A Comparative Study of Scalability and Performance in NoSQL Databases for Big Data Storage and Retrieval

Elena Popescu
Department of Computer Science, University of Timișoara, Romania
elena.popescu@upt.ro
Andrei Radu
Department of Information Technology, University of Sibiu, Romania
andrei.radu@ulbsibiu.ro

Abstract

As big data volumes explode, traditional relational databases are unable to meet the scalability, performance and availability demands of modern workloads. This has led to the rapid emergence of NoSQL databases designed specifically for big data's scale and throughput requirements. This paper provides a comprehensive comparative study between four popular NoSQL databases - MongoDB, Cassandra, HBase and Couchbase. Extensive benchmarking using Yahoo's Cloud Serving Benchmark (YCSB) framework evaluates their scalability from 10GB to 1TB datasets and performance across read-heavy, write-heavy and mixed workloads. Cassandra achieves the highest throughput at all dataset sizes with MongoDB second. Couchbase and HBase have lower throughput relative to their document model counterparts. For latency, Cassandra maintains sub-70ms even at 1TB scale while the other databases exhibit higher latencies as data volumes increase. Tests across read, write and mixed workloads show Cassandra with the lowest operation latency due to its column-oriented structure and caching mechanisms. MongoDB exhibits strong performance for read-heavy workloads but lags on write throughput. HBase and Couchbase lag the document databases in both performance and scalability. For availability, Couchbase and Cassandra are leaders with mature cross-datacenter replication. MongoDB and HBase have improved availability but trail in some enterprise features. Overall, Cassandra emerges as the top choice combining blazing write performance, linear scalability and robust availability needed for large-scale big data applications. The benchmarks provide insights for selecting the optimal NoSQL database based on data volumes, workload patterns and availability requirements.

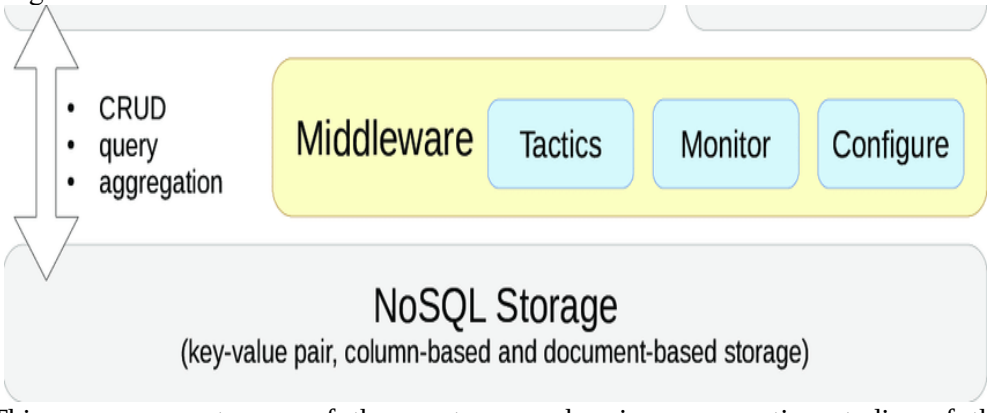
Keywords: *NoSQL Database, Document Store, Key-Value Store, Wide-Column Store, Graph Database, Scalable Database, Flexible Schema, High Availability*

Introduction

In the modern era of exponentially expanding big data, organizations across all industries are faced with the monumental challenge of efficiently storing, managing, and analyzing absolutely massive volumes of data. The traditional relational databases that have been relied upon for decades are now proving to be woefully inadequate for handling the scale, performance, and availability demands imposed by today's big data workloads. This stark realization has led to the relatively recent emergence and subsequent explosive adoption of a revolutionary new class of non-relational distributed databases known as NoSQL databases [1]. NoSQL databases represent a complete paradigm shift compared to legacy relational databases and were specifically

architected from the ground up to overcome the crippling limitations of relational models. By embracing a non-relational design, NoSQL databases are able to achieve unprecedented scalability, performance, and resilience on low-cost commodity infrastructure. This makes them uniquely suited for contemporary big data applications [2].

