Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

Data Storage and Speed: Why Some Businesses Use Both MongoDB and Aerospike

Mukesh Reddy Dhanagari
Manager, Software Development & Engineering, Charles Schwab, USA
ORCID: 0009-0005-7281-2192
dhanagari.mukeshreddy@gmail.com

ARTICLE INFO

Received: 01 Dec 2024

Revised: 01 Jan 2025

Accepted: 08 Jan 2025

ABSTRACT

This paper presents a research study that seeks to understand the performance optimization that can be established through the co-deployment of MongoDB and Aerospike databases in contemporary applications and their relative advantages concerning supporting different types of workloads. The document-based NoSQL database is MongoDB, which excels in schema flexibility, extensive document manipulation, and complex queries, making it ideal for dynamic setups like user profiles or product inventory examples. Aerospike is a low-latency, high-performance key-value store, tailored toward low-latency, high-throughput workloads, and specifically to use cases such as session management and real-time data feeds. Operating Aerospike and MongoDB concurrently enables an organization to leverage the support of both databases to query and make high-volume transactions whenever a workload comes along. The study investigates the realization of such high-performance architecture through the aggregation of the said systems and enlists the benefits and compromises of this dual-stacking technique. Some significant areas of future research are on understanding the performance of various configurations, making latency and storage costs controllable, and the predictive modelling of workload behavior using machine learning-based techniques such as correlation-based feature selection (CFS) and principal component analysis (PCA). Also, the paper points out a lack of recent empirical knowledge about dual-stack deployment. It presents future work opportunities, including the use of larger datasets, comparisons of MongoDB and Aerospike with additional NoSQL systems, and performance tuning automation via online learning techniques. Results indicate that co-deployment of MongoDB and Aerospike is optimal when there is a need to operate flexibly querying systems that must achieve reliable, low-latency behavior, resulting in a developer-friendly system that is operationally scalable.

Keywords: Dual-stack architecture, Tail latency, NoSQL databases, High-cardinality indexes, multi-cloud deployment

1. Introduction

The quick growth of the world digital market has contributed to the development of an appetite for systems that could sustain the huge and erratic workloads. Among the most urgent concerns here is the possibility of keeping a balance between the flexibility of a schema and the necessity of ad-hoc queries in terms of predictable latency in the 95th (p95) and 99th (p99) percentiles. These difficulties are even aggravated in the realization of spiky traffic patterns that are manifested in applications that are global and of high demand. There is also a dilemma most businesses go through with regards to a database solution that can scale quickly to adapt to fluctuating transaction volumes, but does not degrade in performance when traffic is high. This conflict between the demands of flexibility of schema

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

and the needs of predictable, low-latency performance under load makes the selection of a database architecture fundamental to the success of modern data-driven enterprises. To satisfy these different needs, organizations are using a dual stack architecture, running two databases in parallel, utilizing the strengths of both individually. MongoDB, a Document-based NoSQL database, is a good fit when flexibility is needed in the schema design, when dealing with rich documents, and when large aggregations are necessary and frequent updates are required. This makes it excellent for dynamic loads like user profiles, catalogs, and product descriptions. Conversely, Aerospike is a key-value store that is optimized to access data very quickly, especially in memory and in flash storage.

Aerospike shines in low-latency operations, making it a compelling solution for workloads where response time is of paramount importance, like session management, counters, and real-time data feeds. The mixed use of MongoDB to supply capabilities in complex queries and flexible schema, and Aerospike with its optimized performance for high-speed, low-latency tasks, allows businesses to form a high-performance system that will adjust to various necessities of their workload. The main aim of the research is to determine the advantages and trade-offs of co-deploying MongoDB and Aerospike to optimize databases in different use cases. The main questions that drive the research include the following: What attributes of workload result in co-deployment yielding a net benefit? Knowing the kinds of workloads that gain the most out of their integration with MongoDB and Aerospike will assist companies in making the right choice in terms of the deployment strategy. What configuration lever prevails in p95 latency and cost per GB? This question aims to determine the key elements of latency and storage cost control and offer real-life guidelines on how to optimize the deployment of databases. Finally, what is the influence of correlation-based feature selection (CFS) and principal component analysis (PCA) on prediction fidelity? In this question, the role of advanced techniques of data analysis in enhancing the accuracy and reliability of performance prediction in both databases will be investigated. The paper also adds an in-depth discussion of the workloads that benefit from the co-deployment of MongoDB and Aerospike, and the predictive models based on which latency, throughput, and storage requirements are estimated. It also brings a decision framework, which overlays the more frequently used use cases, including sessions, counters, catalogs, targeting, and ingestion, to the particular deployment patterns. These insights also present a complete guideline on how organizations can pursue the merger of these two databases to meet their performance and flexibility demands.

The multi-article will be organized as follows: Section 2 will give a literature review, which presents an overview of the NoSQL world and the internals of MongoDB and Aerospike. Section 3 describes the experimentation steps that were applied, such as preprocessing the data after its collection, selecting the relevant features, and exploration. The fourth section displays the modeling and prediction framework. Section 5 talks about the experiments and results, whereas Section 6 provides a discussion of the findings and their implications in practice. Section 7 states possible ways of future work, and Section 8 concludes with a selection of tips gleaned.

2. Literature Review

2.1. NoSQL Context: Document vs. Key-Value vs. Wide-Column; CAP, PACELC, and Practical Consistency Models

NoSQL databases are designed to bring flexibility and elasticity in manipulating massive and unstructured data. Each of the three main variations of NoSQL databases, document, key-value, and wide-column, has its applications, as well as its benefits. Document stores (MongoDB) are those that can process more complex, hierarchical data and are schema-flexible. They enable developers to put semi-structured data in a way that could be changed over time, offering agility to applications whose data models may often have to change. Key-value stores, Aerospike, are optimized to support high-speed access of a value by use of a distinct key. Use cases where such a design is appropriate include session handling and caching, where it is important to retrieve data as quickly as possible. A wide-

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

column store, e.g., Cassandra, stores data in columns instead of rows and can therefore be used in high-throughput analytical workloads that require querying large volumes of data. Concerning distributed database systems, the CAP theorem, which states that a system can never ensure more than two of the three properties of Consistency, Availability, and Partition Tolerance, constitutes a fundamental principle (1). An example is MongoDB, one of the implementations that emphasizes availability and partition tolerance and allows eventually consistent semantics in some settings. Aerospike, on the other hand, keeps the main index in memory and therefore provides good consistency to critical applications that require instant consistency among distributed nodes.

The PACELC theorem generalizes CAP, with a trade-off between latency and consistency in non-partitioned states, adding more precision in system design choices (31). Both MongoDB and Aerospike trade-offs are implemented differently: MongoDB offers a higher degree of flexibility in its consistency model to fit the needs of web applications where high availability is a crucial-factor as it can bring gain in performance with much traffic whereas the strong consistency mode in Aerospike is usually selected on the condition when the low-latency and real-time performance is crucial, e.g., financial transactions. One of the important considerations in choosing the correct database to adopt in an application is the practical consistency models. Such systems offer tunable consistency (such as those supporting MongoDB), with applications having the choice of eventual or strong consistency. In contrast, Aerospike can offer strong consistency in specific system settings (its design is targeted at demanding real-time systems that cannot tolerate data inconsistencies).

As shown in Table 1, the comparison of NoSQL database types highlights key features and use cases for different database structures. Document databases like MongoDB offer schema flexibility and complex query capabilities, making them ideal for applications such as user profiles and product catalogs.

