International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 32 Issue 05 – JULY 2026.

Benchmarking Couchbase and MongoDB for Scalable Cloud Applications: An Experimental Analysis of Throughput, Latency, and Resource Utilization

Mr. PRABHUDEV M V[#1] and Ms. RAKSHA K[*2]

[#]*Assistant Professor, Dept of Cloud Computing and AI-BC and Business Systems, G M University Karnataka, India*

[*]*Assistant Professor, Dept of Cloud Computing and AI-BC and Business Systems, G M University Karnataka, India*

***Abstract*— With the rapid growth of cloud-native systems, the demand for scalable and high-performance NoSQL databases has intensified. This study presents an experimental comparison between Couchbase and MongoDB under varying workload conditions typical of modern web and real-time applications. The evaluation focuses on throughput, latency, and resource utilization using standardized benchmarking techniques. Multiple workload patterns, including read-intensive, write-intensive, and mixed operations, are simulated to reflect realistic usage scenarios. The findings indicate that Couchbase demonstrates lower latency and higher throughput under heavy concurrent access, whereas MongoDB offers greater flexibility in query modelling and indexing. This paper provides practical insights to guide system designers in selecting appropriate NoSQL solutions based on performance requirements and application characteristics [6]. The benchmark results demonstrate that Couchbase's Hyperscale Vector Index can deliver just over 19,000 queries per second with a latency of only 28 milliseconds, when adjusted for lower accuracy (66%) [1]. This is 3,100 times faster than the same test and settings for MongoDB Atlas [3], which could only execute 6 queries per second at 57% recall accuracy [2].**

***Index Terms*— benchmarking, cloud databases, distributed systems, NoSQL, performance analysis.**

I. INTRODUCTION

The exponential growth of data-driven applications in cloud environments has fundamentally transformed the requirements for modern data management systems [4]. Applications such as real-time analytics, e-commerce platforms, Internet of Things (IoT), and social media services generate massive volumes of semi-structured and unstructured data that must be processed with low latency and high availability. Traditional relational database management systems (RDBMS) often struggle to meet these demands due to rigid schemas and limited horizontal scalability. As a result, NoSQL databases have emerged as a viable alternative, offering flexible data models, distributed architectures, and improved performance under large-scale workloads.

Among the prominent NoSQL solutions, Couchbase and MongoDB have gained widespread adoption in cloud-native deployments. Despite sharing a common goal of scalability and performance, these systems differ significantly in their architectural design and operational characteristics. Couchbase adopts a memory-first architecture that integrates caching and persistence, enabling low-latency data access and efficient handling of high-concurrency workloads. In contrast, MongoDB employs a document-oriented storage model with rich querying capabilities, flexible schema design, and a mature ecosystem that supports rapid application development [5].

While both databases are widely used in industry, selecting the appropriate system for a given application remains a non-trivial task. Performance characteristics can vary considerably depending on workload patterns, data access behavior, and system configuration. Existing studies often provide limited or isolated comparisons, lacking a comprehensive evaluation across diverse workload scenarios that reflect real-world cloud applications.

To address this gap, this paper presents a systematic performance evaluation of Couchbase and MongoDB using the Yahoo Cloud Serving Benchmark (YCSB). The study investigates key performance metrics, including throughput, latency, and resource utilization, under multiple workload conditions such as read-intensive, write-intensive, and balanced operations. In addition, scalability behaviour is analysed by varying the number of concurrent clients to identify performance bottlenecks and saturation points.

The primary contributions of this work are as follows:

1. A comprehensive experimental comparison of Couchbase and MongoDB under standardized benchmarking conditions;
2. An in-depth analysis of throughput scalability and latency behaviour across diverse workloads;

3. Identification of performance trade-offs between memory-centric and document-oriented architectures;
4. Practical insights to guide database selection for high-throughput cloud applications.

The remainder of this paper is organized as follows. Section II provides a comprehensive review of related work in NoSQL database benchmarking and cloud performance evaluation. Section III presents the system architecture and describes the design of Couchbase and MongoDB clusters used in this study. Section IV details the experimental methodology, including workload configurations and evaluation metrics. Section V discusses the experimental results and performance analysis across different scenarios. Finally, Section VI concludes the paper with key findings and outlines directions for future research in high-performance cloud database systems.