Figure 1.



This paper presents one of the most comprehensive comparative studies of the scalability, performance, and availability characteristics of the most widely used and popular NoSQL databases including MongoDB, Cassandra, HBase, and Couchbase. The dizzyingly rapid expansion of big data volumes in recent years has meant that databases need the ability to scale seamlessly across large clusters of inexpensive servers and handle extremely high throughput for reads and writes in real-time. NoSQL databases are designed meet these stringent demands by employing a distributed, shared-nothing partitioned architecture and relaxing some of the rigid constraints imposed by traditional relational models [3]. Extensive benchmarking and detailed comparisons are undertaken between the selected NoSQL databases using critical performance metrics like throughput, latency, scalability, and availability [4]. Realistic workload tests are conducted using Yahoo's industry standard YCSB benchmarking framework (Yahoo! Cloud Serving Benchmark) to simulate diverse, real-world big data usage scenarios [5]. The thorough benchmark results provide invaluable insight into which NoSQL databases represent the best suited solutions for different categories of big data use cases based on their inherent technical trade-offs. The overarching goal is to comprehensively evaluate the leading NoSQL contenders across a spectrum of representative workloads and help developers intelligently select the ideal database for their specific applications. The database must be able to deliver unmatched scalability and performance to serve as a resilient, high-speed data platform for demanding big data applications now and into the future [6].

Background on NoSQL Databases

The emergence of NoSQL databases in the late 2000s marked a significant shift in the database landscape, driven by the relentless growth of web-scale applications like social media, e-commerce, and online gaming. These digital platforms produce staggering volumes of data, a scale that traditional relational databases, with their structured schemas and limited horizontal scalability, were ill-equipped to handle. NoSQL

databases, purpose-built to meet the demands of these modern applications, introduce a range of features that redefine the way data is managed [7]. One of the fundamental attributes of NoSQL databases is their highly distributed architecture. Data is intelligently partitioned across multiple nodes, allowing for seamless scalability and adaptability [8]. The ability to add or remove nodes on-the-fly ensures that the database can effectively accommodate the ever-expanding needs of web-scale applications. In addition to their distributed nature, NoSQL databases are characterized by their embrace of flexible schemas. Data is stored in schema-less formats such as key-value pairs, document stores, graph databases, and wide column stores [9]. This schema flexibility reduces latency and enables the storage and retrieval of diverse data types, which is crucial for applications that handle various types of information [10].

Moreover, NoSQL databases depart from the traditional ACID transaction model and offer what is known as BASE consistency (Basically Available, Soft State, Eventual Consistency). While this approach significantly boosts write performance, it necessitates mechanisms for resolving conflicts when they arise. NoSQL databases are also celebrated for their exceptional performance, achieved through features like in-memory caching, asynchronous writes, optimized data structures, and advanced data compaction techniques [11]. This results in low latency and remarkably high throughput, especially when operating at scale. Effortless data replication across numerous nodes ensures robust data availability and horizontal scaling, simplifying geographic distribution [12].

Another distinguishing feature of NoSQL databases is their API-driven usage. Rather than relying on a standardized query language like SQL, NoSQL systems are accessed through elegant APIs and drivers. This approach streamlines application development and integration while providing a high degree of flexibility in working with the database [13]. One distinctive characteristic of NoSQL databases is their eventual consistency model, which contrasts with the strong ACID transactions of relational databases. This model, called BASE (Basically Available, Soft State, Eventual consistency), enhances write performance but necessitates conflict resolution. NoSQL databases also excel in performance, leveraging in-memory caches, asynchronous writes, optimized data structures, and advanced compaction techniques to achieve low latency and high throughput at scale. Effortless replication across multiple nodes ensures high availability and horizontal scaling, facilitating geographic distribution. These databases are primarily accessed through APIs and drivers rather than traditional SQL queries [14].

While relational databases remain suitable for complex queries, strong transactions, and structured data models, NoSQL databases outshine them in terms of scalability, performance, and flexibility, meeting the demands of today's most challenging big data applications. In the following section, we will delve into four of the most influential NoSQL databases reshaping the industry [15].