Table 1: ***Comparison of NoSQL Database Types***

Database Type	Example	Key Feature	Use Case
Document	MongoDB	Schema flexibility, complex queries, and large aggregations	User profiles, catalogs, and product descriptions
Key-Value	Aerospike	High-speed access, low-latency performance	Session management, counters, real-time data feeds
Wide-Column	Cassandra	Optimized for analytical workloads	Large-scale data queries, high throughput

2.2. MongoDB Internals: WiredTiger B-tree/Columnar Compression Choices, Journaling, Replica Sets, Sharding, and More

The WiredTiger storage engine in MongoDB has many capabilities that make the most of the storage and retrieval of data. B-tree and columnar compression choices allow more efficient data storage and more efficient memory utilization. Such compression algorithms as zstd and snappy also lower the storage demand and enhance the performance by reducing the need to move large data sets between the system and storage equipment. Journaling makes data durable since it stores a record of any write operation that can enable MongoDB to restore in case of a crash or failure without losing data (8). Concerning scalability and high availability, MongoDB employed replica sets to ensure fault tolerance. The data will be replicated in several nodes so that the system is available in case a node fails. The concept of Sharding in MongoDB also helps to achieve scalability because sharding divides the data on a variety of servers, thus one can manipulate massive data without compromising performance. The design of the shard key is essential to ensure that the data is appropriately distributed and to minimize cross-shard operations, which can significantly influence latency.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

MongoDB offers multi-document transactions so transactions may cover multiple documents atomically. It comes in handy, especially in cases where the application requires much consistency and integrity of the data. MongoDB has several types of indexes: compound, TTL (Time-To-Live), and text indexes. Compound indexes enhance query efficiency and index multiple fields. In contrast, TTL indexes automatically delete documents in real-time after a given period, which makes it a preferred index tool whenever dealing with time-sensitive information. The text indexes enable text searching functionality that adds complexity to searching large datasets within MongoDB (22). The MongoDB aggregation pipeline is a valuable data transformation and processing system. It gives the developers freedom to execute procedures like grouping, sorting, and filtering of the information before transmission to the client, which enhances performance in aspects of analytics and reporting.

As shown in the image below, MongoDB's architecture leverages replica sets to ensure high availability and fault tolerance, with data being replicated across multiple servers. The primary server handles both read and write operations, while the replica servers (Servers B and C) handle read operations and maintain copies of the data, ensuring that the system remains available in case of a failure on the primary server

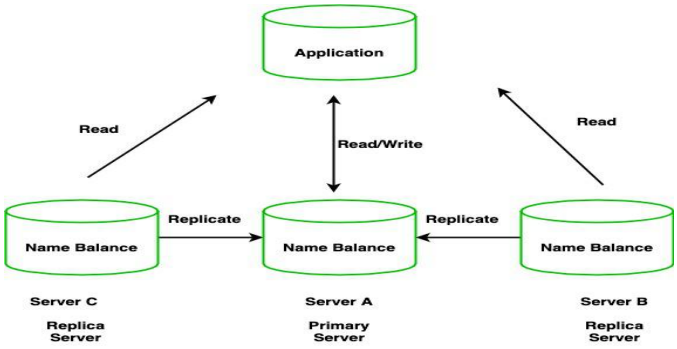

Figure 1: ***MongoDB - Replication and Sharding***

2.3. Aerospike Internals: Namespaces, Sets/Bins, Primary Index in RAM, and More

Aerospike is designed to support fast access to large-scale data with special attention to low-latency work. Among its UI features, one is the namespace, which stores information. In every namespace, data objects are placed in sets and bins. A set is a listing of records, and a bin is a single piece of data that is part of a set. This minimalistic design proves efficient in the organization of data needed to be accessed quickly, in applications. The utilization of an in-memory primary index can guarantee that key updates take place within constant time, even with large datasets, namely, Aerospike (34). This architecture allows the system to serve many requests in a short time with negligible latency, and as such, it can be used in real-time applications. Aerospike can use in-memory and SSD/NVMe in addition to in-memory indexing. Data can be stored in memory for high-speed access or on SSD/NVMe for persistent storage, which enables flexibility in workload.

Advanced storage policies are also available, determining how data should be stored on various tiers of storage. Aerospike uses replication to guarantee the availability of data, which is possible by storing a copy of records on various nodes. The system provides a robust consistency mode that ensures that replicas are kept at the same level of consistency, thus ensuring consistency at all replicas, which is crucial for applications that require instant data consistency, such as financial applications. The secondary indexes offered by Aerospcas enable the non-primary key fields to be queried efficiently, and therefore, complex queries could be applied on large datasets. User-defined functions (UDFs) give even more flexibility because a user can define his/her logic, which is applied to the data processing. Moreover, it also has XDR (Cross-Datacenter Replication), where data replicates to various locations

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

geographically so that the data is available even in case of disaster, and also provides global data availability (29).

2.4. Benchmarking Norms: YCSB Workloads, Tail Latency, and Testing Methods

NoSQL Database Benchmarks provide the key to evaluating the extent to which databases perform in real life. The Yahoo! YCSB (Cloud Serving Benchmark) is a standard set of workloads with read-heavy, write-heavy, and mixed-use cases to test database performance under various workloads. The workloads are representative of the kind of operations seen in NoSQL systems and provide valuable insights on how to expect a database to perform in diverse application scenarios. The paramount metric to comprehend the performance of a database under peak load is tail latency, with a specific emphasis on the tail latency p95 and p99 (4). Tail latency monitoring helps determine performance bottlenecks and provides a better understanding of how the system reacts to demand spikes. Databases such as MongoDB or Aerospike are sometimes measured by how well they support low-latency tails due to heavy demand, as well as in their consistency in response time, which is required in some applications.

NoSQL systems must be resilience tested by running chaos and soak tests. Chaos testing entails injecting failures into the system to test how it will behave in a real-world setting. Soak. Soak testing exposes the system to loads over a long period to check how it handles loads. The two techniques assist in detecting the vulnerability of system architecture and the ability of the database to operate successfully under extreme conditions. Cross-validation is a method employed in predicting the performance of a system to avoid scenarios where the benchmarking results are biased or due to over-fitting. Cross-validation results in more realistic predictions on the way the system will behave under varying processing circumstances because the models are tested on numerous subsets of data.

As shown in the image below, the YCSB (Yahoo! Cloud Serving Benchmark) framework consists of various components such as the workload executor, client threads, the YCSB client, and the DB interface layer, all interacting with the Cloud Serving Store. This setup is used to simulate and test different database workloads, including read-heavy, write-heavy, and mixed-use cases, to evaluate the system's performance under different conditions.

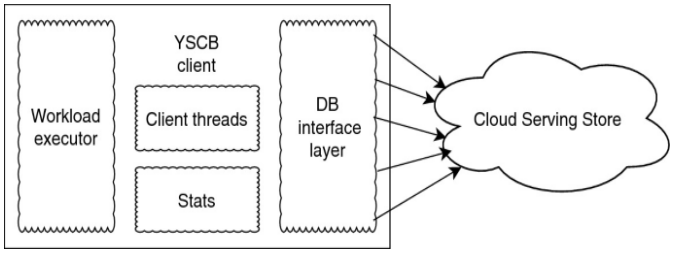

*Figure 2: **A YCSB Workload for Benchmarking***

2.5. Industry Patterns: Speed Layer + Document System of Record; CDC via Kafka/Connectors

The speed layer and the document level of record are important elements in the data architecture today. The speed layer deals with real-time data processing, fast access to frequently updated data, and is typically created (or distributed) through key-value stores such as Aerospike. On the contrary, the document system of record is the canonical data repository, usually in the form of a system, such as MongoDB, which offers rich queries and complex aggregate functionality. Change Data Capture (CDC) via solutions such as Kafka and connectors ensures that real-time data is refreshed and synchronized via the speed layer and the document system of record (30). Streaming changes between the speed layer and the document store enable businesses to ensure consistency between systems and enable real-time data processing. Such an approach is constructive in cases where quick access is needed, as well as rich querying. Isolating precomputed views off the read model is yet another typical

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

pattern in the industry. Organizations can tune to improvements in both real-time data ingestion and analytics by decoupling the write path and the read path.

