical requirements.*

Keywords—NoSQL, Big Data, SETINS, SQL over NoSQL, CAP Theorem, OLAP Benchmarking.

I. INTRODUCTION

The digital era has ushered in a "Big Data" revolution where data is characterized by Volume, Velocity, and Variety. Traditional RDBMS models are designed for vertical scaling, which becomes cost-prohibitive as datasets reach petabyte scales [5]. NoSQL (Not Only SQL) databases were developed to address these limitations by providing horizontal scalability across commodity hardware clusters.

However, moving away from relational models introduces the CAP Theorem challenge: a distributed system can only provide two out of three guarantees: Consistency, Availability, and Partition Tolerance. Most NoSQL systems prioritize Availability and Partition Tolerance (AP) or Consistency and Partition Tolerance (CP), leading to the "BASE" (Basically Available, Soft state, Eventual consistency) consistency model [4].

II. ARCHITECTURAL CLASSIFICATIONS

NoSQL databases are differentiated by their data storage mechanisms. According to Gupta et al. [5], they are categorized into four major models:

A. Key-Value Stores

Data is stored as a collection of key-value pairs. This is the simplest NoSQL model, offering high performance for session management and caching. Examples include Redis and Riak.

B. Document Stores

These databases store data in documents (typically JSON, BSON, or XML). They provide high flexibility and are ideal for content management systems. MongoDB is the most prominent example, supporting complex nested structures [4].

C. Wide-Column Stores

Unlike RDBMS which stores data in rows, column stores group data in column families. This is highly effective for analytical queries and time-series data. Examples include Cassandra and Apache Kudu.

D. Graph Databases

These focus on the relationships between data entities using nodes and edges. They excel in social networking and fraud detection applications where data connectivity is paramount (e.g., Neo4j, ArangoDB).

III. STORAGE EFFICIENCY TECHNIQUES (SETINS)

As data volume grows, storage and bandwidth costs in cloud environments increase. Bhatia and Jangra [1] proposed the SETINS (Storage Efficiency Techniques in No-SQL) prototype.

A. Data Deduplication

Deduplication identifies and eliminates redundant copies of data. In NoSQL environments, where data is often replicated for availability, deduplication ensures that only one unique instance of data is physically stored, significantly reducing the storage footprint.

B. Data Compression

SETINS utilizes compression algorithms to reduce the size of data during internode transfers. Experimental results using Hadoop MapReduce and MongoDB demonstrate that combining deduplication with compression saves subsequent storage space and network bandwidth, which is vital for cloud-based designs [1].

IV. SYSTEM ARCHTECTURE OF SETINS

In figure 1, the flow diagram of the proposed framework has been shown in multiple layers. First data is fetched from Hadoop Distributed file system (HDFS). It is designed for storing very large data files with streaming data access patterns, running on clusters of various nodes that are hundreds of megabytes, gigabytes, or terabytes in size. [11]

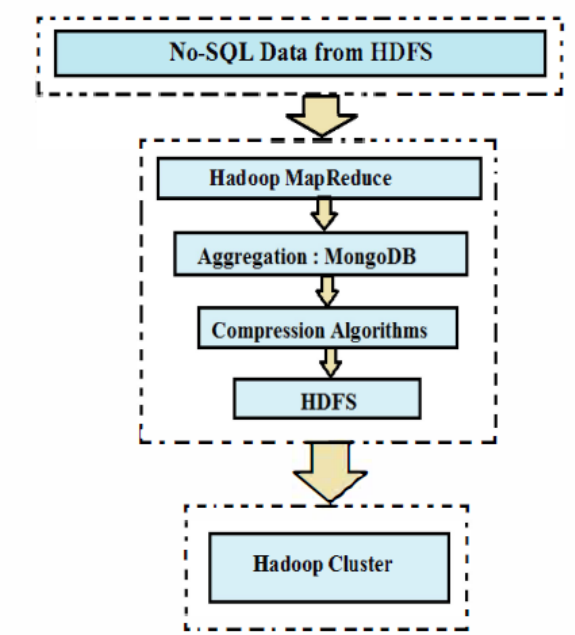

Figure I. Flow Chart of the Proposed System

Data are stored in the form of flat text files oftenly in cloud. It is not efficient to store data in this raw form because it increases OS overhead in fetching and storing the data. Traditional RDBMS is not a suitable platform to handle the bulk data. MapReduce works great in such Conditions. To applying SETiNS efficiently for large data, the file is first deduplicated using MapReduce paradigm in Hadoop framework. The output of MapReduce phase are the various key-value pairs, on the basis of which pointer table is created using MongoDB. Key-value pairs with the pointer table are now stored as a single document. Using MongoDB, a document may contain further various nested documents with different key-value pair and key-array pairs. So, MongoDB is used for this purpose, as document based NO-SQL storage is more efficient and easy in handling. [3]

IV. PERFORMANCE EVALUATION FOR OLAP

Online Analytical Processing (OLAP) requires complex transformations and high-speed query execution. Mohan et al. [4] conducted a benchmarking study comparing MongoDB, Redis, Kudu, and ArangoDB using a standardized pipeline.

