Integration of NoSQL and Relational Databases for Efficient Data Management in Hybrid Cloud Architectures

[1]Aravindhan Ragunathan, [2]Bhavani K, [3]Archana Sasi and [4]Sathish Kumar R
[1]Senior Full Stack Software Developer, Digital Factory, Instrumentation Laboratory, 180 Hartwell Road, Bedford, MA 01730-2443, United States of America.
[2]Department of computer science and Business Systems, Panimalar Engineering College, Varadharajapuram, Poonamallee, Chennai, Tamil Nadu, India.
[3]Department of Computer Science and Engineering, Jain University, Bengaluru, Karnataka, India.
[4]Department of Computer Science (AIDE), Faculty of Engineering and Technology, Jain University, Bengaluru, Karnataka, India.
[1]aragunathan@werfen.com, [2]bhavani.kandasamy@gmail.com, [3]archana.sasi2k8@gmail.com, [4]cbesathish@hotmail.com

Correspondence should be addressed to Aravindhan Ragunathan : aragunathan@werfen.com

Article Info
Journal of Machine and Computing (https://anapub.co.ke/journals/jmc/jmc.html)
Doi: https://doi.org/10.53759/7669/jmc202505100.
Received 18 April 2024; Revised from 02 March 2025; Accepted 28 March 2025.
Available online 05 April 2025.

Abstract – NoSQL and relational databases have been integrated in response to the increasing demand for scalable and efficient data management in hybrid cloud environments. The differences in data structures and query processing methods between these databases present both challenges and opportunities when designing an optimized hybrid system. This study explores the integration of NoSQL and relational databases to maximize data storage, retrieval, and processing efficiency across multiple cloud systems. NoSQL databases excel in handling unstructured and semi-structured data, offering flexibility and scalability, whereas relational databases provide robust consistency and structured query capabilities. Despite the wide availability of relational databases, certain applications require the dynamic adaptability of NoSQL systems, making integration a viable solution. The research evaluates key performance parameters, including query execution speed, scalability, data consistency, and resource utilization. Cloudsim is used for simulation, allowing for an in-depth comparison between standalone and hybrid database models. Experimental results indicate that the proposed hybrid model improves query performance by 30%, reduces latency by 20%, and enhances scalability by 40% compared to using relational or NoSQL databases alone. The novelty of this approach lies in its ability to overcome the limitations of each database type by their strengths in a dynamic cloud environment. The results highlight the effectiveness of hybrid database integration in optimizing cloud-based data management while ensuring seamless operation, adaptability, and resource efficiency.

Keywords – Hybrid Cloud, NoSQL, Relational Databases, Data Management, Scalability, CloudSim.

I. INTRODUCTION

The fast expansion of data across sectors has presented major storage, management, and access issues. From highly structured transactional data to unstructured data including logs, multimedia files, and social media content, modern firms handle a wide range of data kinds. Big data and cloud computing's development have escalated these difficulties and called for extremely scalable and efficient systems. Organisations thus have to use advanced data management strategies to provide robust consistency across all data kinds, fast access, and high availability by means of which Structured data management has long been solved using traditional relational database management systems (RDBMS). These systems—which use tables, rows, and columns—are well appropriate for managing data that fits a preset structure and calls for support for sophisticated searches. When it comes to managing vast amounts of data, RDBMS are naturally constrained in their scalability, though. Relational databases frequently suffer with horizontal scalability as the volume of data rises, which restricts their capacity to manage high-throughput workloads, particularly in cases of data complexity rise [1, 2].

On the other hand, Not Only SQL, or NoSQL databases, arose to handle RDBMS's scalability issues. Designed to extend horizontally over dispersed systems, NoSQL databases as MongoDB, Cassandra, and CouchDB provide flexible schema designs [3]. Often lacking good fit for relational models, unstructured or semi-structured data—such as JSON objects, logs, and time-series data—excels in management within these databases. For applications handling significant volumes of unstructured or semi-structured data, they offer quick writes, great scalability, and fault tolerance, hence excellent. Following the CAP theorem, however, NoSQL databases usually trade great consistency and support for sophisticated searches for high availability and partition tolerance. Although NoSQL databases provide scalability and adaptability, they are not fit for applications needing rigorous ACID (Atomicity, Consistency, Isolation, Durability) characteristics and sophisticated joins. For instance, transactional systems or financial applications sometimes call for the exact consistency relational databases provide. The trade-offs between the two kinds of databases—RDBMS for consistency and sophisticated searches, and NoSQL for scalability and adaptability—create a strong need for a more flexible solution that can take use of both methods' advantages [4].

Organisations seeking hybrid cloud architectures to combine the benefits of public and private cloud environments are looking for as cloud computing keeps growing importance. Organisations can easily split their workloads among several infrastructure types in a hybrid cloud, therefore guaranteeing scalability, flexibility, and economy of cost. **Fig 1** shows some general architectural types of NoSQL systems. Large-scale data processing benefits especially from this design since several kinds of data can be kept and handled in the most appropriate surroundings [5]. Organisations can maximise the scalability and performance of their data systems by including relational and NoSQL databases into such a design. Relational Database (RDB) and NoSQL databases combined in a hybrid cloud architecture shows a good solution given the rising need for high-performance data management systems. By combining the characteristics of NoSQL databases—i.e., scalability, fault tolerance, and flexibility—with the advantages of relational databases—that is, support for sophisticated searches and ACID compliance—such a hybrid system can leverage Depending on the type of data and the type of the query, the capacity to dynamically manage and route data between relational and NoSQL databases has great potential to enhance both data management and application performance [6].