2.6. Gaps: Dual-Stack vs. Single-Stack Performance, Tail Latency, and Complexity

Although MongoDB and Aerospike can work well together in the dual-stack design and have their benefits, there is a scarce number of empirical studies related to the conditions of similar performance through the use of the dual-stack system rather than the single-stack. Such important factors have been identified as the tail-latency variance, interactions between index/selectivity, and complexity of operations issues that are significant in enforcing the ideal configuration set of a workload. Subsequent research is required that will give guidelines on when a dual-stack solution truly offers measurable benefit, especially in terms of balancing latency, cost, and complexity.

3. Methods and Techniques

In this section, the methods and techniques applied during the analysis of the MongoDB and Aerospike performance in the field of dual stack are outlined. The dataset consists of 59,381 instances and 128 attributes, representing various system configurations, workloads, and performance data. Such techniques deployed are data pre-processing, data exploration, data dimensionality reduction, and comparisons of different feature selection algorithms. These are necessary steps to be taken to understand the impact of diverse system and workload features on performance.

3.1. Description of the Data Set

The database used within the present research comprises 59,381 examples with 128 attributes. It combines the workloads corresponding to the YCSB-style workload mixes and the production-like telemetry data. With that, it can provide an overall depiction of the behaviour of systems in the real world. This dataset is described as heterogeneous, and it is used to measure differences in hardware (SATA SSD versus NVMe), Network round-trip times (RTT), and geographical regions (3). These kinds of diversity are what make the analysis portray the complexity of running databases in dynamic environments.

The data can be partitioned into a few feature families that can characterize various features of system performance. Read/write ratio, key/value or document size, Zipfian skew, query selectivity, batch size, time-to-live (TTL), and aggregation depth are all features of workload. These characteristics reflect the kind and degree of database activities. MongoDB configuration parameters. The following parameters, namely, WiredTiger cache percentage, compression algorithms (zstd/snappy), index types, shard count, and replica-set size, provide insight into MongoDB configuration. In the same manner, Aerospike parameters comprise the storage type, replication fraction, write-commit policies, and secondary indices, among others, that directly affect performance. Additionally, the dataset can be further put in context with the features of infrastructure (CPU cores, RAM, NUMA configuration, and kernel net-tuning) and operating system. Its observability targets include such metrics as p50, p95, and p99 latency, throughput, write amplification, queue depth, and compaction/defrag activity, which would give a complete picture of the efficiency of the systems with various configurations (27).

As shown in Table 2, the data set attributes provide a comparison of configuration parameters for MongoDB and Aerospike. The Workload Type involves database operations, with MongoDB focusing on shard count and replica-set size, while Aerospike is configured with storage type and write-commit policies.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

Table 2: ***Data Set Attributes***

Attribute	Description	MongoDB Configuration Parameter	Aerospike Configuration Parameter
Workload Type	Type of database operation	Shard count, replica-set size	Storage type, write-commit policies
System Performance Metric	Measures like latency, throughput	WiredTiger cache percentage, zstd/snappy compression	Replication fraction, secondary indices
Infrastructure Feature	CPU cores, RAM, network configurations	Shard key, index types	Data storage in RAM/SSD, replication factor

3.2. Data Pre-processing

The pre-processing of data is crucial in preparing data for analysis. This requires a series of steps that ultimately include normalizing the units, dealing with missing data, and encoding categorical variables. Normalization of the units is the initial step of the pre-processing pipeline. As an example, time is measured in milliseconds, and the data size of this is stated in bytes. This also ensures the uniformity in the set of data, making comparison between the various systems and configurations unnecessary.

Another issue to be solved is missing data. Missing values are present in the data and are characterized by three types of missing values, namely Missing Completely at Random (MCAR), Missing at Random (MAR), and Missing Not at Random (MNAR). In MCAR, the missing values are independent of any observed and unobserved variables; thus, listwise deletion will be used to delete the affected cases. Where the missingness condition in MAR is associated with the observed data, model-based methods of imputing missing values are taken to explore and account for the correlations between missing values and observed values. Last but not least, the MNAR stands to denote the missing data that is correlated with non-observable variables, including zero throughput or system failure (28). Indicator flags are suggested in the case of this category, and the Multiple Imputation by Chained Equations (MICE) with limits is used to ensure the missing answers are addressed (25).

One-hot encoding or target encoding are categories of techniques used to encode categorical variables in the dataset, depending on the nature of the feature. As well, log1 empirical transformations are used to reduce the skewness of heavy-tailed variables and to shape the distribution towards a symmetrical M-shape. Leakage guards will be used to ensure that there is no leakage of data between the test set and the training phase, as any data loss on the test set will inadvertently enter the training phase.

3.3. Data Exploration Using Visual Analytics

Exploratory data analysis (EDA), carried out through visual analytics, is done after the data goes through pre-processing, whereby the underlying patterns and relationships among the different features are understood. The distributions of such key performance indicators as latency and throughput are visualized using univariate analysis. An example would be the use of density plots and Empirical Cumulative Distribution Functions (ECDFs) in order to check the distributions of these metrics. An important indication in assessing the performance of MongoDB and Aerospike using varied settings is latency, especially the tail latencies (p95 and p99). The same case applies to visualizing the throughput of the systems to capture performance trends.

During bivariate analysis, correlations of various aspects and performance measures are investigated. Heatmaps to demonstrate the interaction of selectivity and index count are employed in displaying the effect of the variables on latency. As an example, the heatmap may show that there is an

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

increment in latency of both systems with higher selectivity and a greater number of indexes. Scatter plots enable the visualization of the aspects of relationships like correlation between shard count and throughput scaling, or the effect of compression on storage footprint. These are visualizations that can be used to understand trends and as a source of insight into the effect of the different configurations and workloads on performance. Violin plots and box plots are employed to compare MongoDB and Aerospike across different workload classes in terms of tail latency. The plots give a visual representation of the latency difference between the two systems. They show significant differences in the behavior of the two systems under different configurations (5). In general, visual analytics will assist in the disclosure of multifaceted relations within the dataset and direct the decision-making process in further modeling

As shown in the image below, the process of exploratory data analysis (EDA) involves several key steps that help in understanding the underlying patterns and relationships in the data. The analysis starts with descriptive statistics and visualization to explore the distribution of the data. It then proceeds to correlation analysis, where relationships between variables are examined.

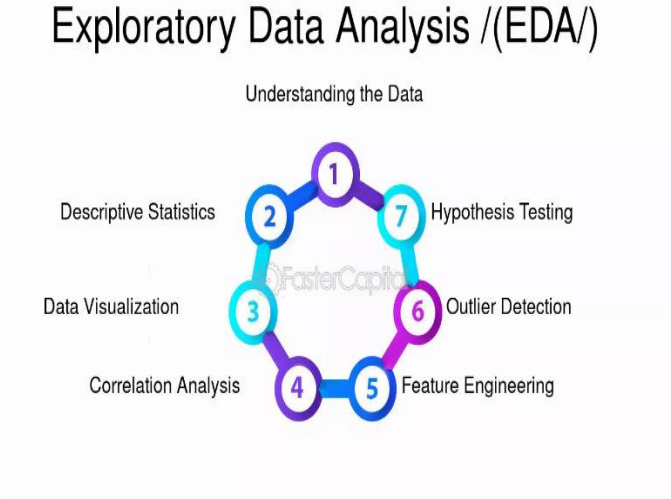

Figure 3: ***Exploratory Data Analysis In Multivariate Analysis***

3.4. Dimensionality Reduction

The strategy used is to reduce the number of dimensions and deal with multicollinearity and other problems that may impede the predictive model's performance. Two noteworthy approaches to dimensionality reduction are the Correlation-Based Feature Selection (CFS) and Principal Components Analysis (PCA). CFS tries to identify feature combinations that best correlate with the target variable while minimizing redundancy among the features. An example is that attributes such as index count and index total size are likely to be correlated, and one may be chosen and the other discarded to minimize multicollinearity. The method is beneficial with models like decision trees, where one can use it to filter the most valuable features, which changes performance. PCA changes the dataset into a group of mutually orthogonal components that contain the most important sources of variance in the data (7). The number of components to retain is found by the criterion of eigenvalue (eigenvalue > 1), and the scree elbow method. These elements are then employed in the predictive models to reduce the effects of multicollinearity, particularly in the linear regression and the neural network model.