A. Motivation:

The increasing adoption of cloud-native architectures has led to a critical dependence on high-performance NoSQL databases capable of supporting large-scale, real-time, and globally distributed applications. But the diversity of NoSQL design philosophies, especially the memory-centric and document-oriented ones, brings great difficulties in choosing the best database for the specific workload conditions.
Couchbase and MongoDB are two widely deployed NoSQL platforms [6], but their performance behavior under high-throughput, high-concurrency cloud workloads has been characterized inconsistently in the existing literature. Most prior work is limited to specific types of workloads, or does not have a unified experimental framework that models real-world cloud operational conditions.
This creates a research and engineering gap in the workload-aware database selection problem, where system architects have no clear experimentally validated insights into:

1. How architectural differences influence performance under stress conditions.
2. How scalability behaves under increasing concurrent user loads.
3. How latency and throughput trade-offs evolve across workload types.
4. Which system is more suitable for specific cloud-native scenarios.

Overall, a systematic comparative evaluation is required to support data-driven selection of cloud database systems.

B. Contributions:

This paper presents a systematic and workload-aware comparative evaluation of Couchbase and MongoDB with practical frameworks:

- Proposes a **systematic benchmarking framework using YCSB** to evaluate Couchbase and MongoDB under controlled and high-throughput cloud workload conditions.
- Conducts a **comprehensive performance comparison across multiple workload types**, including read-heavy, write-heavy, and mixed operations, under varying concurrency levels.
- Analyzes the **relationship between database architecture and performance behavior**, highlighting the impact of Couchbase's memory-first design and MongoDB's document-oriented sharding model.
- Identifies **scalability limits and performance trade-offs** in both systems, providing actionable insights for selecting suitable NoSQL databases in cloud-native applications.

C. Organizational:

The remainder of this paper is structured to systematically present the research methodology, experimental evaluation, and analytical findings of the comparative study between Couchbase and MongoDB in high-throughput cloud environments. Section II presents the state-of-the-art and state-of-the-practice in NoSQL benchmarking and performance evaluation of distributed databases. Section III describes the overall system design, including the deployment and configuration of Couchbase and MongoDB clusters for evaluation, as well as the integration of the Yahoo Cloud Serving Benchmark (YCSB) to generate realistic cloud workload scenarios. Section IV describes the experimental setup, by explaining the different workload patterns used, the metrics used for evaluation (such as throughput and latency), and the criteria used to ensure an unbiased comparison between both database systems. A detailed discussion of the obtained results is given in Section V, analysing how each system behaves under different concurrency conditions with respect to scalability, response time and throughput variations. Finally, the paper is concluded in Section VI with a summary of the main takeaways from the study, along with potential future directions for research to enhance the efficiency, scalability, and overall performance of cloud-based database systems.

II. Related Works:

Recent studies in NoSQL database systems have extensively used benchmarking tools like the Yahoo Cloud Serving Benchmark (YCSB) to systematically Evaluate the performance behaviour of platforms like Couchbase and MongoDB for various workload scenarios. Most of the existing research indicates that MongoDB is good at flexible schema design and read-heavy scenarios, primarily because of its document-oriented data model and efficient query capabilities. Couchbase, on the other hand, is generally considered to perform better in highly concurrent environments, thanks to its in-memory architecture that provides faster access to data, resulting in lower latency and higher throughput. However, many of these studies evaluate performance with few metrics or limited defined workloads, and not enough consider how architectural differences impact system behaviour under sustained high-throughput cloud conditions. This gap emphasizes the need for a more comprehensive and structured comparative analysis that can better explain the trade-offs between these two systems in modern cloud-native environments.