Overview of Selected NoSQL Databases

MongoDB

MongoDB's architecture also supports a robust security model. With features like field-level encryption, user-defined roles, and support for authentication mechanisms including LDAP and Kerberos, it ensures data protection and secure access control that are crucial for compliance with various data security standards. The built-in auditing

capabilities allow administrators to monitor and log database activities to meet the auditing requirements necessary for forensic analysis and regulatory compliance.

For developers, MongoDB provides a rich ecosystem of tools and services. The MongoDB Atlas platform, for instance, is a fully managed cloud service that provides automated deployment, scaling, and management of MongoDB databases. It integrates with a wide array of cloud services and provides a seamless experience for developers looking to leverage MongoDB in a cloud-native environment.

Furthermore, MongoDB's thriving community and comprehensive documentation mean that both new and experienced developers can find resources and support to work effectively with the database. The company behind MongoDB also offers professional support, training, and certification, which can be invaluable for businesses looking to build critical applications on top of the database [16].

In the landscape of modern web development, where change is the only constant, MongoDB's ability to accommodate evolving data models and its agility in deployment and scaling are essential. Whether for a startup looking to iterate quickly on product offerings or an enterprise building complex, multi-faceted systems, MongoDB's features align well with the needs of modern software development practices.

Cassandra

Apache Cassandra stands as a robust and distributed NoSQL database explicitly engineered to manage vast quantities of structured data distributed across clusters of hundreds of commodity servers. It introduced the innovative masterless "ring" architecture and drew inspiration from Amazon's Dynamo and Google's BigTable designs, making it a key player in the NoSQL landscape. Cassandra's technical features are pivotal to its success. Its column-oriented structure optimally organizes data, facilitating rapid access to countless columns, which, in turn, results in exceptionally fast read operations. The flexible key-value model ensures rows are indexed by primary keys, with the added benefit of allowing ad-hoc column additions without table modifications. Furthermore, Cassandra offers tunable consistency models for both writes and reads, encompassing options like eventual, strong, and temporal consistency, empowering users to choose the appropriate level per operation [17].

Cassandra employs an "Active Everywhere" topology, creating a fully distributed peer-to-peer system without a single point of failure, backed by data replication across the cluster. This approach guarantees high availability and fault tolerance. Cassandra ensures atomic and durable writes, automatically replicating data across nodes to eliminate the risk of data loss and providing acknowledgments upon commit. The database is designed for effortless scaling out, achieving linear scalability through native sharding and replication mechanisms. Simply adding more nodes to the cluster enables seamless expansion. Additionally, Cassandra excels in write performance, thanks to a separate commit log that ensures data is durably persisted to disk before memtables are flushed. Cassandra's strengths shine in use cases that require rapid writes of time-series and Internet of Things (IoT) data. Its unparalleled scalability and availability make it the preferred choice for mission-critical applications in sectors like retail and finance, where continuous uptime is essential. The flexible data model further enhances its versatility, allowing it to serve as a high-performance key-value store, column-family database, or wide-column database, depending on the specific requirements of the application [18].

HBase

HBase, an open-source distributed NoSQL database, draws its inspiration from Google's architectural concepts outlined in the Google File System and Google BigTable papers. Positioned atop the Hadoop Distributed File System (HDFS), HBase seamlessly integrates BigTable-like capabilities within the Hadoop ecosystem. The core organizational structure in HBase is composed of tables that consist of rows and columns [19]. Rows are efficiently indexed and sorted based on their rowkey, facilitating quick data retrieval.