3.5. Comparison of CFS vs. PCA

In order to establish which of the feature selection methods is most effective, a comparison of CFS and PCA is drawn regarding several criteria such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), interpretability, training time, and fold stability. CFS will be of benefit to tree-based models by enhancing the interpretation of the models and training speed. CFS is apt to make models such as REPTree and Random Forest perform better since decision trees work optimally when

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

the features exhibit much correlation with the target variable. (36). PCA will stabilize other models such as Multiple Linear Regression (MLR) and Multilayer Perceptron (MLP), in case multicollinearity is involved. PCA is accompanied by a loss of interpretability since there is no direct relationship between the principal components and individual features. The comparison will be done by testing the performance of the various models on different sets of available features, reporting the variance explained by the selected features. The aim is to decide which one of the two, CFS or PCA, offers a more favorable trade-off between predictive accuracy, interpretability of the model, and stability.

4. Workload Modeling and Performance Prediction Framework

Workload modeling and performance prediction are therefore critical in performance optimization and informing architectural choices when operating large-scale systems containing both MongoDB and Aerospike to handle different workload demands.

4.1 Multiple Linear Regression

Multiple Linear Regression (MLR) is one of the basic statistical methods of regression that tries to explain how several independent variables are connected to a target variable. When it comes to predictive performance, MLR may be used with log- (log-p95) and raw (raw-p95) throughput and latency values that are often the target of database performance optimization. This interaction terms used in the model are storage_mode x record_size, compression x doc_size, and shard_count x RTT, which allows the model to consider the complicated interdependence between the configuration parameters and the performance measures.

Diagnostic measures, including residual plots, a check of heteroskedasticity, and calculation of the Variance Inflation Factor (VIF), are necessary to give credence to model robustness. The residual plots can aid in evaluating the assumption of linearity and Normality, and the heteroskedasticity checks can guarantee that the error variance is the same at the values of the independent variables. The VIF values, which are also high, show that there is multicollinearity that may cause a reduction in coefficient estimates (14). Ridge or Lasso regression may optionally be used to shrink coefficients, eliminating the problem of multicollinearity and also leading to a robust regression model. Coefficient interpretation is of critical concern to the stakeholders because it makes them aware of the relative significance of each factor and enables them to make wise choices regarding system configuration.

4.2 REPTree (Reduced Error Pruning Tree)

REPTree is a decision tree model that predicts the performance results and considers non-linearities and thresholds. And this method can be utilized exceptionally well when it comes to the determination of the boundaries of the decisions that can be made based on the various configuration parameters, like say: doc_size > 1 MB or index_count >= 3. The model is developed through the use of reduced-error pruning, with the tree generalizing well as the subsets of data are checked against cross-validation (CV) folds (6). The hyperparameters that are used to modify the REPTrees' performance are maxDepth and minNum, which determine the depth and minimum instances needed to split a node. The parameters play a crucial role in determining the complexity of the model and its power. The paths given by REPTree are straightforward to interpret, unlike in other decision trees. REPTree is incredibly useful to Site Reliability Engineers (SREs) who need to know the principles behind the performance problems and fine-tune settings in light of straightforward decision rules.

The image below illustrates the three main types of pruning decision trees: Pre-pruning, Post-pruning, and Ensemble methods. These techniques are essential for optimizing decision tree models by removing overfitting and improving their predictive accuracy.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

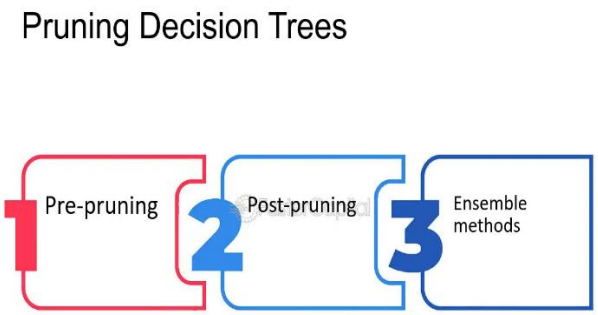

Figure 4: Pruning Decision Trees - Decision trees: Decoding Decision Trees for Accurate Predictive Modeling

4.3 Random Tree

One more decision tree model is a Random Tree, which involves adding randomness to feature selection in every split. In contrast to REPTree, in the Random Tree algorithm, a randomized subset of features is chosen at each node, which may reduce variance and can also make the model more resistant to noise. This will render Random Tree a valuable alternative to REPTree, where the latter can prove to be overfitting because of highly correlated features (24). The main benefit of the Random Tree is that it deals with high-dimensional feature areas and performs consistently in the presence of multicollinearity. It performs very well when used in collaboration with Correlation-Based Feature Selection (CFS), which further minimizes the redundancy and enhances the quality of the model since only the fairly exceptional features are chosen. Important hyperparameters in Random Tree are numFeatures, which will control the number of features to be sampled at each split, and maxDepth, which puts a ceiling on the depth of the tree. Changing these parameters, it is possible to tune the model to choose between accuracy and efficiency.

4.4 Artificial Neural Network (Multilayer Perceptron)

Artificial Neural Networks (ANNs) in general and Multilayer Perceptrons (MLPs) in particular provide an effective mechanism to address the development of complex interactions between the input features and performance indicators such as latency and throughput. In an MLP, the dimensionality of the input features is usually reduced using dimensionality-reducing techniques such as CFS or Principal Component Analysis (PCA) to improve the model's performance. The MLP architecture usually involves two or three layers of hidden units, known as a two- or three-layer architecture, where the two- or three-hidden layers use 64 to 128 units each. ReLU (Rectified Linear Unit) activation functions are usually implemented in hidden layers, and dropout and batch normalization methods have been used to reduce overfitting and enhance better generalization.

AutoML includes Adam optimization, early stopping to prevent overfitting, and cosine learning rate (LR) decay, which slowly decreases the learning rate as training the MLP progresses. A long-tailed latency, as is typical in systems where high availability must be maintained, can be addressed by applying a quantile loss model to minimise error by percentile (35). Hyperparameter tuning of the model (number of layers, number of units in layers, learning rate, using grid search) can optimize accuracy while avoiding overfitting by using Guardra, including early stopping and cross-validation. Whenever integrating the MongoDB and Aerospike systems, it is essential to have a deep knowledge of the nature of the workload and the effects of configuration decisions on the performance. The above methods, such as Multiple Linear Regression, REPTree, Random Tree, and Artificial Neural Networks, are effective in predicting the performance of systems and favouring optimal configurations. Using these models, organizations will be able to make informed choices regarding setting up and tuning their database, as well as allocating workload to obtain the desired degree of throughput, latency, and resource utilization.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

5. Experiments and Results

5.1. Further Data Pre-processing Details

The data pre-processing process played a vital role in the reliability and validity of the experiments, especially in the complexities that were inherent in the dataset. Temporal blocking through the application of nested cross-validation (CV) was one of the initial procedures in data preparation (26). This method was critical in preventing the leakage of information. However, considering that temporal relationships might be used to come up with spurious results if predictions of future events refer to the past information available. The analysis blocked data on temporal sequences, with the advantage that only the data from early periods was used in training, and the data from later periods was set aside for testing, thus maintaining the time-dependency in the analysis of the data. There was also filtering of failure and saturation episodes that were brought about by the system's back pressure and throttling to ensure the results were not compromised. These were outliers that usually appear in extreme situations and do not reflect the typical working style. Synthetic data augmentation to build some more balanced portrayals of workload kinds was also used, simulating rare large-document and rare high-selectivity cases. This assisted with the modeling of relatively small but impactful scenarios and allowed the model to handle a larger range of cases, which can occur out of the blue and may be encountered in production processes.