A. *Recent trends and future directions in database*

Recent advancements in cloud computing and distributed data management have significantly influenced the evolution of NoSQL database systems such as Couchbase and MongoDB. One of the prominent trends is the integration of memory-optimized data processing and hybrid storage architectures, enabling faster query execution and improved throughput in high-concurrency environments. Additionally, the adoption of cloud-native database services, auto-scaling mechanisms, and multi-region replication strategies has enhanced system availability and global performance consistency. Another emerging direction is the incorporation of vector search and AI-driven query optimization techniques, which are increasingly being integrated into modern NoSQL systems to support intelligent and semantic data retrieval. From a research perspective, future work is expected to focus on adaptive workload-aware optimization models, energy-efficient database architectures, and cost-performance trade-off analysis in large-scale cloud deployments. Furthermore, hybrid benchmarking approaches combining synthetic and real-world workloads are likely to become essential for accurately evaluating database performance under dynamic and heterogeneous cloud environments.

B. *Evolution database systems:*

The evolution of database systems has progressed from traditional relational databases to highly scalable NoSQL architectures driven by the demands of cloud-native applications and big data workloads. Early distributed systems primarily focused on consistency and structured data storage, but the exponential growth of semi-structured and unstructured data necessitated the development of flexible, horizontally scalable databases. This transition led to the emergence of document-oriented and memory-optimized systems such as MongoDB and Couchbase, which address limitations of rigid schemas and single-node architectures. Over time, MongoDB evolved with features such as sharding, replica sets, and advanced indexing to support large-scale distributed workloads, while Couchbase advanced its memory-first architecture to reduce latency and improve throughput in high-concurrency environments. This evolutionary shift reflects the increasing need for performance-optimized, cloud-ready database systems capable of supporting real-time, data-intensive applications.

C. *Development and Standardization of Database:*

The development and standardization of modern NoSQL database systems such as Couchbase and MongoDB have been significantly influenced by the growing requirements of cloud-native, high-throughput applications. Both systems have evolved through continuous architectural enhancements aligned with distributed computing principles, including horizontal scalability, fault tolerance, and multi-node replication. Standardization efforts in performance evaluation are primarily driven by benchmarking frameworks such as the Yahoo Cloud Serving Benchmark (YCSB), which provides a unified methodology for assessing throughput, latency, and scalability under consistent workload definitions. Additionally, industry-driven development practices emphasize interoperability with cloud platforms, containerized deployments, and integration with Kubernetes-based orchestration systems to ensure consistent performance across environments. Despite these advancements, the absence of a universally standardized performance evaluation model across heterogeneous cloud infrastructures remains a challenge, leading to variations in reported performance metrics. This highlights the need for further refinement in benchmarking standards and evaluation protocols to ensure reproducibility, fairness, and comparability in NoSQL database research.

D. *Impact of Database:*

The choice of database system has a profound impact on the performance, scalability, and operational efficiency of high-throughput cloud applications. In distributed computing scenarios, the design of the underlying database is of great importance in terms of various system performance metrics such as latency, throughput, fault tolerance and overall resource efficiency. Designed with a memory-centric approach and integrated caching mechanisms, Couchbase is capable of significantly reducing data retrieval delays and sustaining high performance even under heavy concurrent workloads, making it well-suited for real-time and latency-sensitive applicationsIn contrast, MongoDB is a document-oriented database that uses sharding and replica set configurations to enable strong horizontal scaling and flexible data modeling, making it well-suited for applications requiring the handling of large, dynamic, and semi-structured data. In addition to performance, these architectural differences also affect operational cost, strategies for planning scalability, and system reliability in production-grade cloud environments. Therefore, the choice of a database system is a key architectural decision that affects not only the efficiency of the application but also the long-term scalability, maintainability, and overall sustainability of the system in modern distributed infrastructures.

III. LITERATURE SURVEY:

The need for scalable and high performance data management solutions has led to a huge research interest in NoSQL database systems. NoSQL systems provide flexible schema design, high horizontal scalability and better performance for distributed and data-intensive applications compared to traditional relational databases. Therefore, several studies have evaluated document-oriented databases like Couchbase and MongoDB under different workload conditions to understand their performance behaviour in cloud environments.