A. Data Models: Flat vs. Snow

The research evaluated performance across two data models:

1. Flat Model: All attributes are in a single large table.
2. Snow Model: Attributes are normalized into multiple related tables.

B. Experimental Results

- Data Loading: Wide-column stores like Apache Kudu showed superior performance in loading large-scale analytical datasets compared to document stores.
- Query Execution: For complex analytical queries involving aggregations, NoSQL databases often outperformed standard SQL solutions like PostgreSQL due to their distributive properties and parallel processing capabilities [4].

V. BRIDGING THE SQL-NOSQL GAP

Despite the benefits of NoSQL, many organizations are reluctant to leave the robust query engines of SQL. This has led to two major research directions:

A. UniqueNOSD: SQL over NoSQL

Gidado and Ezeife [2] introduced UniqueNOSD, a framework designed to provide a SQL layer over NoSQL systems. This framework aims to mitigate data redundancy and provide the consistency

obtainable in relational databases without sacrificing the scalability of NoSQL. It addresses the drawbacks found in current systems like SparkSQL and Hive by optimizing the query engine layer.

B. Text-to-NoSQL Translation

To lower the technical barrier for non-experts, Lu et al. [3] introduced the Text-to-NoSQL task. They developed the TEND (Text-to-NoSQL Dataset) and the SMART framework. SMART uses Small Language Models (SLMs) and Retrieval-augmented Generation (RAG) to convert natural language questions into complex NoSQL queries (e.g., MongoDB's aggregation pipelines). While execution accuracy is currently around 44.76%, it represents a promising frontier for user-friendly database interaction [3].

VI. CHALLENGES AND LIMITATIONS

1. Complexity: Managing a distributed NoSQL cluster requires more operational expertise than a single-node RDBMS.
2. Lack of Standardization: Unlike SQL, which is standardized, every NoSQL database has its own unique API and query language.
3. Security: Early NoSQL adoptions often lacked robust security protocols, making them vulnerable if not properly configured with third-party tools [5].

VII. CONCLUSION

The transition from SQL to NoSQL is a response to the demands of modern data-intensive applications. This paper has analyzed the efficiency of storage through SETINS, the performance of various models in OLAP workloads, and the emergence of hybrid SQL-NoSQL frameworks. While NoSQL offers unparalleled scalability and flexibility, the trade-off in consistency remains a critical factor. Future research should focus on improving the accuracy of natural language interfaces and further optimizing the storage efficiency of distributed graph models.

REFERENCES

[1] V. Bhatia and A. Jangra, "SETiNS: Storage efficiency techniques in No-SQL database for Cloud based design," *2014 International Conference on Advances in Engineering & Technology Research (ICAETR)*, Unnao, India, 2014, pp. 1-5.

Gidado, A., & Ezeife, C. I. (2025). *UniqueNOSD: a novel framework for NoSQL over SQL databases. Journal of Big Data, 12*(255). https://doi.org/10.1186/s40537-025-01307-2

Lu, J., Song, Y., Qin, Z., Zhang, H., Zhang, C., & Wong, R. C.-W. (2025). *Bridging the gap: Enabling natural language queries for NoSQL databases through text-to-NoSQL translation*. arXiv. https://arxiv.org/abs/2502.11201

Mohan, R. K., Kanmani, R. R. S., Ganesan, K. A., & Ramasubramanian, N. (2024). *Evaluating NoSQL databases for OLAP workloads: A benchmarking study of MongoDB, Redis, Kudu and ArangoDB*. arXiv. https://arxiv.org/abs/2405.17731

Gupta, A., Tyagi, S., Panwar, N., & Sachdeva, S. (2017). *NoSQL databases: Critical analysis and comparison*. In *2017 International Conference on Computing and Communication Technologies for*

Smart Nation (IC3TSN) (pp. 1–?). IEEE. https://doi.org/10.1109/IC3TSN.2017.8284494