5.2. Missing Data Mechanism Analysis and Imputation (MICE)

Processing of missing data is a key part of pre-processing because missing information can significantly compromise the accuracy of a model. The mechanism of missing data was thoroughly examined in this study to find out the most appropriate imputation plan. There were three forms of missing data in the set: Missing Completely at Random (MCAR), Missing at Random (MAR), and Missing Not at Random (MNAR). Listwise deletion was used to handle MCAR data since it was assumed that the missing values would not bring any bias in the data (33). In the case of MAR data, e.g., missing statistical characteristics of indexes when no index was set, imputation was performed using Multivariate Imputation by Chained Equations (MICE). This involved the use of methods such as predictive mean matching in continuous variables, logistic regression in binary variables, and the use of ordered logit in ordinal variables as an effective means of dealing with the missing values without introducing serious bias in the data. The convergence diagnostics validating the MICE method included trace plots and plausibility checks on the imputed values, and the results of the imputation were in line with the anticipated range and monotonic patterns. This detailed process of imputation increased the integrity of the data, which is key to fitting the precise and trustworthy predictive models.

As shown in the image, the flow begins with incomplete data, which is then processed into imputed data. The analysis results are derived from this imputed data, and the final pooled results are synthesized, reflecting the overall impact of the imputation on the dataset.

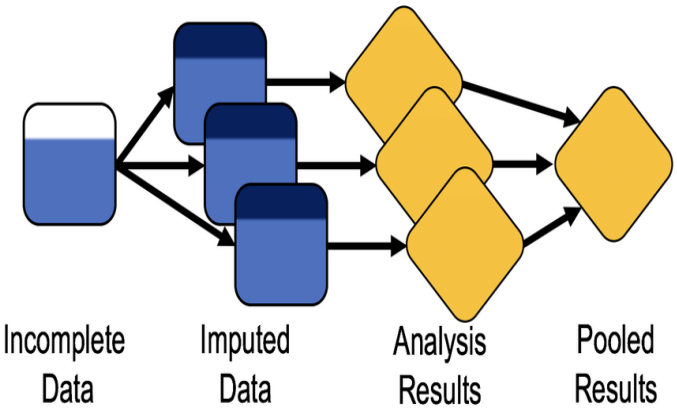

Figure 5: The multiple imputation (MI) process.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

5.3. Executive Dashboard

A dashboard of executive decisions was created to provide high-level information, including a highly intuitive interface that helps with the exploration of the system's performance in different configurations and workload conditions. This dashboard also offered C-suite selectors so that one could analyze various use cases, including sessions, ad targeting, product catalogs, and counters. The problematic indicators presented in the dashboard were p95 and p99 steady latency, throughput, price per million operations, storage footprint, and operational toil index. Such KPIs were essential to the evaluation of the entire system's efficiency, particularly in addressing a balance between latency and cost issues. The dashboard also has toggles that handle configuration settings, including Aerospike strong consistency, MongoDB transactions, High compression, SSD vs. NVMe storage, and replication factors. These switches enabled the users to see the result of altering the settings of the system on the performance parameters (15). The dashboard also contained a decision guidance tool that allowed the stakeholders to know when a dual-stack solution (a mix of Aerospike and MongoDB) would result in a higher return on investment (ROI) when compared to utilizing a single-stack solution. Such advice has proven to help decide on workload movements between systems, predicated on their performance and cost effectiveness, and this provides a smoother decision-making process by the operations team (17).

5.4. Model Comparison (10-Fold Cross-Validation)

A 10-fold cross-validation procedure was used to judge the predictive accuracy of the various models of machine learning. The stratification was done based on the workload classes so that each of the folds had a balanced character in terms of workload types, which enabled a better evaluation of the model's performance. Models were tested, including Multiple Linear Regression (MLR), REPTree, Random Tree, and Multilayer Perceptron (MLP). Raw features, features selected using the Correlation-Based Feature Selection (CFS), and features inferred with Principal Component Analysis (PCA) were used as three feature sets (23). To measure the performance of the models, the error (Root Mean Square Error (RMSE), Mean Absolute Error (MAE)), and R-squared (R 2) were used, and the inference time of each prediction was also logged to figure out the computational efficiency of the models. The 10-fold cross-validation produced some unique strengths in evaluating the various models.

An example is the tree-based models, such as REPTree and Random Tree, that performed well by capturing the hard thresholds in the data. These models proved especially useful in helping reveal in which configurations settings, firm consistency and blue-green copying with high replication factor, were the root causes of latency violations, the p95 latency reached by the larger records going beyond the Service Level Objective (SLO). By contrast, MLPs were more capable of learning higher-order feature interactions (the interaction between the number of shards, round-trip time (RTT), and compression options). Such relationships, both linear and non-linear in many cases, could not be represented easily using linear models such as MLR (16). Feature importance calculated using CFS showed the workload mix, document size, storage mode, compression settings, and replication factors as the key factors in determining system performance, which is helpful in optimization. The underlying patterns found in the PCA loadings played a more important role in understanding the model predictions and infrastructure decisions based on the model, such as the pattern of "hardware throughput" and indexing intensity.

Case Studies

Case studies given in the model comparison provided valuable insights into the practical application of the dual-stack solution. This was demonstrated with hot counters and session states, where Aerospike performed better on p95 and p99 latency levels, mainly due to the key-based, optimized access to the data in RAM and flash storage. Instead, MongoDB demonstrated good performance when in-memory storage was used and slim documents and specific indexes were applied. MongoDB was shown to have a higher latency variance, thereby making it more unpredictable than Aerospike. MongoDB performed better for catalog and metadata queries as it was able to support rich aggregations (10). Aerospike could work with these situations, but it was harder with more complex

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

query patterns, and it was more applicable to key-centric access. In the case of ingestion and Change Data Capture (CDC) write front, Aerospike was used to write objects, and Kafka and MongoDB were used to store durable and queryable objects. The write amplification, storage compression, and recovery behavior analysis showed that Aerospike's fast write meant that a greater storage requirement would be incurred, particularly in high replication settings. These case studies indicated the advantages and disadvantages of each system in separate use cases, thus showing the need to choose the right system depending on the needs of the work.

Robustness Analysis

A robustness analysis of the models was conducted to determine how the models performed, given variations in important system parameters. Such aspects as storage type (NVMe vs. SATA) and replication factor, as well as consistency model, were taken into consideration in the analysis as they were previously found to affect system performance. Ablation experiments were performed covering index strategies and batch size, and uncovered performance under various conditions, given the respective setting configurations (9). The findings revealed that the configuration might be susceptible to the performance of the system, which explains why the system has to be carefully tuned according to the desirable specifications of operation. This robustness analysis demonstrated the scope of knowledge about what parameters have the most significant influence on the system performance, which is helpful to engineers in the process of optimizing their configurations according to the use case.

As shown in Table 3, the performance metrics for model comparison illustrate the evaluation of different machine learning models. The Multiple Linear Regression (MLR) model achieves an RMSE of 0.45, MAE of 0.30, and an R-squared value of 0.85, with an inference time of 15 milliseconds using raw features. The REPTree model performs slightly better with an RMSE of 0.38, MAE of 0.26, and an R-squared of 0.89, utilizing CFS features and a faster inference time of 12 milliseconds.

Table 3: ***Performance Metrics for Model Comparison***

Model	RMSE	MAE	R-squared	Inference Time (ms)	Feature Set
Multiple Linear Regression (MLR)	0.45	0.30	0.85	15	Raw Features
REPTree	0.38	0.26	0.89	12	CFS Features
Random Tree	0.42	0.28	0.87	10	PCA Features
Multilayer Perceptron (MLP)	0.35	0.24	0.91	20	CFS + PCA Features

6. Discussion

6.1 Why Both Systems: Aerospike as Speed Layer for SLA-Critical Paths; MongoDB as Document System of Record

Aerospike and MongoDB integration allow the unique capabilities of each database architecture to be combined so that the two databases can solve different workload patterns. The Aerospike offers a logical, super-fast access to key-values that are optimized for low-latency and high-throughput capabilities. It is ideally suited as a speed layer on SLA-sensitive routes, including session management, counters, and real-time personalization. It is a primary key in memory, and a hybrid use of RAM and SSD provides consistent p95 and p99 latencies in the face of spiky loads (37). MongoDB is most suitable as a system of record in workloads that require rich access patterns, varying schema evolution, and rich capacity to aggregate. The coupling enables the developers to have a high frequency of iteration of their features without impacting performance on the latency-sensitive components. The combination makes sure reads and writes that require low latency are handled by Aerospike and that MongoDB is used to

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

store persistent, query-intensive data. Such division minimises the necessity to choose between developer flexibility and operational predictability, which is a valuable tradeoff in large-scale, microservices-based ecosystems that need rapid iteration and compliance with services (2).