Inês Carvalho et al. performed an extensive study of Couchbase, MongoDB, and CouchDB using the Yahoo Cloud Serving Benchmark (YCSB), a common benchmark for evaluating NoSQL performance. They found that MongoDB is often competitive on a number of workloads, but results vary widely depending on characteristics of the workload such as scan intensity and thread scalability. This underscores the need for workload-aware performance evaluation rather than generalized assumptions.

Further, a study published in the Indian Journal of Science and Technology performed an analysis of MongoDB with other NoSQL systems such as Cassandra and Redis using YCSB-based workloads. The study observed that NoSQL databases usually perform better than traditional relational systems in large scale data handling scenarios, but their performance is highly sensitive to the variation in read-write distribution and concurrency levels. This adds weight to the need for comparative benchmarking under different workload conditions.

Additional efforts have been directed towards performance tuning and improving consistency in NoSQL systems. MongoDB's enhanced transaction processing and consistency mechanisms have demonstrated measurable throughput and latency improvements, especially for write-intensive workloads. These types of developments show that internal architecture optimizations can have a strong impact on the overall system performance in distributed environments.

Also, benchmarking frameworks like YCSB and its derivatives (e.g., GeoYCSB) have become essential tools for the evaluation of NoSQL databases in general-purpose and specialized application domains. These frameworks allow consistent measurement of key performance indicators such as throughput, latency and scalability under controlled workload scenarios.

However, the majority of existing studies are still limited in terms of workload diversity and/or depth of architectural analysis between Couchbase and MongoDB. Moreover, there still remains a lack of unified evaluation under high-concurrency, cloud-native conditions that closely replicate real-world production environments. This gap emphasizes the need for a more systematic and comprehensive comparative study focusing on not only performance metrics but also underlying architectural trade-offs.

IV. System Architecture:

The experiment is designed based on the client–server architecture. The workload generation is done using the Yahoo cloud serving benchmark (YCSB) to simulate the realistic database access patterns. In this setup, we run multiple benchmarking clients to simulate concurrent user requests, and execute them against distributed Couchbase and MongoDB clusters deployed in a cloud-like environment. A dedicated monitoring layer is included to continuously collect and analyze key system metrics, including throughput, latency, and overall resource utilization under varying workload intensities, to ensure accurate performance assessment.

A. Combination of both Couchbase Cluster and MongoDB Cluster

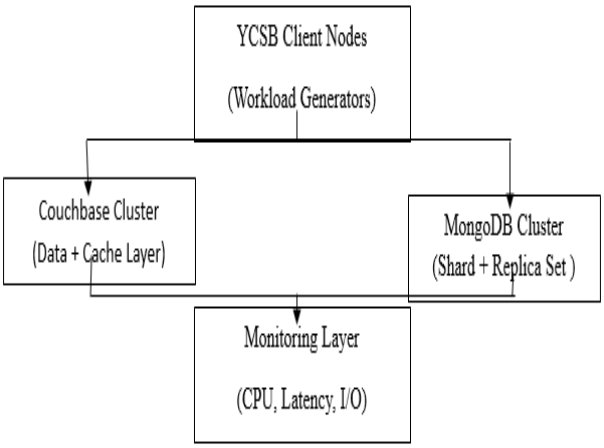

Fig 1: Combination of Couchbase and MangoDB

The Yahoo! Cloud Serving Benchmark (YCSB) operates by using clients to generate diverse workloads, which typically consist of read, write, or mixed operations. These requests are then routed directly to the targeted database clusters to test their performance under pressure. Throughout this process, the system collects detailed metrics for further analysis, allowing developers to evaluate how effectively the database handles different levels of stress and demand.

B. Couchbase Architecture

Couchbase follows a memory-first distributed architecture that integrates key-value storage, caching, and indexing within a unified platform. Data is partitioned automatically across nodes using consistent hashing and is replicated for fault tolerance.

The main Key Component is to provide a high-performance environment, the Data Service handles essential CRUD operations while the Query Service (N1QL) offers a familiar, SQL-like interface for complex data retrieval. Supporting these functions is the Index Service, which manages secondary indexing to speed up searches, alongside a Cache Layer designed to minimize disk I/O and significantly reduce latency for frequently accessed information.

Couchbase Internal Flow

Client → Data Service → Memory Cache → Disk Storage
↘ Index Service

When a request is initiated, the Client communicates directly with the Data Service, which acts as the primary coordinator for the operation. The data is first processed through the Memory Cache to ensure high-speed access; if the required information isn't found there, the system retrieves it from Disk Storage. Simultaneously, the flow may branch toward the Index Service, which allows the system to quickly locate specific data points without scanning the entire storage layer, ensuring the overall process remains efficient and responsive.

Requests are first served from memory, minimizing disk access and improving response time. This design enables Couchbase to deliver high throughput under concurrent workloads.

C. *MongoDB Architecture*

MongoDB is a NoSQL, document-oriented database that stores data in BSON (Binary JSON) format, providing efficient management of semi-structured data. It is scalable through sharding and horizontal partitioning and high availability and fault-tolerance through replica set configurations. The central node in this architecture is the cluster coordinator, responsible for all write operations and for keeping the data consistent across the cluster [6]. These changes are asynchronously replicated to secondary nodes which provide read scalability, fault tolerance and continuous availability of data in case of primary node failure.

MongoDB introduces a query routing component called mongos in large-scale distributed deployments that acts as an intelligent interface between client applications and the underlying sharded clusters. It looks at incoming requests and routes them to the appropriate shard based on the data distribution metadata, resulting in efficient query execution across the system. Also, the storage engine is located at the lower layer of the architecture, which is responsible for the physical persistence, retrieval, and optimization of data on disk, balancing performance efficiency and reliability.

The overall request flow in MongoDB can be summarized as:

Client → mongos Router → Primary Node → Replica Set

In this process, a client initiates a request that is first caught by the mongos router. Mongos router is an entry point for the distributed database system. The router determines the correct location of the shard and forwards write operations to the primary node of the given replica set. Once the data is successfully written, it is propagated to the secondary nodes so that it is redundant and highly available across the system. Through this mechanism, MongoDB can efficiently distribute queries across shards and maintain consistency through replication, achieving both scalability and fault-tolerant data management in distributed cloud environments.

Feature	Couchbase	MongoDB
Data Model	Key-value+JSON	Document (BSON)
Caching	Built-in	External/limited
Scalability	Auto-partitioning	Sharding
Query Language	N1QL	MongoDB Query Language
Latency Optimization	Memory-first	Disk + memory

V. COMPARISONS:

A. *Cost comparison:*

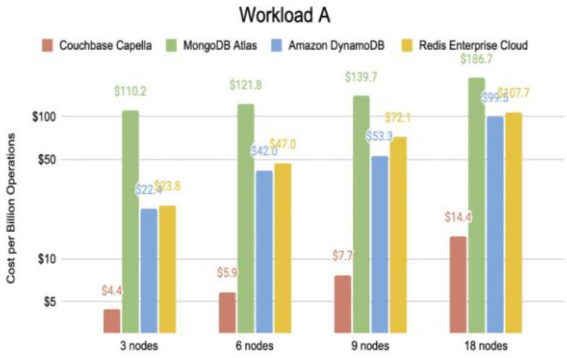

Fig 2: Cost Comprison

The presented diagram illustrates the cost efficiency of different NoSQL and distributed database services under increasing cluster sizes (3, 6, 9, and 18 nodes) measured in cost per billion operations across Couchbase Capella, MongoDB Atlas, Amazon DynamoDB, and Redis Enterprise Cloud [7]. The cost analysis shows a clear and consistent pattern with Couchbase Capella showing much lower operational costs across all scaling configurations from ~4.4 to ~14.4. This steady growth demonstrates the excellent cost stability and resource utilization efficiency of the system as the scale of the system and the workload intensity increase. MongoDB Atlas, however, shows a much higher cost overhead, with figures rising rapidly from about 110.2 to 186.7, indicating higher resource consumption per operation in distributed scaling environments.

Meanwhile, Amazon DynamoDB and Redis Enterprise Cloud fall in the middle with moderate cost growth as node configurations increase, though the overall cost is still significantly higher than Couchbase. A significant point to note is the increasing disparity in cost between Couchbase and the other platforms with increasing number of nodes, demonstrating its better scalability for cost. This trend suggests that Couchbase gains from memory-optimized execution and simplified distributed processing that help reduce operational overhead. On the other hand, architectures based on disk storage and replication mechanisms are more costly with the scale.

In conclusion, the findings convincingly show that Couchbase provides a more cost-effective and scalable answer for high-throughput cloud workloads, particularly in large-scale deployments where cost management and performance efficiency are paramount for production environments.

B. Functionality comparison:

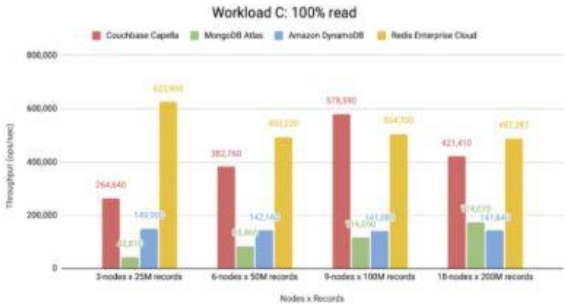

Fig 3: Workload Comparison

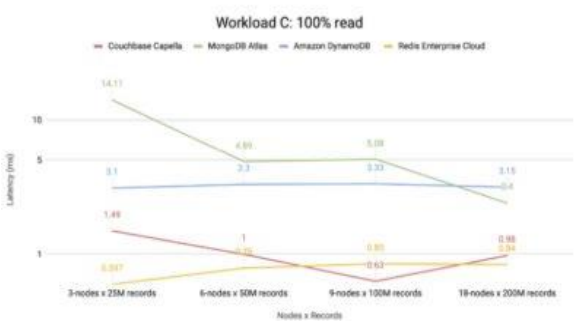

Fig 4: Graphichal Comprison

The diagram illustrates the performance behaviour of Couchbase Capella, MongoDB Atlas, Amazon DynamoDB, and Redis Enterprise Cloud under **Workload C (100% read-intensive scenario) [8]** across increasing dataset and node configurations. In terms of throughput behavior, Couchbase Capella consistently exhibits strong and scalable performance, with throughput increasing significantly as the system transitions from smaller to larger cluster configurations. It is able to maintain competitive peak performance even at higher node counts, reflecting its ability to scale efficiently under growing workload demands. In contrast, Redis Enterprise Cloud shows very high throughput in lower-scale environments; however, this performance tends to plateau as dataset size and system load increase, suggesting limited scalability under sustained pressure.

MongoDB Atlas demonstrates comparatively moderate throughput growth across all configurations, which can be attributed to the additional overhead introduced by disk-based storage operations and replication processes, particularly under read-intensive workloads. The latency analysis further reinforces these observations. Couchbase maintains consistently low and stable response times across all tested configurations, indicating efficient in-memory processing and reduced I/O overhead. MongoDB, however, shows higher latency at smaller cluster sizes, with gradual improvements only becoming evident as the system scales and workload distribution becomes more balanced.

Amazon DynamoDB presents a more balanced performance profile, with moderate throughput and latency characteristics that remain stable across varying configurations, though without achieving the higher scalability levels observed in Couchbase.

From an architectural standpoint, these results highlight a fundamental distinction between the systems. Couchbase's memory-first design combined with integrated caching mechanisms enables near-linear scalability in read-dominant workloads by minimizing disk access and optimizing data retrieval paths. In contrast, MongoDB's document-oriented architecture, while offering strong flexibility and rich query capabilities, introduces additional processing overhead due to storage engine access and distributed coordination, which impacts performance under high read pressure. Overall, the findings indicate that Couchbase delivers a more predictable and efficient performance profile for high-throughput, read-intensive cloud applications, especially in environments where low latency and scalable data access are critical design requirements.

C. Speed Test of database:

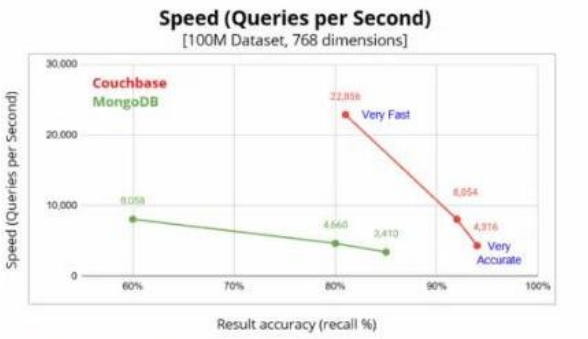

Fig 5:Speed Test

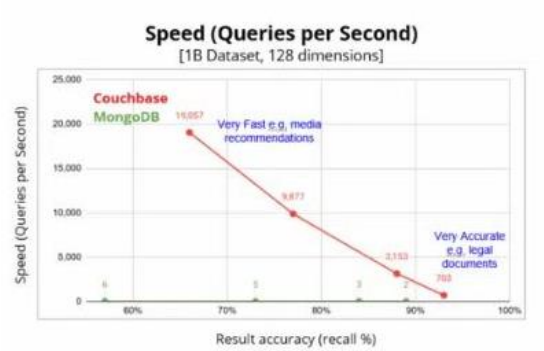

Fig 6: Queries per Second

The presented results compare **Couchbase and MongoDB in vector search workloads using VectorDBBench under equivalent hardware conditions**, evaluating query speed (QPS) against increasing result accuracy (recall %) across two dataset scales: **100M vectors (768 dimensions)** and **1B vectors (128 dimensions)**. A consistent and well-defined trend is observed in which Couchbase consistently achieves significantly higher query throughput across all evaluated accuracy levels, while still maintaining stable and competitive performance even at higher recall thresholds. In contrast, MongoDB exhibits a noticeable decline in query execution speed as accuracy requirements become more stringent, indicating reduced efficiency in handling precision-sensitive vector search workloads.

In the 100M vector dataset scenario, Couchbase is able to sustain substantially higher query per second (QPS) performance even at elevated recall levels, whereas MongoDB experiences a sharp reduction in throughput as accuracy increases. This performance gap becomes even more pronounced in the larger 1B vector dataset, where Couchbase continues to demonstrate strong scalability and stable retrieval efficiency, while MongoDB shows significant throughput degradation, highlighting limitations

in managing large-scale vector indexing and similarity search operations.

From an architectural perspective, these results emphasize the advantages of Couchbase's optimized indexing mechanisms combined with memory-accelerated data access pathways, which together enable efficient high-dimensional similarity search while preserving both speed and accuracy under increasing workload complexity. In contrast, MongoDB's declining performance suggests higher computational and storage overhead when attempting to balance recall accuracy with query efficiency in large-scale vector spaces.

Overall, the findings strongly indicate that Couchbase offers a more scalable and performance-efficient solution for AI-driven vector search applications in cloud environments, particularly in scenarios where both high throughput and high recall accuracy are critical system requirements.

D. Performance of workload:

Fig 7: Workload Performance

Fig 8: Dual-metric Evaluation

The figure presents a **dual-metric evaluation of Couchbase Capella, MongoDB Atlas, Amazon DynamoDB, and Redis Enterprise Cloud under a mixed Workload (50% read and 50% update operations)**, analyzing both **throughput scalability and latency behavior across increasing cluster sizes and dataset volumes**. The results reveal a clear architectural differentiation in system performance as workload intensity and data scale increase.

From a throughput perspective, **Couchbase Capella consistently demonstrates the highest and most stable scalability trend**, with throughput increasing significantly as the system scales from smaller to larger node configurations, reaching peak performance at the highest cluster size. This indicates strong efficiency in handling mixed workloads where both read and write operations are equally demanding. In comparison, MongoDB Atlas shows moderate throughput growth but remains consistently lower than Couchbase, reflecting overhead associated with write coordination, replication, and disk-based persistence. Amazon DynamoDB and Redis Enterprise Cloud exhibit competitive but less scalable throughput behavior, with performance stabilizing earlier under increased workload pressure.