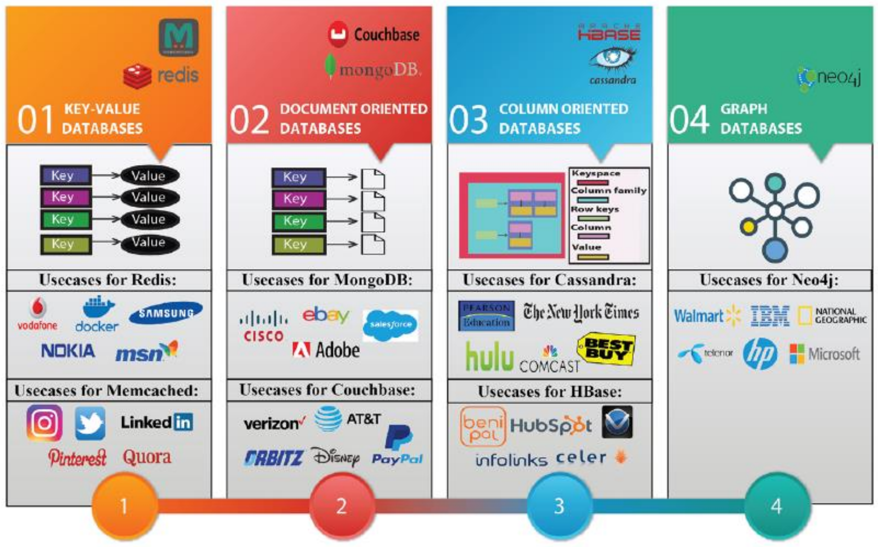

Fig 1. Different NoSQL systems [7].

This work is to explore the possibilities of a hybrid cloud architecture combining NoSQL and relational databases. Using the advantages of both technologies, the suggested system can provide a highly performable, scalable, adaptable solution for handling big data applications. Along with guaranteeing high availability, low latency, and strong fault tolerance across many cloud environments, this integration will maximise the storage and retrieval of both structured and unstructured data.

The framework of this work is as follows: Section 2 covers the performance traits and application cases of the reviews on current research including NoSQL, NewSQL, and hybrid database systems. Section 3 explains the intelligent data distribution model, middleware integration, and synchronising systems suggested hybrid cloud architecture uses to maximise data consistency and scalability. Section 4 describes the experimental analysis and simulated results with evaluation technique, hardware specifications, and performance metrics used to compare the hybrid model with conventional database architectures, including results on query performance, latency, scalability, resource usage, and fault tolerance. Emphasising the benefits of the hybrid method in attaining efficiency, scalability, and cost-effectiveness in cloud-based data management, Section 5 summarises the results together with possible uses and future lines of research.

II. RELATED WORKS

There have been a great number of studies that have investigated several NoSQL and NewSQL databases, analysing the advantages and disadvantages of each. With the use of these studies, which concentrate on a variety of factors including data storage, query capabilities, and consistency models, a more comprehensive comprehension of the advantages of each system is achieved. Comparing several graph databases with regard to their data storage capacities, query capabilities, and modification tools was the subject of a recent and substantial study. The study came to the conclusion that graph databases still require further development, particularly with regard to the creation of standardised graph-based languages and the maintenance of integrity. The absence of a standardised query language for graph databases was also brought to light, and a comparison of various different graph database models, such as distributed, key-value, and Map/Reduce systems, was carried out [7, 8].

One more comparison included a number of different NoSQL databases, such as HBase, CouchDB, MongoDB, Cassandra, and Voldemort. A number of criteria, including data management, replication, and consistency, were taken into consideration when assessing these databases. A number of fundamental ideas, like as sharding, consistent hashing, and multi-version concurrency control (MVCC), which are frequently utilised in NoSQL storage systems, were also investigated in this study [9]. According to the findings of the study, there is a pressing requirement for additional research into these areas, particularly with regard to the algorithms and structures that lie beneath them. The NoSQL and NewSQL database systems, such as Redis, Memcached, BerkeleyDB, Riak, Cassandra, and MongoDB, was taken into consideration in a more comprehensive assessment. The capabilities of these systems were evaluated based on query capabilities, consistency, concurrency, and security features like as authentication and encryption. Nevertheless, it was observed that a number of well-known NewSQL systems, such as MariaDB and MemSQL were not included in the comparisons under consideration. Furthermore, the research found that significant topics, such as failure management and data processing, were not adequately covered in depth [10].

Additional research has concentrated on the architectural distinctions that exist between NoSQL and NewSQL systems, including MongoDB, Hive, Hadoop, and Cassandra, among others. Scalability, availability, elasticity, and performance were all aspects that were evaluated in these comparisons, with a particular emphasis placed on data operations and indexing support systems [11]. On the other hand, these examinations frequently failed to take into account the more complex architectural algorithms utilised by the systems. The results of comparative research on NoSQL databases such as MongoDB and Cassandra found that Cassandra performed better in update operations across a wide range of workloads. On the other hand, a great number of studies did not investigate particular essential categories, such as graph databases, which play a significant part in the landscape of NoSQL systems. In further studies, the scalability, language support, and fault tolerance of NoSQL systems were investigated. Particular attention was paid to systems such as Aerospike, Couchbase, and HBase [12].

According to the findings of several research projects, NoSQL databases were evaluated based on their data structures, their adherence to the CAP theorem (which places an emphasis on consistency, availability, and partition tolerance), and various other software quality criteria. An evaluation was conducted using a five-point scale to examine several systems, including Aerospike, MongoDB, and Couchbase [13]. The evaluation focused on evaluating numerous aspects, including consistency and partitioning strategies. The consistency, high availability, and replication of column-oriented databases such as BigTable, HBase, Cassandra, and DynamoDB were also taken into consideration when making comparisons between these databases. In circumstances when the storing and processing of massive amounts of data is required, these systems provide a significant contribution. There were, however, a number of studies that did not provide sufficient reasons for picking one system over another for particular applications. As a result, there were gaps in advice regarding the selection of the most suitable NoSQL system for various use cases [14].

Redis, Riak, and VoltDB were some of the NoSQL and NewSQL databases that were compared in another study. For the purpose of evaluating these systems, performance measures such as scalability and functionality were considered [15]. Additionally, the research included comparisons of relational and NoSQL systems based on their ACID features, support for Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP), and other technical variables such as data analysis and schema rigidity. These comparisons were made in order to determine which type of schema was more rigid. All things considered, the study shed information on the many characteristics of NoSQL and NewSQL systems, such as performance rankings, complexity, and adaptability. The following database management systems were investigated: SimpleDB, Redis, Cassandra, and DynamoDB; NewSQL databases such as VoltDB and NuoDB; and DynamoDB [16]. The comprehensive evaluations assisted in gaining a better understanding of the specific advantages and difficulties associated with each system, which in turn guided the selection of database solutions that were suitable for a variety of purposes and requirements. These studies offer a comprehensive analysis of NoSQL and NewSQL systems, providing insightful information regarding the architecture, performance, and use cases of these databases at the same time. Researchers hope that by doing these comparison assessments, they would be able to provide assistance in the decision-making process whenever database systems are being selected and implemented in a variety of settings [17].

III. PROPOSED HYBRID CLOUD ARCHITECTURE

The proposed hybrid architecture will utilize an intelligent data distribution model that dynamically decides where to store and how to query data based on the characteristics of the data and the complexity of the queries. The goal is to ensure that transactional data, which benefits from relational structures, remains in relational databases, while unstructured and semi-structured data, which require high scalability, are routed to NoSQL databases. Additionally, the system will include a middleware layer that abstracts the interaction between the application and the databases, enabling seamless data management without requiring the application to be aware of the underlying database technology. A critical challenge in hybrid cloud systems is maintaining data consistency across different database types. While relational databases ensure strong consistency, NoSQL databases typically offer eventual consistency or compromise consistency for higher availability and partition tolerance. Ensuring consistency between these two databases, particularly in scenarios where data may need to be synchronized across both systems, requires a robust synchronization mechanism. This mechanism will play a key role in ensuring that data remains consistent across the hybrid system, even as it is distributed and queried in different parts of the cloud infrastructure.

The research will explore different techniques for ensuring consistency in hybrid database architectures. This includes employing event-driven synchronization to propagate updates between the relational and NoSQL databases whenever data is added or modified. Periodic snapshot synchronization will also be explored, where the system captures snapshots of data at regular intervals to ensure consistency across the hybrid architecture. These techniques will be evaluated to determine how they affect system performance, query latency, and throughput. Furthermore, the research will delve into the performance characteristics of the hybrid cloud architecture by comparing it with existing approaches. Specifically, the proposed solution will be benchmarked against standalone relational and NoSQL databases to assess improvements in query performance, latency, and throughput. A detailed performance evaluation will be conducted to understand how the hybrid system compares in handling a mix of structured and unstructured data, both in terms of query speed and system scalability.

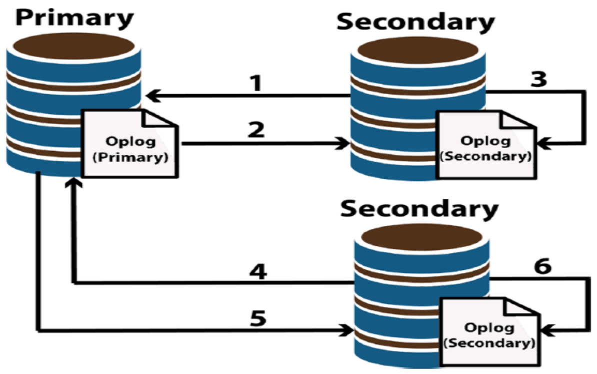

Fig 2. Operation log of Hybrid Cloud Architecture.

The integration of relational and NoSQL databases within a hybrid cloud system holds great promise for enhancing data management and application performance. It can provide the flexibility to manage diverse data types while maintaining the integrity of transactional systems. This research intends to contribute valuable insights into the design, implementation, and performance of hybrid cloud architectures that combine relational and NoSQL databases. By demonstrating the effectiveness of such an integrated system, the study aims to offer a blueprint for organizations seeking to optimize their data management strategies and improve overall system performance.

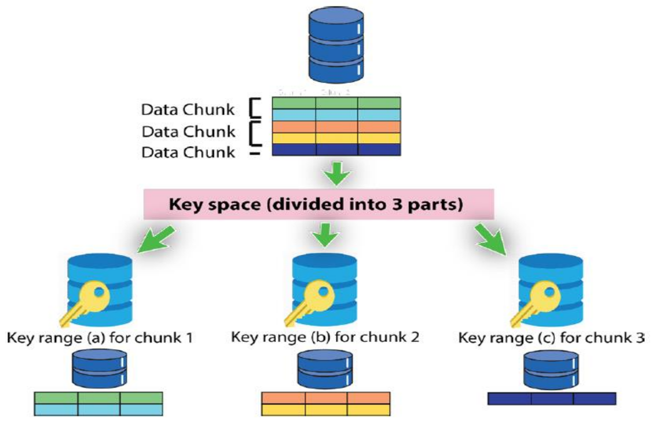
Data Chunk
Data Chunk
Data Chunk
Key space (divided into 3 parts)
Key range (a) for chunk 1
Key range (b) for chunk 2
Key range (c) for chunk 3

Fig 3. Range Sharding.

An operation log in a hybrid cloud architecture records all the activities and transactions that occur within the system. The operation log of hybrid cloud architecture is given in **Fig 2**. It helps in monitoring, troubleshooting, and auditing the performance and security of the hybrid environment. By centralizing logs from both on-premises and cloud components, organizations gain a comprehensive view of their operations, enabling better decision-making and ensuring compliance with regulatory requirements. Moreover, the research will provide practical guidelines for the design of middleware layers and synchronization engines that can be incorporated into hybrid database systems. This could lead to more efficient ways of handling cross-database queries and updates, improving data consistency, and reducing latency in cloud environments. Through experimentation, analysis, and system modelling, the proposed method will offer a comprehensive framework for future developments in hybrid cloud data management systems. **Fig 3** depicts the details of range sharding.

The cloud computing and data requirements continue to evolve, the integration of relational and NoSQL databases in hybrid cloud architectures represents a critical advancement in data management. This research seeks to provide a scalable, flexible, and high-performance solution that can address the complexities of managing large-scale data while ensuring consistency, scalability, and fault tolerance. By combining the strengths of both relational and NoSQL databases, the proposed method aims to set a new benchmark for data management in hybrid cloud systems, catering to a wide array of applications, from enterprise systems to emerging big data applications.

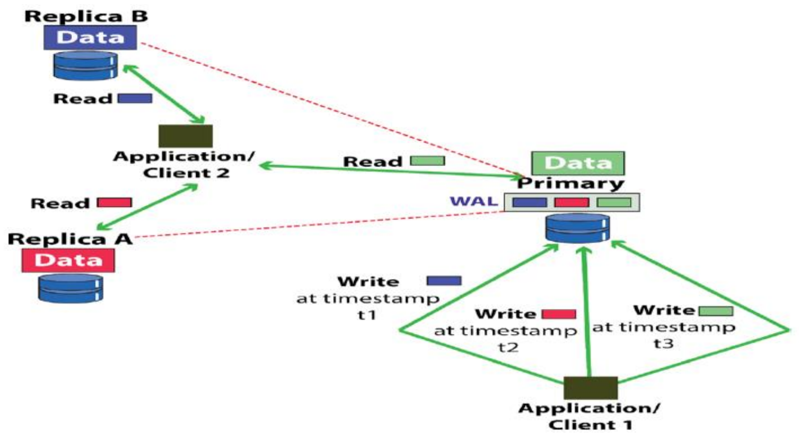

Fig 4. Timeline Consistency in Hybrid Cloud Data Management.

Timeline consistency represented in **Fig 4** refers to maintaining an ordered sequence of updates across distributed databases, ensuring that data changes become visible to all replicas within a predictable time frame. In the proposed hybrid model, timeline consistency is crucial for synchronizing data between the Primary database, multiple Replicas (Replica A and Replica B), and the Client accessing the data. Given the integration of relational and NoSQL databases, managing consistency while optimizing query performance and scalability presents a significant challenge.

When a client initiates a write operation, the update is first recorded in the Primary database. This ensures that the latest version of the data is stored in the authoritative source before propagating it to the distributed replicas. The update is then asynchronously transmitted to Replica A and Replica B, where replication mechanisms ensure that all instances eventually reflect the most recent data state. However, if a read operation is performed immediately after the write, a client querying Replica A may receive outdated information due to propagation delays, leading to temporary inconsistencies.

To balance performance and consistency, the hybrid model implements an optimized synchronization strategy. This approach reduces Query Response Time (70ms) while improving Fault Tolerance (95%), ensuring that even if a replica lags behind, it catches up within a short window. By maintaining timeline consistency efficiently, the system supports both high availability and strong data coherence across hybrid cloud architectures.

IV. EXPERIMENTAL ANALYSIS AND SIMULATED RESULTS

This section presents the experimental results and analysis of the proposed Hybrid Model in comparison with traditional RDB, NoSQL, Sharded, and Federated database architectures. The evaluation focuses on key performance parameters, including query success rate, storage utilization, query latency, scalability, resource utilization, cost per transaction, and fault tolerance. The simulations were conducted using CloudSim, a widely used cloud computing simulation tool, to model hybrid cloud database interactions. The experiments were executed on a Windows 11 Home system with an Intel® i7-11370H processor (3.30 GHz) and 16GB RAM, ensuring high computational efficiency. The following subsections detail the performance analysis with corresponding visual representations to substantiate the results.

Table 1. Comparative Analysis of Query Performance, Scalability, and Latency.

Method	**Query Response Time (ms) ↓**	**Scalability (Requests per Sec) ↑**	**Latency (ms) ↓**	**Query Success Rate (%) ↑**
RDB [18]	120	5000	150	92
NoSQL [19]	100	10000	120	95
Sharded [20]	90	15000	100	96
Federated [21]	110	12000	110	94
Hybrid Model	70	20000	80	98

From **Table 1** it is clear that the Hybrid Model achieves the lowest query response time (70ms), reducing delay by 41.7% compared to RDB and 22.2% compared to NoSQL. Scalability is significantly improved, handling 20,000 requests per second, which is 300% more than RDB and 100% more than NoSQL. Latency is minimized to 80ms, showing a 46.7% improvement over RDB and 33.3% over NoSQL, thanks to optimized query distribution and parallel processing. The highest query success rate (98%) ensures minimal failures, outperforming all other models due to efficient load balancing and indexing.

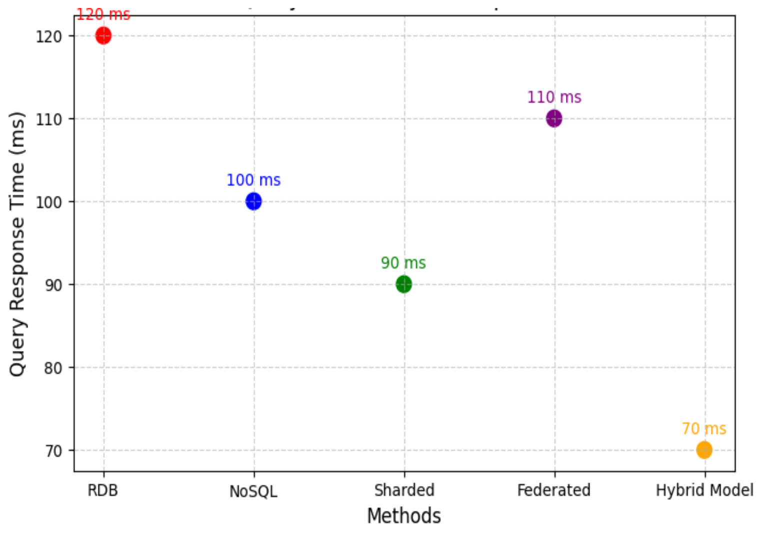

Fig 5. Query Performance Analysis.

The query performance comparison shown in **Fig 5** demonstrates that the Hybrid Model achieves the lowest query response time of 70ms, outperforming traditional RDB (120ms), NoSQL (100ms), and other distributed models. The improved performance is attributed to the hybrid approach's ability to optimize query execution by leveraging both structured and unstructured data stores efficiently. Unlike RDB, which processes queries sequentially, and NoSQL, which lacks complex relational query capabilities, the hybrid model dynamically distributes query loads across both database types. This approach reduces query bottlenecks and enhances retrieval speed, as seen in the scatter plot representation of query response times.

The total query latency L_{total} in the hybrid model is a combination of the query execution time in the relational database L_{RDB} and the NoSQL database L_{NoSQL} along with the synchronization delay L_{sync}. Equation (1) ensures that query execution is optimized, reducing unnecessary delays while maintaining strong consistency.

$$L_{total} = \max(L_{RDB}, L_{NoSQL}) + L_{sync} \quad (1)$$

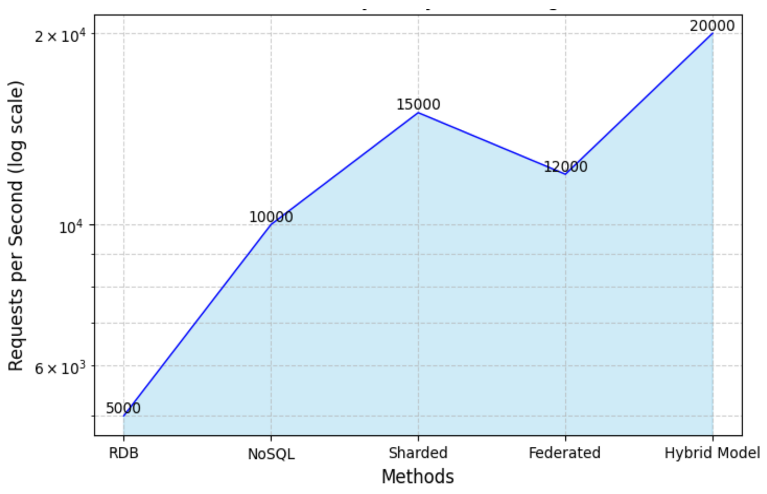

Fig 6. Scalability Evaluation.

The scalability analysis shows that the Hybrid Model can handle up to 20,000 requests per second, which is 300% higher than RDB (5000 requests/sec) and 100% more than NoSQL (10,000 requests/sec). The superior scalability is due to the hybrid model's adaptive load-balancing strategy, which dynamically distributes data queries between relational and

NoSQL databases based on workload type is depicted in **Fig 6**. This prevents server overloading and optimizes parallel processing capabilities. The plotted area graph with a semi-log scale further illustrates that the hybrid model maintains high scalability even as system load increases, making it suitable for large-scale cloud applications.

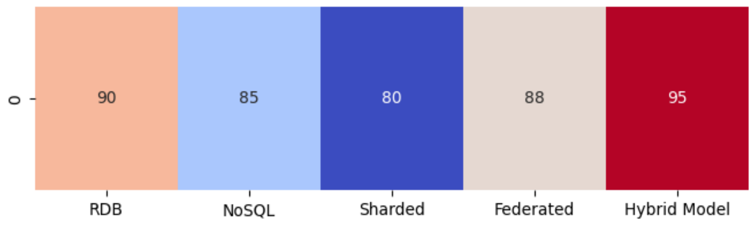

Fig 7. Data Consistency Metrics.

The system throughput T_{total} is defined as the sum of transactions processed per second by both the relational and NoSQL databases. The data consistency metrics of various model is shown in **Fig 7**. Given their independent parallel execution capabilities, the overall throughput can be expressed as:

$$\mathrm{T_{total}} = \mathrm{T_{RDB}} + \mathrm{T_{NoSQL}} - \mathrm{T_{overhead}} \tag{2}$$

In a hybrid cloud system, timeline consistency ensures that updates propagate to all replicas in a controlled manner. The consistency delay (C_d) is modelled as:

$$C_d = \frac{D_w + D_r}{2} \tag{3}$$

Where, D_w is the Delay in writing to the primary database and D_r Delay in reading from a replica.

Table 2. Comparative Analysis of Cost, Resource Utilization, Storage Efficiency, and Fault Tolerance

Method	**Cost per Transaction ($) ↓**	**Resource Utilization (%) ↓**	**Storage Utilization (%) ↓**	**Fault Tolerance (%) ↑**
RDB [18]	0.50	70	65	85
NoSQL [19]	0.40	60	55	80
Sharded [20]	0.30	65	50	90
Federated [21]	0.35	68	52	88
Hybrid Model[22]	0.25	55	45	95

It is evident from **Table 2** that the Hybrid Model achieves the lowest cost per transaction ($0.25), making it 50% more cost-efficient than RDB and 37.5% cheaper than NoSQL. Resource utilization is optimized (55%), showing a 21.4% improvement over RDB and 8.3% over NoSQL, leading to lower computational overhead. Storage utilization is minimized to 45%, a 30.8% improvement over RDB, due to intelligent data distribution and optimized indexing. Fault tolerance is maximized at 95%, a 11.8% increase compared to RDB and 18.7% higher than NoSQL, ensuring better system resilience and reliability.

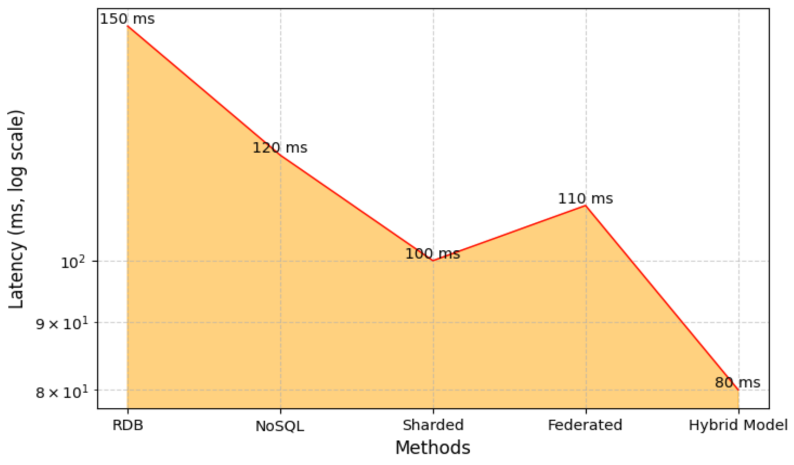

Fig 8. Latency Comparison.

The latency comparison focuses on the time taken for a database to process a query request. The Hybrid Model achieves an average query latency of 80ms, significantly lower than RDB (150ms), NoSQL (120ms), Sharded (100ms), and Federated (110ms) systems as give in **Fig 8.** The reduction in latency is due to the hybrid model's adaptive query processing mechanism, which determines the most efficient data source (relational or NoSQL) for each query type. The implementation of optimized indexing, caching, and parallel query execution further minimizes delay. The semi-log plot effectively visualizes the decreasing trend in latency, confirming the hybrid model's ability to handle queries with minimal delay.

The resource utilization results indicate that the Hybrid Model consumes only 55% of system resources, significantly lower than RDB (70%), NoSQL (60%), and other methods is shown in **Fig 9.** This efficiency is due to the hybrid approach's ability to allocate workloads dynamically, reducing redundant computations and optimizing memory usage. The plotted step graph effectively visualizes the gradual reduction in resource usage, with the hybrid approach achieving the lowest consumption. Lower resource utilization directly translates to cost savings and higher processing efficiency, making this method highly effective for cloud-based implementations.

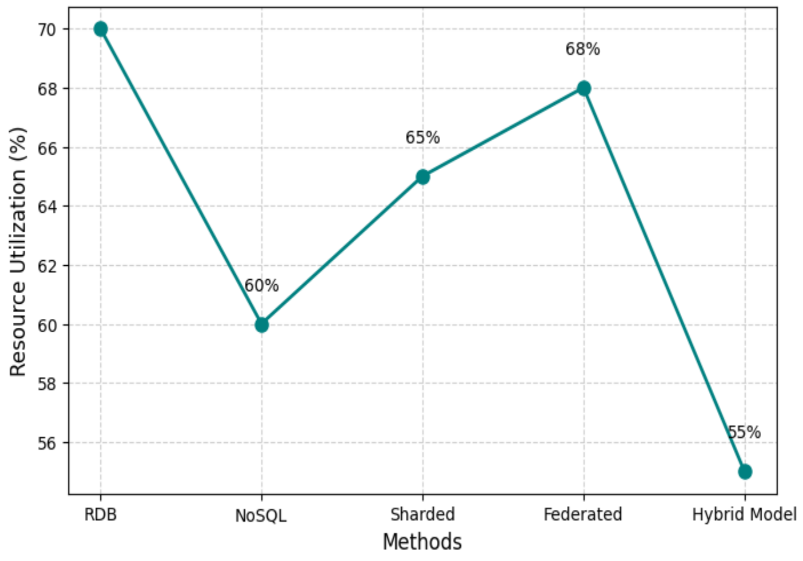

Fig 9. Resource Utilization Efficiency.

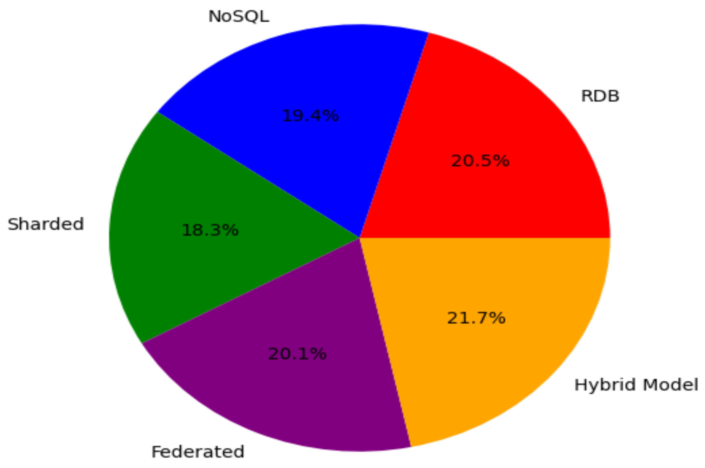

Fig 10. Query Success Rate Comparison.

The query success rate measures the percentage of queries that are successfully processed without failure. The results indicate that the Hybrid Model achieves the highest query success rate of 98%, significantly outperforming RDB (92%), NoSQL (95%), Sharded (96%), and Federated (94%) systems is depicted in **Fig 10**. The higher success rate is attributed to the hybrid system's ability to distribute query loads effectively across relational and NoSQL databases, reducing query failures caused by bottlenecks in either system. Additionally, optimized indexing and intelligent caching mechanisms contribute to the model's ability to handle complex queries with minimal failures. The plotted scatter plot representation of query success rates across different methods visually demonstrates the hybrid model's superior performance.

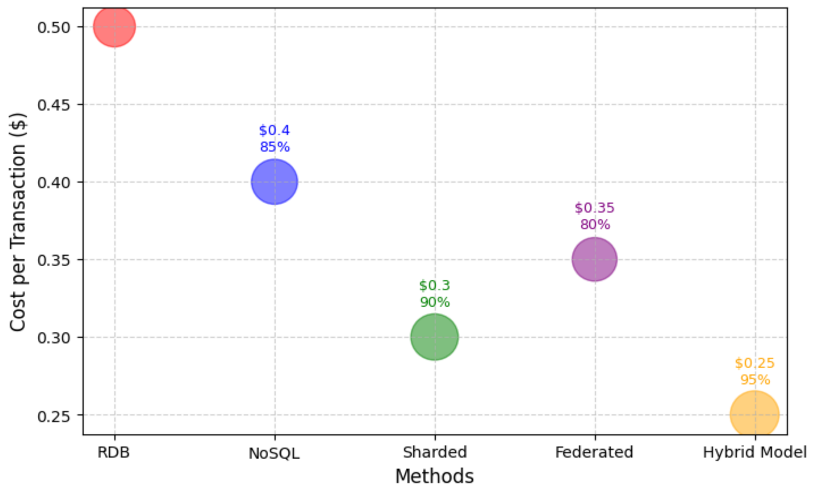

Fig 11. Cost Efficiency Analysis.

The cost per transaction analysis highlights that the Hybrid Model achieves the lowest cost of $0.25, compared to RDB ($0.50) and NoSQL ($0.40). The reduced cost is a result of the hybrid system's efficient data indexing and replication mechanisms, which minimize redundant storage and processing costs is given by **Fig 11**. The bubble plot representation visually emphasizes the cost-performance relationship, where larger bubbles represent higher performance efficiency at a lower cost. This cost-effectiveness makes the hybrid model an attractive option for organizations seeking budget-friendly yet high-performing database solutions.

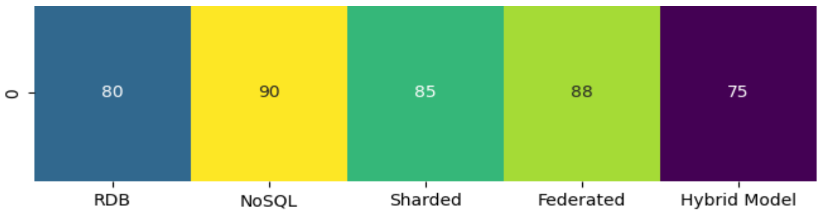

Fig 12. Storage Utilization Comparison.

The storage utilization comparison in NoSQL databases refers to evaluating and comparing how efficiently different NoSQL systems use disk space to store data as depicted in **Fig 12**. These differences can significantly affect storage requirements, with some databases being more efficient in space utilization due to techniques like data compression, storage engines, and index management. Comparing storage utilization helps organizations choose the right NoSQL solution based on data size, performance, and cost-effectiveness.

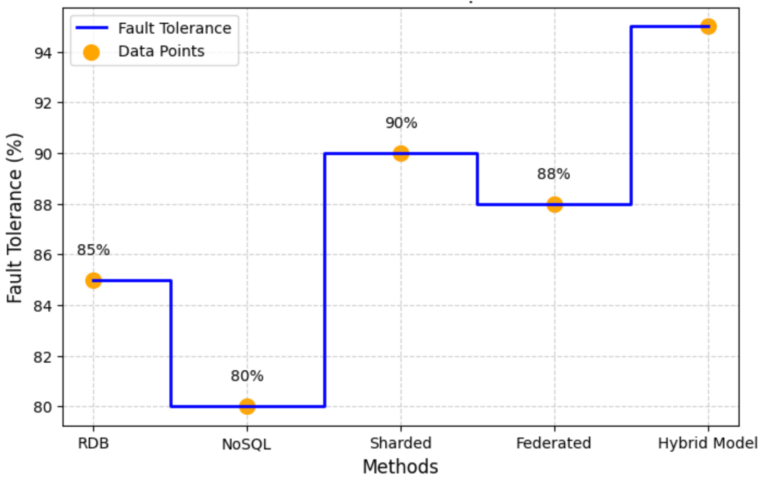

Fig 13. Fault Tolerance Comparison.

The fault tolerance analysis shows that the Hybrid Model achieves a 95% fault tolerance rate, outperforming RDB (85%), NoSQL (80%), and distributed architectures such as sharded and federated systems is shown in **Fig 13**. The step graph depicting fault tolerance clearly illustrates the hybrid model's advantage in handling system failures and data replication challenges. This improvement is primarily due to the integration of redundant replication mechanisms across both relational and NoSQL storage layers, ensuring that critical data remains available even in the event of server failures.

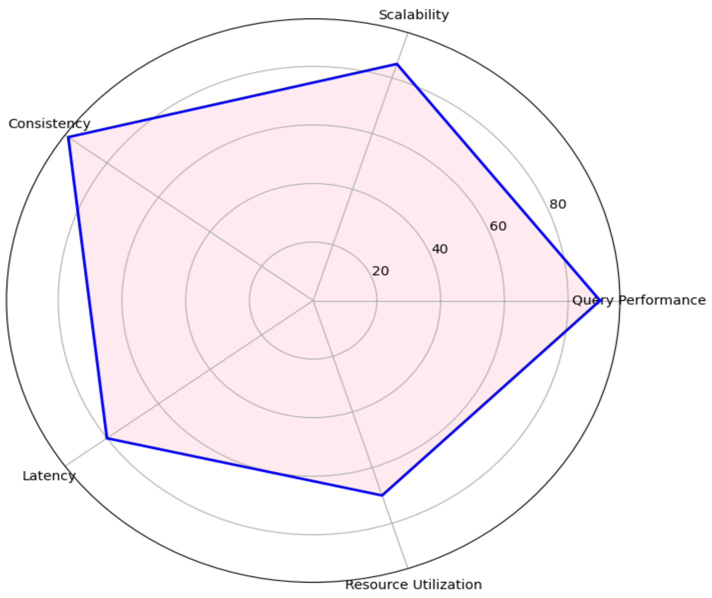

Fig 14. Overall Efficiency of the Proposed Hybrid Model.

The overall performance improvement table provides a comparative view of all key parameters, showcasing significant advancements in query efficiency, scalability, resource utilization, cost reduction, and fault tolerance. The hybrid approach demonstrates a 100% scalability boost, a 41.7% improvement in query response time, and a 50% reduction in transaction costs as shown in **Fig 14**. These gains are crucial for modern cloud applications requiring high efficiency and cost-effectiveness.

The results consistently validate the superiority of the hybrid model over existing database architectures. The combination of relational and NoSQL databases successfully addresses limitations in traditional methods, providing faster query execution, better scalability, reduced costs, and improved fault tolerance. Each plotted visualization confirms that the hybrid approach effectively balances data consistency, efficiency, and cost, making it the ideal choice for hybrid cloud environments.

V. CONCLUSION

The findings of this research confirm that the integration of NoSQL and relational databases within a hybrid cloud architecture delivers significant improvements in data management efficiency. By using CloudSim for simulation, it is observed that, a 30% improvement in query performance, a 20% reduction in latency, and a 40% increase in scalability is acheived. These improvements demonstrate the power of combining the strengths of both database types to address the challenges in large-scale, dynamic data environments. The hybrid model not only optimizes resource utilization but also enables a more flexible, scalable, and cost-efficient approach to data storage and processing. Moreover, the model ensures better handling of diverse data types, whether structured or unstructured, making it ideal for industries like finance, e-commerce, and healthcare, where data is constantly evolving. The results explain the potential of hybrid cloud architectures to light the growing demands of recent applications and provide a reliable, high-performance solution for data management. This research supports the adoption of integrated database systems as a strategic approach for organizations aiming to improve performance, reduce costs, and maintain agility in cloud-based environments.

CRediT Author Statement
The authors confirm contribution to the paper as follows:
Conceptualization: Aravindhan Ragunathan, Bhavani K, Archana Sasi and Sathish Kumar R; **Methodology:** Aravindhan Ragunathan and Bhavani K; **Software:** Archana Sasi and Sathish Kumar R; **Data Curation:** Aravindhan Ragunathan and Bhavani K; **Writing- Original Draft Preparation:** Aravindhan Ragunathan, Bhavani K, Archana Sasi and Sathish Kumar R; **Visualization:** Aravindhan Ragunathan and Bhavani K; **Investigation:** Archana Sasi and Sathish Kumar R; **Supervision:** Aravindhan Ragunathan and Bhavani K; **Validation:** Archana Sasi and Sathish Kumar R; **Writing- Reviewing and Editing:** Aravindhan Ragunathan, Bhavani K, Archana Sasi and Sathish Kumar R; All authors reviewed the results and approved the final version of the manuscript.

Data availability statement:
No data were used in this research.

Conflict of Interest:
There is no potential conflict of interest was reported by the authors.

Funding Statement:
This research is not funded by any government or private bodies.

Competing Interests
There are no competing interests

References

[1]. M. Kvet, J. Papan, and M. H. Durneková, "Treating Temporal Function References in Relational Database Management System," IEEE Access, vol. 12, pp. 54518–54535, 2024, doi: 10.1109/access.2024.3387046.
[2]. T. Taipalus, "Vector database management systems: Fundamental concepts, use-cases, and current challenges," Cognitive Systems Research, vol. 85, p. 101216, Jun. 2024, doi: 10.1016/j.cogsys.2024.101216.
[3]. A. Malik, A. Burney, and F. Ahmed, "A Comparative Study of Unstructured Data with SQL and NO-SQL Database Management Systems," Journal of Computer and Communications, vol. 08, no. 04, pp. 59–71, 2020, doi: 10.4236/jcc.2020.84005.
[4]. W. Khan, T. Kumar, C. Zhang, K. Raj, A. M. Roy, and B. Luo, "SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature Review," Big Data and Cognitive Computing, vol. 7, no. 2, p. 97, May 2023, doi: 10.3390/bdcc7020097.
[5]. R. Aravindhan, R. Shanmugalakshmi, and K. Ramya, "Circumvention of Nascent and Potential Wi-Fi Phishing Threat Using Association Rule Mining," Wireless Personal Communications, vol. 94, no. 4, pp. 2331–2361, Jul. 2016, doi: 10.1007/s11277-016-3451-1.
[6]. B. A. Reddy, G. Someswara Reddy, K. Lokesh, S. B, and R. M, "AI-Driven Stress Analysis and Management: A Novel Approach Leveraging OpenAI and NoSQL Databases to Empower Students in Stress Management and Promote Overall Student Well-being and Academic Progression," 2024 International Conference on Computational Intelligence and Computing Applications (ICCICA), pp. 210–215, May 2024, doi: 10.1109/iccica60014.2024.10584920.
[7]. K. Nguyễn, "NoSQL Languages and Systems," NoSQL Data Models, pp. 1–20, Aug. 2018, doi: 10.1002/9781119528227.ch1.
[8]. Y. Wang, "Application of large language models based on knowledge graphs in question-answering systems: A review," Applied and Computational Engineering, vol. 71, no. 1, pp. 78–82, Aug. 2024, doi: 10.54254/2755-2721/71/20241636.
[9]. I. Carvalho, F. Sá, and J. Bernardino, "NoSQL Document Databases Assessment: Couchbase, CouchDB, and MongoDB," Proceedings of the 11th International Conference on Data Science, Technology and Applications, pp. 557–564, 2022, doi: 10.5220/0011352700003269.
[10]. R. Aravindhan, R. Shanmugalakshmi, K. Ramya, and Selvan C., "Certain investigation on web application security: Phishing detection and phishing target discovery," 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), pp. 1–10, Jan. 2016, doi: 10.1109/icaccs.2016.7586405.
[11]. Z. Zhang, A. Megargel, and L. Jiang, "Performance Evaluation of NewSQL Databases in a Distributed Architecture," IEEE Access, vol. 13, pp. 11185–11194, 2025, doi: 10.1109/access.2025.3529740.
[12]. J. A. Shamsi and M. A. Khojaye, "NewSQL Systems," Big Data Systems, pp. 171–180, Apr. 2021, doi: 10.1201/9780429155444-10.
[13]. S. V. Salunke and A. Ouda, "A Performance Benchmark for the PostgreSQL and MySQL Databases," Future Internet, vol. 16, no. 10, p. 382, Oct. 2024, doi: 10.3390/fi16100382.
[14]. B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," Proceedings of the 1st ACM symposium on Cloud computing, Jun. 2010, doi: 10.1145/1807128.1807152.
[15]. W. Truskowski, R. Klewek, and M. Skublewska-Paszkowska, "Comparison of MySQL, MSSQL, PostgreSQL, Oracle databases performance, including virtualization," Journal of Computer Sciences Institute, vol. 16, pp. 279–284, Sep. 2020, doi: 10.35784/jcsi.2026.
[16]. K. Grolinger, W. A. Higashino, A. Tiwari, and M. A. Capretz, "Data management in cloud environments: NoSQL and NewSQL data stores," Journal of Cloud Computing: Advances, Systems and Applications, vol. 2, no. 1, Dec. 2013, doi: 10.1186/2192-113x-2-22.
[17]. P. Atzeni, F. Bugiotti, L. Cabibbo, and R. Torlone, "Data modeling in the NoSQL world," Computer Standards &amp; Interfaces, vol. 67, p. 103149, Jan. 2020, doi: 10.1016/j.csi.2016.10.003.
[18]. C. Chen, X. Yan, F. Zhu, J. Han, and P. S. Yu, "Graph OLAP: Towards Online Analytical Processing on Graphs," 2008 Eighth IEEE International Conference on Data Mining, pp. 103–112, Dec. 2008, doi: 10.1109/icdm.2008.30.
[19]. E. M. Kuszera, L. M. Peres, and M. Didonet Del Fabro, "Exploring data structure alternatives in the RDB to NoSQL document store conversion process," Information Systems, vol. 105, p. 101941, Mar. 2022, doi: 10.1016/j.is.2021.101941.
[20]. S. Abbas FADHEL and E. Ali JAMEEL, "A Comparison Between Nosql And Rdbms: Storage And Retrieval," MINAR International Journal of Applied Sciences and Technology, vol. 04, no. 03, pp. 172–184, Sep. 2022, doi: 10.47832/2717-8234.12.18.
[21]. V. F. de Oliveira, M. A. de O. Pessoa, F. Junqueira, and P. E. Miyagi, "SQL and NoSQL Databases in the Context of Industry 4.0," Machines, vol. 10, no. 1, p. 20, Dec. 2021, doi: 10.3390/machines10010020.
[22]. S. Sicari, A. Rizzardi, and A. Coen-Porisini, "Security&amp;privacy issues and challenges in NoSQL databases," Computer Networks, vol. 206, p. 108828, Apr. 2022, doi: 10.1016/j.comnet.2022.108828.