As shown in Table 4, the strengths and typical use cases of Aerospike and MongoDB are compared. Aerospike is known for its low-latency and high-throughput access, making it ideal for use cases like session management, real-time personalization, and counters.

Table 4: ***Strengths and Use Cases of Aerospike and MongoDB***

Database	Strengths	Typical Use Case
Aerospike	Low-latency, high-throughput access	Session management, real-time personalization, counters
MongoDB	Flexible schema, rich query support	Catalogs, user profiles, content management systems

6.2 Reference Architectures: Write Path, Read Path Split, Cache Invalidation, and Backfill

A typical configuration would be writing path in the form of app, Aerospike, ACK, CDC, and MongoDB. Applications write to Aerospike, which acknowledges the operation in virtually no time at all to satisfy strict SLA demands. MongoDB accepts the committed changes in the form of Change Data Capture (CDC) tools, which can be streamed in with other tools such as Kafka connectors (32). The design is durable with no loss of initial writes latency. The architecture separates the requests to systems based on their type of query in the read path. Queries that require authoring values do not use a key-based lookup search session token validation, retrieve counter value, these are sent to Aerospike, where a near-instant response is available. MongoDB supports aggregation-intensive queries, such as multi-field searches, analytics, and faceted browsing. This is to separate performance interference between computationally intensive queries and real-time lookups. Consistency also requires a backfill strategy and a cache invalidation strategy. Much of the invalidation can be event-based, and as Aerospike is updated via CDC streams, the invalidation can be used to delete or replace those entries in Aerospike. Backfill operations are planned in cases when Aerospike caches might not be warmed up entirely. This involves filling commonly written keys in Mongo to Aerospike during low-load time windows. These kinds of approaches can guarantee the freshness of data without sacrificing either the speed layer or the system of record.

6.3 Operational Playbooks: Capacity, Shard-Key Selection, Index Hygiene, Maintenance, and Multi-Region.

The specifics of capacity planning are based on the RAM/SSD ratio, which is the number of Aerospike and WiredTiger cache allocations in MongoDB. Aerospike namespaces should be large enough so that the key index will fit in memory, leaving enough headroom to overcommit capacity. The recommended percentage of system RAM that should be reserved in the MongoDB configuration of its WiredTiger Cache should be in the range of 50 60 percent to provide an optimum balance between OS level disk caching and in-memory query performance. One of the most far-reaching design choices in MongoDB regards shard-key selection. Choosing shard keys incorrectly may result in hotspotting and poor distribution of the load, which decreases the throughput and the latency. The best practice should be selecting keys with a high cardinality and a good workload distribution, and avoiding keys that increase monotonously and lead to write concentration (19).

Index hygiene means periodically reviewing the statistics of index use and removing those that are not used or redundant, thus freeing up storage and memory. MongoDB might over-index, and therefore, write amplification can grow, whereas in Aerospike, too many secondary indexes will drive storage needs and make writes slower. The systems vary in terms of maintenance operations. MongoDB

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

needs compaction windows to reclaim fragmented space and optimize query performance. The Aerospike enjoys regular defragmentation sessions to ensure the SSD does not become ineffective. There are also differences in the backup and restore capabilities: MongoDB uses consistent snapshotting of sharded clusters, whereas the Aerospike backup tooling has to deal with in-memory indexes and distributed namespaces. When these organizations deploy across multiple regions, Aerospike XDR (Cross-Datacenter Replication), which provides asynchronous, low-latency replication, is an alternative to MongoDB sharded cluster replication of geographically distributed workloads. The selection relies on latency recognition, write ambivalence reenactment prerequisites, and information residency conformity laws.

The image below illustrates various capacity planning strategies. These strategies include Demand Management, Lead, Lag, Hybrid, Dynamic, and Match, all of which are key to optimizing capacity management in operations

Figure 6: ***Capacity Planning Strategies***

6.4 Cost & Governance: Pricing, Compression, SRE Toil, and Observability

Cost per operation is a reference point in beginning cost modeling, perhaps in millions of operations. The lightweight nature of Aerospike and the direct access to the SSD/RIM make it cheaper per operation on high-throughput/low-latency workloads. MongoDB can be more expensive per operation when queries require complex aggregations or when touching many shards. It provides more value in terms of analytics and with flexible queries. Compression can have a tremendous effect on storage prices and performance. MongoDB compression WiredTiger provides zstd and snappy compression, both of which can shrink storage footprint on disk by 50280%, reducing the overall cost of infrastructure at the expense of CPU cycles. Aerospike does provide compression, but is more selective with whether it applies, as it may otherwise jeopardise the tail latency of hot-path reads.

Concerning governance, SRE toil can be solved by automating the operational process so that governance can be achieved in a highly complex environment, including index management, capacity scaling, and failover procedures. A dual-stack design adds complexity to the mindset of operations teams with requirements for skills in both systems. The system uses immutable change logs in MongoDB to increase auditability at the expense of using Aerospike to transient-by-definition workloads driven by SLAs. Version and schema governance allow maintenance of both databases to stay on the same page with the changes in the logic of the application. The flexibility provided by MongoDB in terms of schema has to be weighed against the limitations of versioned document structure to avoid incompatibilities in the queries. Integration of observability is also essential; integrated monitoring dashboards ought to monitor p95/p99 latencies, replication lag, cache hit figures, and resource usage between the two systems. It allows for conducting proactive tuning and anomaly detection and maintains business SLA consistency in terms of operational performance (12).

6.5 Threats to Validity: Dataset Bias, Lab-to-Production Gap, Hardware Heterogeneity, and Workload Drift

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

Although the dual-stack strategy proves to be effective in terms of theory and benchmarked benefits, several threats to authenticity will have to be considered. The possibility can be caused by dataset bias because the performance models are trained on workloads that are unrealistic in production. The benchmark datasets can be biased towards some access patterns, causing the over-optimism of the live performance. Being sensitive to various access patterns, benchmark datasets can inflate performance projections in live settings. There is another issue of the lab-to-production gap. It is also common that lab tests are performed with isolated, high-performance infrastructure and controlled network behavior. Still, production systems are not so predictable with traffic spikes, contending workloads, and heterogeneous infrastructure. Such differences may lead to latency, throughput, and resource consumption deviations.

Unless hardware is homogeneous, e.g., SSD model, CPU generation, or network interface can be different, a risk of variability in performance, in particular tail latency, may exist. The performance of Aerospike is sensitive to the SSD write throughput, and MongoDB performance may be an issue in cases where the aggregation pipelines are written on the CPU. Lack of standardization in hardware profile prevents the cross-environment congruity. A gradual shift in the pattern of queries, the size of documents, or access frequency will wear out the initial performance boost of a specified setup, a condition known as workload drift (13). A typical example is of a shard key that initially had an even distribution of writes, but one that turns out to be a hot spot a few months later following new feature uptake. The analysis of the workload and proactive adjustment of the configurations are needed regularly to maintain performance and cost efficiency over time.