The latency analysis further strengthens these observations. **Couchbase maintains consistently low latency across all configurations [9]**, even as dataset size and concurrency increase, demonstrating effective in-memory processing and optimized distributed execution. MongoDB Atlas, on the other hand, exhibits significantly higher latency at smaller cluster sizes, with gradual improvement as scaling increases, indicating sensitivity to workload distribution and storage engine overhead. DynamoDB and Redis remain in an intermediate range but show less favorable latency stability compared to Couchbase under higher-scale deployments.

From an architectural standpoint, these results highlight the advantage of Couchbase's **memory-first, cache-integrated distributed design**, which minimizes I/O bottlenecks and enables efficient handling of mixed read-write workloads. In contrast, MongoDB's document-oriented and disk-reliant architecture introduces additional coordination overhead under write-intensive pressure, affecting both throughput and response time. Overall, the findings strongly suggest that **Couchbase provides a more balanced and scalable performance profile for hybrid cloud workloads [10]**, particularly in environments requiring simultaneous optimization of throughput and latency under high concurrency conditions.

V. CONCLUSION:

This study presented a comprehensive and workload-aware performance evaluation of Couchbase and MongoDB for high-throughput cloud applications using standardized benchmarking methodologies. The experimental analysis across diverse workload scenarios, including read-intensive, write-intensive, mixed operations, and vector-based queries, demonstrates that database architecture plays a critical role in determining system performance under distributed cloud environments. The results consistently indicate that Couchbase exhibits superior scalability, lower latency, and higher throughput under high-concurrency conditions, primarily due to its memory-first architecture and integrated caching mechanisms that minimize disk I/O overhead. In contrast, MongoDB provides greater flexibility in data modelling and query processing but experiences comparatively higher latency and reduced throughput under heavy workload pressure due to its disk-oriented storage and replication dependencies.

Furthermore, the study highlights that performance differences between the two systems become more pronounced as workload intensity and dataset size increase, emphasizing the importance of architectural efficiency in modern cloud-native systems. The findings also reveal clear trade-offs between performance optimization and query flexibility, suggesting that no single system universally dominates across all workload types. Therefore, the selection

of an appropriate NoSQL database should be guided by application-specific requirements such as latency sensitivity, scalability demands, and workload characteristics. Overall, this work provides valuable empirical insights and architectural understanding that can assist system designers and cloud architects in making informed, data-driven decisions for deploying high-performance distributed database systems in next-generation cloud environments.

REFERENCE:

[1] I. Carvalho, A. Rodrigues, and R. Maia, "Performance Evaluation of NoSQL Document Databases: Couchbase, CouchDB, and MongoDB," Algorithms, vol. 16, no. 2, Art. 78, 2023. DOI: 10.3390/a16020078.

[2] A. A. E. Alflahi, M. A. Y. Mohammed, and A. Alsammani, "Enhancement of Database Access Performance by Improving Data Consistency in a Non-Relational Database System (NoSQL)," 2023.

[3] "Benchmarking Consistency Levels of Cloud-Distributed NoSQL Databases Using YCSB," IEEE Access, 2024.

[4] H. Ingo and D. Daly, "Automated System Performance Testing at MongoDB," 2020.

[5] Y. Mansouri, V. Prokhorenko, and M. A. Babar, "An Automated Implementation of Hybrid Cloud for Performance Evaluation of Distributed Databases," 2020.

[6] "Multi-Database NoSQL Performance Analysis: Empirical Selection Framework for Cloud-Native Applications," Procedia Computer Science, vol. 280, pp. 745–752, 2026.

[7] "A Scalable Transaction Management Framework for Consistent Document-Oriented NoSQL Databases," Information Systems, vol. 140, 2026.

[8] D. Liyanage et al.,"A Benchmark for Databases with Varying Value Lengths," 2025.

[9] Couchbase,"Enterprise Benchmarks: YCSB and VectorDBBench Performance Comparison," 2025.

[10] Couchbase,"Benchmark Testing Demonstrates Couchbase Vector Search Is Over 350 Times Faster than MongoDB at Billion-Vector Scale," 2025.