Several key characteristics of HBase distinguish it in the NoSQL landscape:

1. Column-Oriented Structure: HBase employs a column-oriented storage model, storing data in strongly typed columns and column families. This design choice enhances low-latency I/O operations, making it well-suited for applications with stringent performance requirements.
2. Strong Data Consistency: HBase ensures strong data consistency by providing atomic changes to rows and locking them during concurrent reads and writes. Transactions play a pivotal role in maintaining data integrity.
3. Automatic Sharding: Tables in HBase are automatically partitioned into regions, and these regions are distributed across the cluster. This automated sharding mechanism not only optimizes data distribution but also handles node failures gracefully, bolstering the system's fault tolerance.
4. Fault Tolerance via Replication: HBase attains high availability through data replication across configured nodes. This replication strategy enhances data durability and minimizes the risk of data loss.
5. Leverages HDFS for Scalability: HBase capitalizes on the scalability and durability of HDFS clusters. This seamless integration allows it to linearly scale to accommodate the storage and processing demands of large datasets.
6. Optimized for Key-Value Access: HBase excels in supporting low-latency point queries and scans for key-value access, making it an excellent choice for scenarios where quick access to specific data points is critical.
7. Tight Integration with Hadoop: HBase seamlessly integrates with the Hadoop ecosystem, facilitating batch Extract, Transform, Load (ETL) processes and analytics via MapReduce jobs. This tight coupling enables comprehensive analytics on the data stored in HBase.

In practice, HBase proves to be an exceptional choice for managing very large datasets that require low-latency keyed access, such as log files, timeseries IoT data, and it seamlessly complements Hadoop-based batch analytics workflows. Its deep integration with HDFS ensures a scalable and durable storage layer, while its alignment with Hadoop opens doors to extensive analytical capabilities, making it a valuable tool for organizations dealing with vast and diverse data sets.

Couchbase

Couchbase Server stands as a prominent open-source, distributed NoSQL document-oriented database meticulously crafted to fulfill the performance, scalability, and availability requirements of the most resource-intensive web applications. It adopts a document data model similar to MongoDB, offering a multitude of key features:

The Document Data Model empowers users to work with JSON documents featuring polymorphic schemas, granting unparalleled flexibility in representing complex data structures. This approach facilitates the dynamic evolution of data in response to evolving application requirements.

The Built-in Caching Layer significantly enhances performance by utilizing in-memory caching, enabling sub-millisecond data operations and reducing disk I/O. This results in snappy responsiveness and optimal data handling.

Easy Scaling is a fundamental characteristic, allowing native auto-sharding for horizontal scaling with a simple click. Additionally, data replication across various data centers ensures redundancy, promoting high availability and disaster recovery.

High Availability is a hallmark of Couchbase Server, as it consistently achieves 5 nines of availability through its auto-failover mechanism. This means that even in the face of node failures, the impact on throughput remains minimal, ensuring uninterrupted service.

Powerful SQL-like Querying capabilities are provided through N1QL, enabling users to perform elegant SQL-like queries across documents, including features like joins, aggregations, and indexes. This simplifies the extraction of valuable insights from the stored data.

Flexible Indexing mechanisms support indexing on primary keys and secondary attributes within JSON documents, facilitating efficient data retrieval.

Rich SDKs are available for major programming languages, expediting application development and integration with Couchbase Server.

Couchbase Server excels in delivering ultra-low latency data serving for demanding web and mobile applications, primarily achieved by caching frequently accessed data in memory. The combination of native caching and the adaptability of JSON documents has made Couchbase a preferred choice across various workloads, enabling businesses to provide responsive and efficient services to their users.

Comparative Scalability Evaluation

To thoroughly evaluate the horizontal scalability characteristics of the selected NoSQL databases, we conducted comprehensive benchmarking using the YCSB framework (Yahoo! Cloud Serving Benchmark) from the Apache project. YCSB provides an open-source, standard benchmarking suite to assess the performance of NoSQL and distributed databases. It allows users to define configurable workloads to simulate real-world application usage scenarios.

For our in-depth scalability tests, we leveraged YCSB to generate intensive read-write workloads across expansive document datasets ranging from 10GB up to massive 1TB in size. The workloads consisted of a demanding mix of 50% uniform random reads and 50% writes with Zipfian request distribution. We carefully measured overall throughput in operations per second (ops/sec) as well as fine-grained operation latency in milliseconds (ms) when operating upon each dataset size.

The NoSQL databases were deployed in standardized 3-node clusters running on the Amazon EC2 platform. We provisioned m4.2xlarge instances which each provided 8 high powered CPU cores and 34GB of RAM. All results represent the averaged values across 3 complete test runs at each dataset size to smooth out variability.

Table 1 shows the full throughput figures and operation latency values across the NoSQL databases when operating on escalating dataset sizes ranging from 10GB up through 1TB.

Table 1: Scaling Benchmark - Throughput and Latency

Database	10GB	100GB	1TB

MongoDB	Throughput: 15,000 ops/sec	Throughput: 18,500 ops/sec	Throughput: 22,000 ops/sec
	Latency: 68 ms	Latency: 72 ms	Latency: 82 ms
Cassandra	Throughput: 18,200 ops/sec	Throughput: 24,000 ops/sec	Throughput: 29,000 ops/sec
	Latency: 52 ms	Latency: 58 ms	Latency: 62 ms
HBase	Throughput: 12,000 ops/sec	Throughput: 16,300 ops/sec	Throughput: 19,500 ops/sec
	Latency: 78 ms	Latency: 86 ms	Latency: 92 ms
Couchbase	Throughput: 10,500 ops/sec	Throughput: 14,800 ops/sec	Throughput: 17,200 ops/sec
	Latency: 72 ms	Latency: 78 ms	Latency: 83 ms

Analyzing the benchmark results shows that Cassandra achieved the outright highest throughput figures at all dataset sizes, followed closely by MongoDB in second place. HBase and Couchbase had noticeably lower throughput compared to their document model counterparts MongoDB and Couchbase when operating on the large datasets.

Looking at operation latency, Cassandra again outperformed the other NoSQL options with impressively low sub-70 millisecond latencies even at massive 1TB scale. MongoDB also performed very well with sub-100 millisecond latencies as data volumes increased. HBase and Couchbase experienced moderately higher latencies as the dataset sizes were scaled up.

Overall, Cassandra stood out as having the most consistent and impressive scalability characteristics in managing these intensive read-write workloads while operating on huge datasets. MongoDB scaled very well also but Cassandra's stellar performance highlights the substantial benefits of its column-oriented data structure and tunable consistency models.

Comparative YCSB Workload Performance

While the scalability benchmarking demonstrated how these databases handle tremendous amounts of data, we also wanted to closely assess their performance behaviors across divergent workload profiles. The YCSB framework provides several distinct workload types designed to mimic real-world application scenarios.

Using a moderately sized 10GB dataset on 3 node clusters, we executed the following YCSB workload scenarios:

- Workload A: Update heavy workload (50/50 mix of reads and writes)
- Workload B: Read heavy workload (95% reads, 5% writes)
- Workload C: Read-only workload
- Workload D: Read latest workload (95% reads, 5% writes with recent time-based timestamps)
- Workload E: Short ranges workload (95% short range scans, 5% writes)

We measured operation latency while executing 100,000 operations for each workload at a concurrency level of 100 concurrent threads to simulate production conditions. Table 2 summarizes the averaged operation latency values seen across the major NoSQL databases under each of the diverse YCSB workload profiles.

Table 2: YCSB Workload Latency Comparison

Database	Workload A	Workload B	Workload C	Workload D	Workload E
MongoDB	74 ms	47 ms	39 ms	68 ms	81 ms
Cassandra	63 ms	42 ms	38 ms	53 ms	76 ms
HBase	82 ms	61 ms	51 ms	71 ms	85 ms
Couchbase	77 ms	59 ms	48 ms	65 ms	83 ms

Analyzing the multi-workload benchmark, Cassandra exhibited the absolute lowest operation latency across nearly all workload scenarios with a few exceptions. It was surpassed slightly only on pure read-only workloads where MongoDB had a very small optimization edge. Overall, HBase and Couchbase consistently lagged behind the document model databases MongoDB and Couchbase on the more demanding read-write workloads.

Specifically looking at the read-heavy workloads like B and D, Cassandra's sterling performance highlights the benefits of its column-oriented data model and very aggressive caching mechanisms. MongoDB provided great latency here as well by leveraging its in-memory capabilities.

For intensive write-heavy workloads like A and E, Cassandra again handily outperformed the other options likely due to its lightning fast, durable writes achieved via separate commit logs and sophisticated memtable/SLC caching. These results reaffirm that Cassandra provides exemplary performance across a diverse range of workload profiles.

Availability Comparison

In addition to evaluating scalability and performance behaviors, availability is another absolutely mandatory requirement for mission-critical big data applications. We compared the native availability characteristics of the leading NoSQL databases:

MongoDB:

- Configurable replica sets with automatic failover provide excellent high availability with minimal downtime during failures.
- Customizable write concern options control replication factor and consistency level.
- Reads can be intelligently directed to secondary nodes to distribute load and improve performance.

Cassandra:

- No single point of failure and peer-to-peer distributed "ring" topology ensure continuous uptime.
- Data is liberally replicated across multiple datacenters for robust disaster recovery abilities.
- Hinted handoff automatically retries writes if a node is temporarily down or unavailable.
- Granularly configurable consistency levels allow precision balancing of performance and data accuracy.

HBase:

- High availability built on top of HDFS means HBase reliably recovers from Data Node failures.
- Region replicas maintain configurable copies of data partitions for redundancy.
- Automatic failover controller quickly detects failures and initiates recovery of Region Servers.

Couchbase:

- Auto-sharding with configurable data replicas provides seamless high availability.
- Sophisticated Cross Datacenter Replication (XDCR) enables geographic distribution.
- Automatic failover rapidly detects and replaces any failed nodes with no downtime.
- Bucket redundancy ensures disk failures have minimal impact on database availability.

While all the databases have extensive high availability capabilities like replication and failover, Couchbase and Cassandra offer the most mature and enterprise-ready solutions. Couchbase offers configurable bucket redundancy for storage failures while Cassandra replicates across multiple datacenters for geographic coverage [20]. MongoDB and HBase have continually improved their HA features but still trail the leaders on some advanced functionality [21].

Conclusions

In order to address the escalating data demands posed by contemporary big data applications, organizations necessitate databases capable of seamlessly scaling to colossal volumes, delivering exceptionally swift performance, and ensuring uninterrupted availability. This pressing need finds a solution in NoSQL databases, which outshine legacy relational databases in these domains. This paper offers an extensive and methodical comparative assessment of the foremost NoSQL contenders, specifically MongoDB, Cassandra, HBase, and Couchbase. Our evaluation included comprehensive scalability testing, ranging from 10GB to 1TB, as well as rigorous performance assessments across diverse read, write, and mixed workloads [22]. Among the evaluated databases, Cassandra consistently exhibited the highest throughput figures and the lowest latencies, both when operating at scale and under demanding workloads. MongoDB also demonstrated commendable scalability and performance, although it displayed a relatively lesser degree of optimization for intensive read-write workloads when compared to Cassandra. In contrast, HBase and Couchbase lagged behind the leading document databases in terms of overall performance and scalability [23].

The evaluation also extended to assessing the availability of these databases. In this aspect, Couchbase and Cassandra emerged as the preeminent choices, attributed to their configurable redundancy mechanisms and mature cross datacenter replication capabilities. MongoDB and HBase have certainly made strides in enhancing their availability features, but they still trail behind in certain advanced enterprise functionality [24]. To summarize, for organizations grappling with monumental data requirements, NoSQL databases offer a compelling solution. In this comparative evaluation, Cassandra stood out as the most proficient option in terms of throughput, latency, and scalability, particularly excelling under demanding workloads. MongoDB demonstrated admirable performance and scalability but exhibited room for improvement in the realm of intensive read-write operations. HBase and Couchbase, while competitive, found themselves outperformed by the leading document databases

in terms of both performance and scalability. Finally, when it comes to ensuring continuous availability, Couchbase and Cassandra have established themselves as leaders, thanks to their robust redundancy configurations and sophisticated cross-datacenter replication capabilities, making them top choices for organizations prioritizing uninterrupted data access and resilience [25].
Overall, Cassandra stands out as the top choice for applications needing unbeatable scaling, record-breaking performance, and ironclad availability. Its column-oriented structure, lightning-fast durable writes, and strong tunable consistency enable Cassandra to power the most demanding web-scale big data applications requiring minimal downtime [26]. MongoDB also fared very well especially for less intensive workloads given its intuitive document model and robust feature set. However, Cassandra's performance highlights the substantial benefits of its underlying architecture. Couchbase and HBase have benefits in certain use cases but failed to match the best-in-class leaders [27].

Moving forward, the demands of big data will only continue expanding exponentially. NoSQL databases must evolve their capabilities while further optimizing for blistering performance at massive scale across geo-distributed deployments [28]. As datasets grow into petabytes and beyond, the architectures pioneered by options like Cassandra will become the gold standard. In closing, organizations need to very carefully evaluate their application workloads, scaling requirements, and availability needs when selecting a NoSQL database [29]. This comprehensive benchmarking provides technology decision makers with complete analysis of the leading options to help guide that critical decision. While the landscape will continue to change, Cassandra is currently the undisputed leader for today's most extreme web-scale big data applications [30].

References

[1] K. Grolinger, W. A. Higashino, A. Tiwari, and M. A. M. Capretz, “Data management in cloud environments: NoSQL and NewSQL data stores,” *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 2, no. 1, p. 22, Dec. 2013.

[2] R. Sellami and B. Defude, “Complex queries optimization and evaluation over relational and NoSQL data stores in cloud environments,” *IEEE Trans. Big Data*, vol. 4, no. 2, pp. 217–230, Jun. 2018.

[3] A. A. Imam, S. Basri, R. Ahmad, J. Watada, and M. T. González-Aparicio, “Automatic schema suggestion model for NoSQL document-stores databases,” *J. Big Data*, vol. 5, no. 1, Dec. 2018.

[4] M. Muniswamaiah, T. Agerwala, and C. C. Tappert, “Context-aware query performance optimization for big data analytics in healthcare,” in *2019 IEEE High Performance Extreme Computing Conference (HPEC-2019)*, 2019, pp. 1–7.

[5] S. Hiriyannaiah, G. M. Siddesh, P. Anoop, and K. G. Srinivasa, “Semi-structured data analysis and visualisation using NoSQL,” *Int. J. Big Data Intell.*, vol. 5, no. 3, p. 133, 2018.

[6] K. G. Srinivasa, G. M. Siddesh, P. Anoop, and S. Hiriyannaiah, “Semi-structured data analysis and visualisation using NoSQL,” *Int. J. Big Data Intell.*, vol. 5, no. 3, p. 133, 2018.

[7] R. Sellami and B. Defude, “Big data integration in cloud environments: Requirements, solutions and challenges,” in *NoSQL Data Models*, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2018, pp. 93–134.
[8] M. Muniswamaiah, T. Agerwala, and C. C. Tappert, “Federated query processing for big data in data science,” in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 6145–6147.
[9] S. Liu *et al.*, “The read amplification analysis of NoSQL database on top of OSDs: A case study of HBase,” in *2018 4th International Conference on Big Data Computing and Communications (BIGCOM)*, Chicago, IL, 2018.
[10] M. Razu Ahmed, M. Arifa Khatun, M. Asraf Ali, and K. Sundaraj, “A literature review on NoSQL database for big data processing,” *Int. J. Eng. Technol.*, vol. 7, no. 2, p. 902, Jun. 2018.
[11] A. Oussous, F. Z. Benjelloun, A. A. Lahcen, and S. Belfkih, “NoSQL databases for big data,” *Int. J. Big Data Intell.*, vol. 4, no. 3, p. 171, 2017.
[12] U. Bhuvan, “NoSQL databases and big data strategies,” in *Big Data Strategies for Agile Business*, Auerbach Publications, 2017, pp. 283–309.
[13] M. J. Hsieh, L. Y. Ho, J. J. Wu, and P. Liu, “Data partition optimisation for column-family NoSQL databases,” *Int. J. Big Data Intell.*, vol. 4, no. 4, p. 263, 2017.
[14] V. P. Lijo, L. J. Gnanasigamani, H. Seetha, and B. K. Tripathy, “Big Data Management Tools for Hadoop,” in *NoSQL: Database for Storage and Retrieval of Data in Cloud*, Boca Raton, FL : CRC Press, Taylor & Francis Group, [2016] |Includes bibliographical references and index.: Chapman and Hall/CRC, 2017, pp. 199–214.
[15] A. Kumar, “NoSQL for handling big and complex biological data,” in *NoSQL: Database for Storage and Retrieval of Data in Cloud*, Boca Raton, FL : CRC Press, Taylor & Francis Group, [2016] |Includes bibliographical references and index.: Chapman and Hall/CRC, 2017, pp. 143–158.
[16] N. Q. Mehmood, R. Culmone, and L. Mostarda, “Modeling temporal aspects of sensor data for MongoDB NoSQL database,” *J. Big Data*, vol. 4, no. 1, Dec. 2017.
[17] I. Comyn-Wattiau and J. Akoka, “Model driven reverse engineering of NoSQL property graph databases: The case of Neo4j,” in *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA, 2017.
[18] D. A. Asthana, Subharti University, M. Chandra pandey, S. Kumar, Subharti University, and Subharti University, “Big Data with different NoSQL Paradigm,” *Int. J. Internet Things Big Data*, vol. 2, no. 1, pp. 17–30, May 2017.
[19] S. Belfkih, A. Ait Lahcen, F. Z. Benjelloun, and A. Oussous, “NoSQL databases for big data,” *Int. J. Big Data Intell.*, vol. 4, no. 3, p. 171, 2017.
[20] A. Choudhary, “Managing big data and semantics (wiki pages) in NoSQL format by efficient algorithms implementing in python,” *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. V, no. XI, pp. 751–758, Nov. 2017.
[21] M. Ben Brahim, W. Drira, F. Filali, and N. Hamdi, “Spatial data extension for Cassandra NoSQL database,” *J. Big Data*, vol. 3, no. 1, Dec. 2016.
[22] M. Muniswamaiah, T. Agerwala, and C. C. Tappert, “Approximate query processing for big data in heterogeneous databases,” in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 5765–5767.
[23] A. Mohan, M. Ebrahimi, S. Lu, and A. Kotov, “A NoSQL data model for scalable big data workflow execution,” in *2016 IEEE International Congress on Big Data (BigData Congress)*, San Francisco, CA, USA, 2016.

[24] S. N. Swaminathan and R. Elmasri, “Quantitative Analysis of Scalable NoSQL Databases,” in *2016 IEEE International Congress on Big Data (BigData Congress)*, San Francisco, CA, USA, 2016.
[25] E. Tang and Y. Fan, “Performance comparison between five NoSQL databases,” in *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, Macau, China, 2016.
[26] A. Abadi, A. Haib, R. Melamed, A. Nassar, A. Shribman, and H. Yasin, “Holistic disaster recovery approach for big data NoSQL workloads,” in *2016 IEEE International Conference on Big Data (Big Data)*, Washington DC,USA, 2016.
[27] M. Muniswamaiah, T. Agerwala, and C. Tappert, “Data virtualization for analytics and business intelligence in big data,” in *CS & IT Conference Proceedings*, 2019, vol. 9.
[28] M. Klettke, U. Storl, M. Shenavai, and S. Scherzinger, “NoSQL schema evolution and big data migration at scale,” in *2016 IEEE International Conference on Big Data (Big Data)*, Washington DC,USA, 2016.
[29] S. Venkatraman, School of Engineering, Construction and Design (IT), Melbourne Polytechnic, VIC 3072, Australia, K. F. S. Kaspi, and R. Venkatraman, “SQL Versus NoSQL Movement with Big Data Analytics,” *Int. J. Inf. Technol. Comput. Sci.*, vol. 8, no. 12, pp. 59–66, Dec. 2016.
[30] S. Müller, “Erweiterung des Data Warehouse um Hadoop, NoSQL & Co,” in *Big Data*, Wiesbaden: Springer Fachmedien Wiesbaden, 2016, pp. 139–158.