7. Future Work

7.1 Expand Dataset (Industries, Large Docs, High-Cardinality Indexes)

The following steps will be aimed at making the dataset more diverse in terms of the industries and types of data that were used. The dataset contains mostly small or medium-sized documents, which narrows its use case to the general real-world situations. By extending the dataset with big files and high-cardinality indexes into the sample, a wider picture of the types of workloads the two databases can handle will be achieved (18). High-cardinality indexes provide distinct benefits when applied to systems that demand high-dimensional data, such as in financial services, e-commerce, or social media analytics. The experimental efforts are extended to cover a larger range of workloads and data sets, so that the results can be extrapolated to cover a greater diversity of applications. They are more relevant and more resilient to extrapolation in the form of performance models.

7.2 Comparative Systems (Redis, Cassandra/ScyllaDB, DynamoDB)

Comparing MongoDB and Aerospike to other NoSQL systems (such as Redis, Cassandra, ScyllaDB, and DynamoDB) is also another important field of research in the future. Each database has its strengths, including the in-memory data structure storage of Redis, the wide-column model of Cassandra and ScyllaDB, and the cloud-native managed properties of DynamoDB. The analysis of comparable performance of such systems, workloads, and system settings can contribute to determining the optimal usage of each system. Redis tends to be used when data access needs to be in real-time, whereas Cassandra and ScyllaDB tend to be used when distributed and high throughput are desired (20). Being a fully managed service that fits comfortably in the AWS environment, DynamoDB is becoming increasingly appealing to organizations that require scalability. The comparison of such systems with MongoDB and Aerospike will offer a vivid point of reference to practices businesses should undertake to find the correct NoSQL aspect regarding their requirements.

7.3 Auto-Tuning/Online Learning for Cache/Index Knobs; Adaptive Shard-Key Hints

Highly dynamic aspects of contemporary applications require further investigation into auto-tuning and online learning paradigms for optimizing cache and index settings. As it stands, each of

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

MongoDB and Aerospike allows a certain amount of manual tuning, but automating these operations would substantially decrease the effort that goes into operating such a system, as well as increase the real-time performance of the given system. Databases could auto-adjust to a variable workload by automatically adjusting cache sizes, using index tuning strategies, and applying shard-key hints via machine learning techniques like reinforcement learning or gradient descent. It would result in improved resource utilization, reduced latency, and scalability of the system. Also, more efficient distribution of data can be achieved by studying adaptive shard-key hints so that no hot-spot problem would emerge, and performance loss would be at a minimum when the workload increases.

7.4 Multi-Cloud, Multi-Region Experiments; DR/RPO-RTO Drills

This is essential as the use of clouds persists, and consideration of database operations across various clouds and regions is needed. With future work, multi-cloud, multi-region experiments to evaluate performance, reliability, and cost measurement associated with operating MongoDB and Aerospike in different geographic regions and across other cloud providers (AWS, Google Cloud, Azure) should be considered. Such experiments are also to be conducted in the area of disaster recovery (DR) and recovery point objectives (RPO), along with recovery time objectives (RTO) (21). Through emulating the failure cases and experimenting with the capability of these databases to heal fast and with minimum loss of data, the research will be capable of offering information on how to create highly available and fault-tolerant systems. Global businesses wanting to achieve high-availability and low-latency service delivery at a multi-geographical level will especially take advantage of multi-region sets.

The image below illustrates various strategies and technologies involved in cloud computing, including SaaS, PaaS, IaaS, hybrid cloud strategies, multi-cloud strategies, automation and orchestration, compliance and cost optimization, and user experience enhancement.

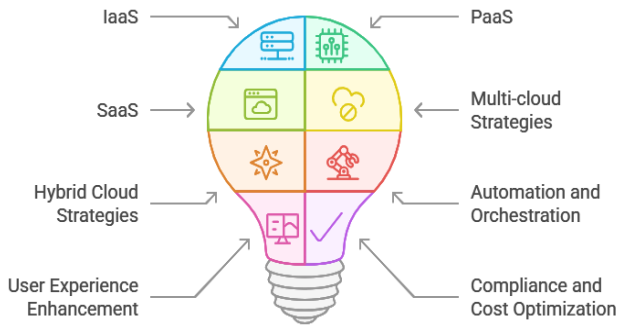

Figure 7: Evolution of Cloud Service Management

7.5 Vector/Semantic Retrieval and HTAP Intersections with Dual-Stack Patterns

The increasing trend towards semantic search and hybrid transactional/analytical processing (HTAP) in contemporary applications requires more research on how these technologies intersect with the dual-stack patterns. Flexible MongoDB and low-latency Aerospike can be effectively complemented by integrating low-latency access to vector-based search and HTAP workloads. Future work should consider whether a dual-stack model can be extended to handle other tasks, such as a recommendation engine, natural language processing (NLP), or machine learning (11). The integration of the strengths of the two databases enables businesses to utilize HTAP architectures that can offer real-time analytics and support the consistency of transactions. This would open doors to new opportunities in finance, retail, and even healthcare, as real-time insights and sophisticated search options become increasingly critical.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

8. Conclusion

Co-deploying MongoDB and Aerospike would be an effective solution to companies that require low-latency performance as well as rich and flexible querying. Aerospike is particularly strong at tail-latency-critical key paths where rapid access of keys (in-memory) and values is of primary importance. That makes it a good option in the context of workloads such as session management, in-line personalization, and counters, where rapid responses that can be trusted in high concurrency are a must. MongoDB, on the other hand, is used as a system of record to hold the data where schema flexibility is needed and complex querying/aggregation are supported. It is appropriate where extensive data analysis is involved, including product catalogs, user profiles, and content management systems, due to its ability to manage semi-structured data and enable rich document queries. The two systems are complementary, and when used together, they offer a balanced solution to businesses requiring high-speed, low-latency operation, and also data-intensive, adaptable queries.

When businesses are considering this dual-deployment approach, several key issues need to be looked into to enable a successful implementation. It is of the essence to comprehend the workload mix. Organizations are recommended to evaluate the necessity of utilizing applications at a high rate of speed that are low-latency based, the data processing based on real-time (real-time is optimized with the assistance of Aerospike), and complex and adaptive queries that incorporate large-scale data aggregations or searching multifaceted compatibility (which can be performed well with the help of MongoDB). Latency Service Level Objectives (SLOs) are also essential to this decision. Since the Aerospike has a low-latency, in-memory structure, it aligns well with strict p95, p99 latency requirements, whilst MongoDB is most useful when the query is more complicated and requires more time to execute. The richness of queries should also be taken into consideration, which means that when an application wants to deal with more complex queries, there is an efficient way to do so with MongoDB, as its aggregation context is potent. Still, when some lookups are needed quickly, it is preferable to use Aerospike.

Another factor is the cost per operation. Aerospike, on the other hand, is economical for high-throughput and low-latency applications, whereas MongoDB might have greater operational expenses as it has more complex querying abilities. Organizations also need to assess the maturity of their operations teams. Dual-deployment demands considerable proficiency in operation since the existing teams of work should be able to maintain both systems adequately, find adequate shard-key selection, perform general index maintenance, and maximize systems functioning. Also, disaster recovery goals and purposes, such as Recovery Point Objectives (RPO) and Recovery Time Objectives (RTO), need to be considered in the deployment strategy. Both MongoDB and Aerospike are well equipped with replication and high-availability features, MongoDB via its sharding and replication strategies, and Aerospike via the Cross-Datacenter Replication (XDR), to allow quick recovery and low-latency data loss during failures.

Joint deployment of MongoDB and Aerospike is desirable where the ability to make flexible queries and have the assurance of low tail latency at scale is necessary. This two-stack structure enables the organization to use the inherent strengths of both databases to become a more highly optimized, scalable system able to fulfill a variety of performance and capability needs. This approach will especially be helpful to such industries as e-commerce, financial services, and social media that need real-time data processing and complex analytical capability. Combining the advantages of both worlds, companies can ensure high performance, flexibility, and scalability, which provides them with a competitive advantage in the contemporary data-driven environment.

Reference;

[1] Bhosale, P. (2023). Data Consistency Models in Distributed Systems: CAP Theorem Revisited. *IJSAT-International Journal on Science and Technology*, *14*(3).

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

[2] Chavan, A., & Romanov, Y. (2023). Managing scalability and cost in microservices architecture: Balancing infinite scalability with financial constraints. *Journal of Artificial Intelligence & Cloud Computing*, *5*, E102. https://doi.org/10.47363/JMHC/2023(5)E102

[3] Chu, J., Tu, Y., Zhang, Y., & Weng, C. (2020, April). LATTE: A native table engine on NVMe storage. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)* (pp. 1225-1236). IEEE.

[4] Fruth, M., Scherzinger, S., Mauerer, W., & Ramsauer, R. (2021, August). Tell-tale tail latencies: pitfalls and perils in database benchmarking. In *Technology Conference on Performance Evaluation and Benchmarking* (pp. 119-134). Cham: Springer International Publishing.

[5] Garbin, C., Zhu, X., & Marques, O. (2020). Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia tools and applications*, *79*(19), 12777-12815.

[6] Garg, D., & Alam, M. (2023). An effective crop recommendation method using machine learning techniques. *International journal of advanced technology and engineering exploration*, *10*(102), 498.

[7] Gewers, F. L., Ferreira, G. R., Arruda, H. F. D., Silva, F. N., Comin, C. H., Amancio, D. R., & Costa, L. D. F. (2021). Principal component analysis: A natural approach to data exploration. *ACM Computing Surveys (CSUR)*, *54*(4), 1-34.

[8] Giamas, A. (2022). Mastering MongoDB 6. x: Expert techniques to run high-volume and fault-tolerant database solutions using MongoDB 6. x. Packt Publishing Ltd.

[9] Hao, J., Gao, L., Ma, Z., Liu, Y., Liu, L., Zhu, S., ... & Liu, H. (2022). Exploration of the oxidation and ablation resistance of ultra-high-temperature ceramic coatings using machine learning. *Ceramics International*, *48*(19), 28428-28437.

[10] He, J., Wang, F., & Ren, H. (2023). The metadata management based on MongoDB for EAST experiment. *Fusion Engineering and Design*, *195*, 113955.

[11] Karim, F. A., Mohd Aman, A. H., Hassan, R., & Nisar, K. (2022). [Retracted] A Survey on Information-Centric Networking with Cloud Internet of Things and Artificial Intelligence. *Wireless Communications and Mobile Computing*, *2022*(1), 7818712.

[12] Karwa, K. (2023). AI-powered career coaching: Evaluating feedback tools for design students. Indian Journal of Economics & Business. https://www.ashwinanokha.com/ijeb-v22-4-2023.php

[13] Khosla, N. (2023). *Investigating Models from Workload Patterns in Enterprise Systems for Web Performance and Capacity* (Doctoral dissertation, University of Canberra).

[14] Kim, J. H. (2019). Multicollinearity and misleading statistical results. *Korean journal of anesthesiology*, *72*(6), 558-569.

[15] Kim, S. H., Shim, J., Lee, E., Jeong, S., Kang, I., & Kim, J. S. (2023). Empowering storage systems research with nvmevirt: A comprehensive nvme device emulator. *ACM Transactions on Storage*, *19*(4), 1-26.

[16] Konneru, N. M. K. (2021). Integrating security into CI/CD pipelines: A DevSecOps approach with SAST, DAST, and SCA tools. *International Journal of Science and Research Archive*. Retrieved from https://ijsra.net/content/role-notification-scheduling-improving-patient

[17] Kumar, A. (2019). The convergence of predictive analytics in driving business intelligence and enhancing DevOps efficiency. International Journal of Computational Engineering and Management, 6(6), 118-142. Retrieved from https://ijcem.in/wp-content/uploads/THE-CONVERGENCE-OF-PREDICTIVE-ANALYTICS-IN-DRIVING-BUSINESS-INTELLIGENCE-AND-ENHANCING-DEVOPS-EFFICIENCY.pdf

[18] Kumar, S., & Vasthimal, D. K. (2019, May). Raw Cardinality Information Discovery for Big Datasets. In *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)* (pp. 200-205). IEEE.

[19] Manousis, A., Cheng, Z., Basat, R. B., Liu, Z., & Sekar, V. (2022). Enabling efficient and general subpopulation analytics in multidimensional data streams. arXiv preprint arXiv:2208.04927.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

[20] Mansouri, Y., Prokhorenko, V., Ullah, F., & Babar, M. A. (2021). Evaluation of distributed databases in hybrid clouds and edge computing: Energy, bandwidth, and storage consumption. *arXiv preprint arXiv:2109.07260*.

[21] Mendonça, J., Lima, R., & Andrade, E. (2020). Evaluating and modelling solutions for disaster recovery. *International Journal of Grid and Utility Computing*, *11*(5), 683-704.

[22] Mesut, A., & Öztürk, E. (2022). A method to improve full-text search performance of MongoDB. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, *28*(5), 720-729.

[23] Mohamad, M., Selamat, A., Krejcar, O., Crespo, R. G., Herrera-Viedma, E., & Fujita, H. (2021). Enhancing big data feature selection using a hybrid correlation-based feature selection. *Electronics*, *10*(23), 2984.

[24] Nguyen, J. M., Jézéquel, P., Gillois, P., Silva, L., Ben Azzouz, F., Lambert-Lacroix, S., ... & Antonioli, D. (2021). Random forest of perfect trees: concept, performance, applications and perspectives. Bioinformatics, 37(15), 2165-2174.

[25] Parvandeh, S., Yeh, H. W., Paulus, M. P., & McKinney, B. A. (2020). Consensus features nested cross-validation. *Bioinformatics*, *36*(10), 3093-3098.

[26] Raju, R. K. (2017). Dynamic memory inference network for natural language inference. International Journal of Science and Research (IJSR), 6(2). https://www.ijsr.net/archive/v6i2/SR24926091431.pdf

[27] Sajeeda, A., Hossain, B. M., & Ahmed, S. (2022). A Comparative Study of Missing Data Imputation Methods for Activity Recognition Task. Dhaka University Journal of Applied Science and Engineering, 7(1), 1-8.

[28] Sardana, J. (2022). The role of notification scheduling in improving patient outcomes. *International Journal of Science and Research Archive*. Retrieved from https://ijsra.net/content/role-notification-scheduling-improving-patient

[29] Seenivasan, D., & Vaithianathan, M. (2023). Real-Time Adaptation: Change Data Capture in Modern Computer Architecture. *ESP International Journal of Advancements in Computational Technology (ESP-IJACT)*, *1*(2), 49-61.

[30] Singh, V. (2023). Large language models in visual question answering: Leveraging LLMs to interpret complex questions and generate accurate answers based on visual input. International Journal of Advanced Engineering and Technology (IJAET), 5(S2). https://romanpub.com/resources/Vol%205%20%2C%20No%20S2%20-%2012.pdf

[31] Thodi, A. K. (2020). Developing a MongoDB Monitoring System using NoSQL Databases for Monitored Data Management.

[32] Van Ginkel, J. R., Linting, M., Rippe, R. C., & Van Der Voort, A. (2020). Rebutting existing misconceptions about multiple imputation as a method for handling missing data. *Journal of personality assessment*, *102*(3), 297-308.

[33] Volminger, A. (2021). A comparison of Data Stores for the Online Feature Store Component: A comparison between NDB and Aerospike.

[34] Vu, V. A., & Walker, B. (2020, October). On the latency of multipath-QUIC in real-time applications. In *2020 16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (pp. 1-7). IEEE.

[35] Wang, H. (2023). Research on the application of Random Forest-based feature selection algorithm in data mining experiments. *International Journal of Advanced Computer Science and Applications*, *14*(10).

[36] Wang, Y., Arya, K., Kogias, M., Vanga, M., Bhandari, A., Yadwadkar, N. J., ... & Bianchini, R. (2021, April). Smartharvest: Harvesting idle cpus safely and efficiently in the cloud. In *Proceedings of the Sixteenth European Conference on Computer Systems* (pp. 1-16).

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited