Int. J. Inf. Cybersec.-2023

Enhancing Database Security: A Machine Learning Approach to Anomaly Detection in NoSQL Systems

Jatin Pal Singh

Abstract

This paper explores the integration of machine learning, specifically recurrent neural networks (RNNs), for automated anomaly detection within NoSQL databases. It addresses the challenges of maintaining data accuracy and security in these flexible and dynamic systems, where traditional methods often fall short. The proposed model features a distributed time series database, edge computing agents, a central data management backbone, and an RNN tailored for anomaly detection. Historical data is incorporated for context, and MQTT protocol ensures efficient communication. The model is evaluated using real-world data, demonstrating its potential to detect anomalies effectively. Furthermore, the paper investigates the impact of replication degree (k) on the performance and scalability of VoltDB, a NewSQL database, using the TPC-C benchmark. Results reveal that increasing k can enhance fault tolerance without significantly sacrificing throughput or latency. This work contributes to the understanding of anomaly detection in NoSQL databases and the trade-offs between consistency and performance in distributed database systems.

Keywords: *Anomaly Detection, Database Security, Machine Learning, NoSQL Databases, Recurrent Neural Networks (RNNs)*

I INTRODUCTION

The use of NoSQL databases has become increasingly popular in recent years due to their ability to handle large amounts of unstructured data [1]. One of the key benefits of NoSQL databases is their flexibility in terms of data schema, which allows them to handle data that does not fit neatly into a traditional relational database [2]. This makes them particularly useful for storing data that is semi-structured or has a variable schema, such as data from social media, IoT devices, or customer feedback [3], [4]. However, the use of NoSQL databases also presents some challenges, particularly in terms of data consistency and accuracy. Because NoSQL databases do not have the strict data typing and validation rules of traditional relational databases, it can be more difficult to ensure that data is accurate and consistent. This can have serious consequences for businesses that rely on this data to make decisions.

MongoDB, a prominent NoSQL database provider, experiences security incidents that highlights the potential risks associated with the use of NoSQL databases [5]. The breach was detected when an unauthorized third party used a phishing attack to gain access to some of MongoDB's corporate applications used for customer support services. The compromised system contained customer account metadata and contact information, including fields such as names, phone numbers, email addresses, and system logs for one customer. MongoDB has removed the unauthorized third party from their corporate applications, and the incident is considered contained. This is one of many incidents are happening around the world. Database anomaly detection has become a crucial factor because of this both for the service providers as well as the clients. The scale of these database system and many more factors are making it increasingly hard to maintain a robust security and real time anomaly detection quickly.

This incident highlights the need for improved anomaly detection techniques in NoSQL databases. As databases continue to scale to massive sizes and handle more diverse data types, maintaining data accuracy and security becomes increasingly challenging through manual processes alone. Traditional rule-based and statistical anomaly detection methods may struggle to keep pace with the flexibility and complexity of modern NoSQL systems [6]. Machine learning offers a promising approach by leveraging patterns in vast amounts of historical and real-time data. Techniques like supervised learning can identify anomalies based on features extracted from both schema-based and schema-less data [7]. Unsupervised methods like clustering and isolation forests can detect outliers without labeled examples. In this paper, we evaluate the application of machine learning for automated anomaly detection directly in NoSQL databases. Our goal is to develop solutions that can operate with minimal configuration and continue learning from new data with minimal human oversight. This could empower database administrators and security teams to more rapidly detect and address a wider range of threats, including previously unknown attack patterns. It may also reduce costs by shifting resource-intensive monitoring tasks from people to automated systems.

The existing literature on anomaly detection in NoSQL databases has primarily focused on traditional rule-based and statistical methods [8-11]. While these approaches provide valuable insights, they may struggle to keep pace with the dynamic and diverse nature of data stored in modern NoSQL systems. Additionally, there is a limited exploration of the integration of machine learning, particularly recurrent neural networks (RNNs), directly within NoSQL databases for automated anomaly detection. Moreover, the evaluation of database performance and scalability, especially in the context of varying replication degrees ((k)), is essential for understanding the trade-offs between consistency and performance. The current research landscape lacks a comprehensive study that combines machine learning-based anomaly detection with an in-depth analysis of distributed database performance under different replication scenarios.

This work proposes a novel approach by integrating machine learning, specifically recurrent neural networks (RNNs), directly within NoSQL databases for automated anomaly detection. The model is designed to adapt to the flexible and dynamic nature of data stored in NoSQL systems, providing a more robust and scalable solution compared to traditional methods.

II RELATED WORKS

Various approaches have been explored to enhance data availability and address performance anomalies. Bayou [12] relies on replication for data availability but lacks mechanisms for handling performance anomalies. Emerging cloud databases like VoltDB and MongoDB [13], [14] leverage enhanced main memory data structures for high data availability, but face challenges from cloud performance anomalies, including network, disk, and main memory malfunctions. Cake [15] focuses on scheduling for data availability but doesn’t identify faulty VMs. Eriksson et al. [16] offer a routing framework to tackle network failures, and Tejo’s anomaly detection alerts complement their work. In the domain of statistical learning for anomaly detection, Gujrati et al. [10] present various prediction models and anomaly detection approaches, each with its strengths and limitations. This research collectively contributes to the understanding of distributed database resilience and the challenges associated with performance anomalies in cloud environments. In this work, the authors extended a previous supervised learning model to detect multiple classes of anomalies based on SLO metrics. The results with Tejo demonstrate that the choice of the learning algorithm and features contributes to enhancing predictive efficiency in detecting performance anomalies. The research addresses the need for effective anomaly detection in distributed databases, especially in the context of cloud environments, where performance anomalies can significantly impact database performance.

III NOSQL DATABASE AND ANOMALY DETECTION

A NoSQL over traditional SQL

NoSQL databases are a type of non-relational database that offers a more flexible and scalable way of storing and retrieving data, particularly for handling large amounts of unstructured or semi-structured data [17]. Unlike traditional relational databases, NoSQL databases do not use a table-based relational model, and instead, they are designed to handle the specific requirements of modern applications that require high availability and scalability. NoSQL databases can be broadly classified into several categories based on their data model, including key-value, document-oriented, graph, and column-family databases. Each of these models has its own strengths and weaknesses, and the choice of which one to use depends on the specific requirements of the application.

Key-value databases, for example, are designed to store and retrieve data based on a unique key value. They are simple and efficient but may not be suitable for complex queries. Document-oriented databases, on the other hand, store data in the form of documents, such as JSON or XML, which can be more flexible than key-value databases. Graph databases are designed to store and query data in the form of nodes and edges, which makes them ideal for applications that involve complex relationships between data. Column-family databases are similar to relational databases but offer more flexibility in terms of data schema. NoSQL databases are horizontally scalable, meaning that they can handle increasing amounts of data and user traffic by adding more machines to the cluster, rather than relying on a single, powerful machine. This makes them well-suited for applications that require high availability and scalability, such as social media, online shopping, and real-time analytics. In addition, NoSQL databases are designed to handle large volumes of data and can be optimized for distributed processing, which allows them to handle massive volumes of data more efficiently. This makes them particularly useful for processing computer network traffic, where they can handle large volumes of data and provide efficient querying and analysis of network activity. Overall, NoSQL databases offer a powerful solution for handling large amounts of data and provide a more flexible and scalable way of storing and retrieving data than traditional relational databases. Their ability to handle unstructured or semi-structured data, scale horizontally, and provide efficient distributed processing make them an attractive option for modern applications that require high availability and scalability. The graphical representa-

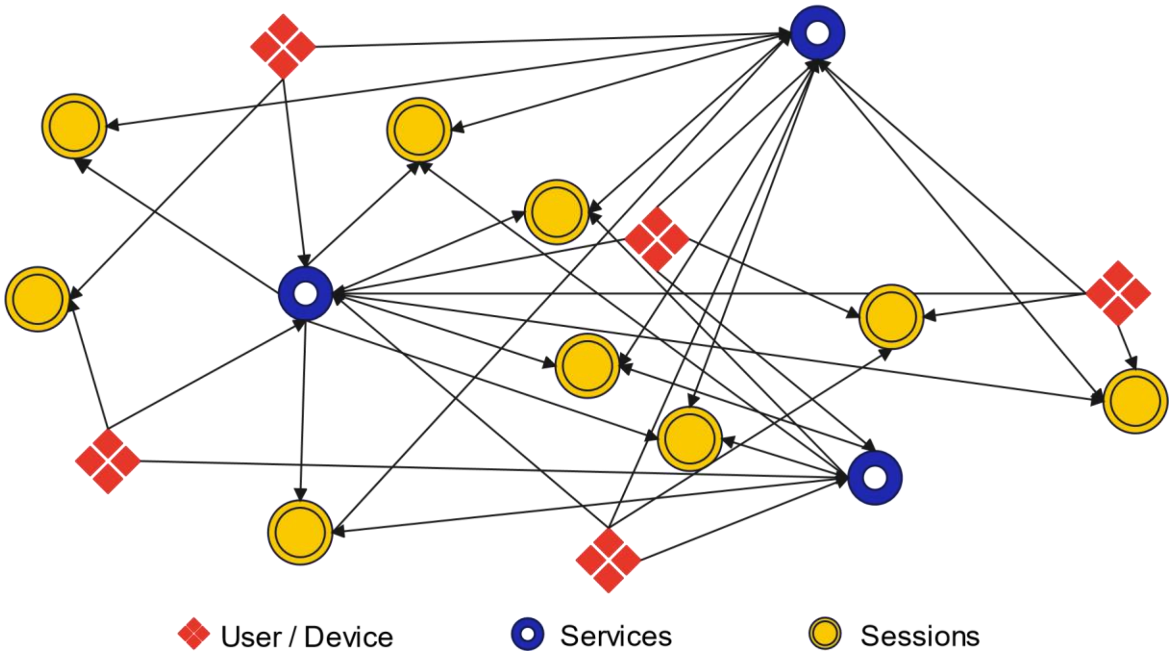
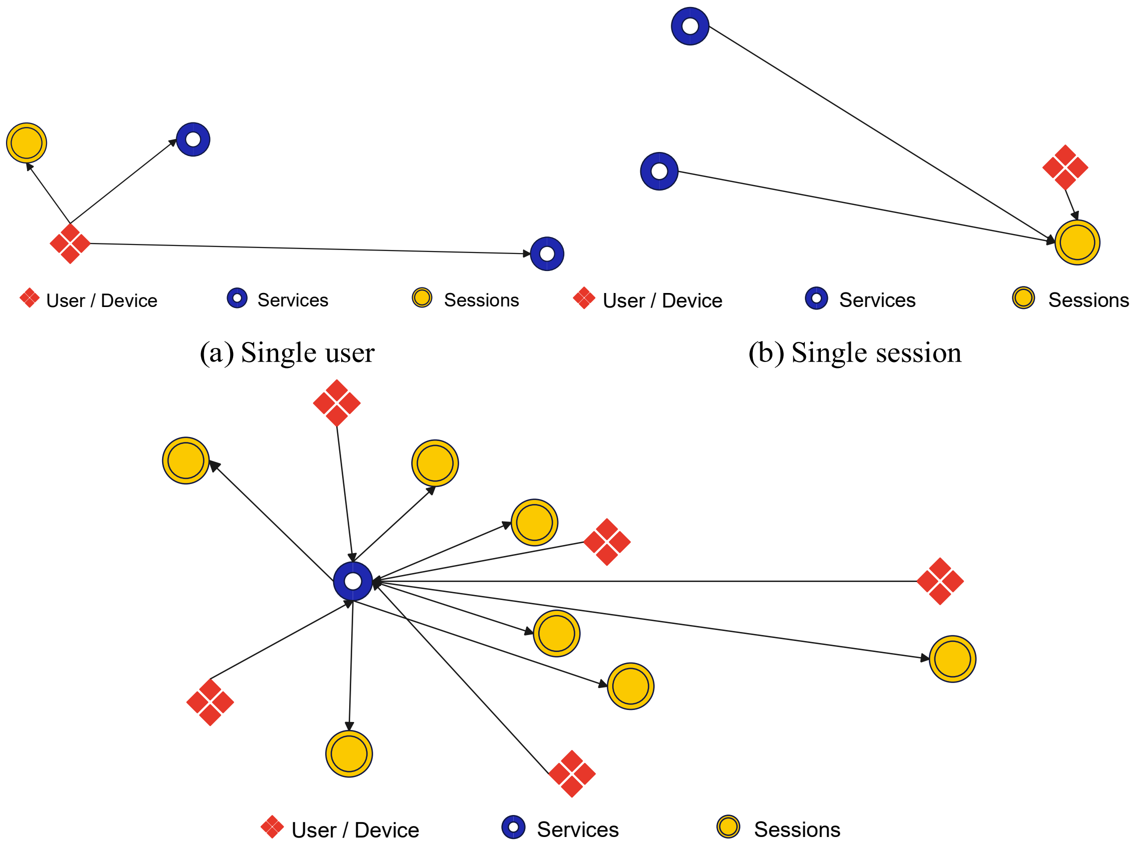


Fig. 1 Relationship between user, services and sessions in a live database

tion in Fig. 1 depicts the relationships among users, services, and sessions in databases. The users/devices are represented by red crosses, and each user/device is connected to multiple sessions and services, indicating that a single user can access multiple services and have multiple sessions. The services are represented by blue circles and are highly connected to both users/devices and sessions, showing that services can be accessed by many users and can have many concurrent sessions. The sessions are represented by yellow circles and have connections with both users/devices and services, demonstrating that each session is associated with a specific user/device accessing a particular service. The diagram illustrates a complex network of interactions between users/devices, services, and sessions in databases, highlighting the flexibility and complexity of user interactions with database driven applications/services. when a user uses a particular service that creats a session, or more than one service. Eventually, a users session starts as soon as they login or access the database. A depiction of the breakdown of primary units in a database, specifically focusing



(a) Single user

(b) Single session

(c) Single service

Fig. 2 Breakdown of primary units in a Database

on the connections between users/devices, services, and sessions are given in Fig. 2. Single User Block In this context, when a user utilizes a service it initiates a session. However, its also possible for sessions to be registered without immediately using any specific service as indicated by your statement. This could mean that in some systems or applications, initiating or opening the application might automatically start a session which then awaits for specific service requests or interactions from the user. Single Sessions are typically registered by services as shown in Fig. 2b. They act as intermediaries allowing communication between users/devices and services ensuring data consistency and security among other roles. Each session can be unique to ensure personalized experiences or data isolation depending on system requirements. Single Service - Fig. 2c illustrates that single service can have multiple communication channels with various users/devices leading to registration of different sessions. This highlights scalability where one service handles requests from multiple sources efficiently ensuring availability and performance.

B Anomaly detection in NoSQL

Detecting anomalies, those deviations from the expected within a dataset, forms a crucial line of defense for securing and understanding our data. In the diverse landscape of NoSQL databases, identifying these anomalies presents unique challenges due to the inherent flexibility and lack of rigid structure. Here, we focus into three key categories of anomalies, exploring their specific characteristics and how they manifest in NoSQL environments.

a Point Anomalies

Imagine a lone data point standing out starkly against the backdrop of its peers. This is the essence of a point anomaly, a single datum that significantly deviates from the established norm. In NoSQL databases, where diverse data formats intermingle, identifying such outliers can be tricky. A user login attempt from an unusual location, a sensor reading exceeding known thresholds, or a sudden spike in a specific data field âAS¸ these are potential˘ red flags demanding investigation. Statistical methods like outlier detection and z-scores serve as sentinels, constantly vigilant for these lone sentinels of the unexpected.

b Contextual Anomalies

While point anomalies jump out individually, contextual anomalies whisper their secrets through subtle shifts in the relationships and behaviors within the data. Consider an office building exhibiting high power consumption during the night âAS¸ a stark deviation from˘ expected patterns based on context. Similarly, unusual website traffic patterns or correlations between seemingly unrelated data points within NoSQL databases can signal anomalies hidden within the intricate dance of data. Techniques like time series analysis and clustering algorithms become crucial interpreters, deciphering these subtle deviations from established patterns to unveil hidden anomalies.

c Collective Anomalies

Not all anomalies stand alone. Collective anomalies emerge as groups of interrelated data points, collectively deviating from normal behavior. Imagine a coordinated cyberattack, where multiple sensors report abnormal readings simultaneously. Or, in NoSQL databases, a surge of unsuccessful login attempts followed by a successful one âAS¸ a pattern hinting˘ at brute-force attacks. Statistical analysis, correlation analysis, and network traffic analysis become instrumental in identifying these groups of interrelated anomalies, revealing the power of the pack in orchestrating malicious activities.

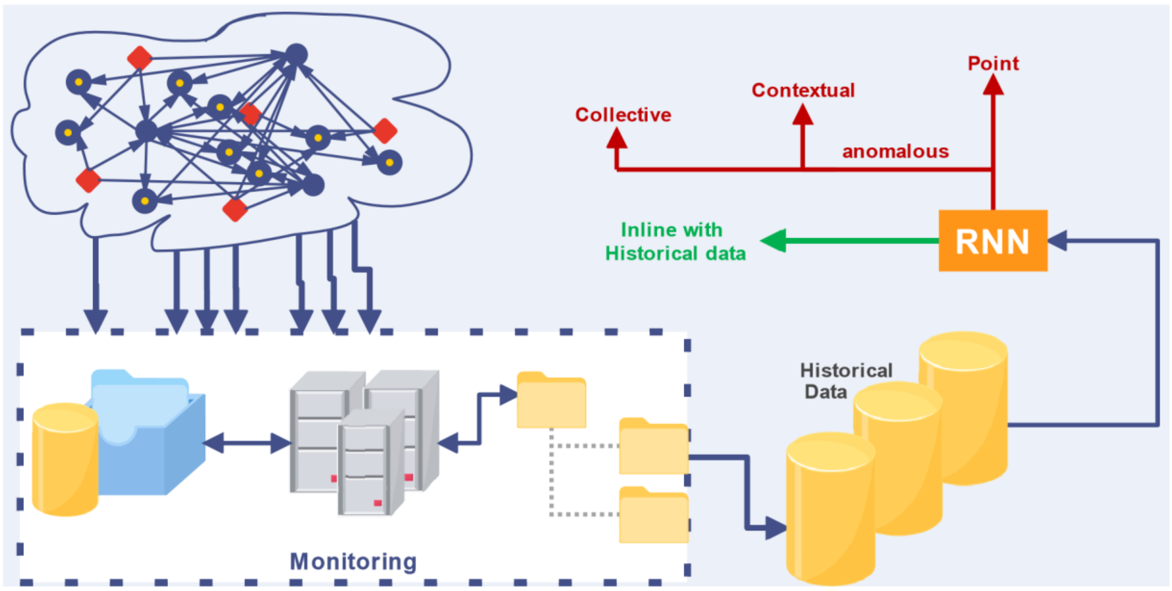


Fig. 3 Proposed model

IV PROPOSED RNN BASED MODEL

The presented figure Fig. 3 outlines a comprehensive model designed for the monitoring and analysis of data, with a focal point on the integration of a Recurrent Neural Network (RNN).

The visual representation unfolds as follows:

A Monitoring Database and Real-Time Processing

Connected to this database is a monitoring system, depicted by grey servers. This interconnection signifies the seamless flow of data from the database to the monitoring system. The monitoring system, in turn, is responsible for the real-time collection and processing of the incoming data. Our focus lies in real-time monitoring of power consumption, performance metrics, and utilization of computing nodes. The proposed model draws inspiration from established practices in distributed systems, incorporating key components and concepts that optimize data collection and analysis.

a Distributed Time Series Database

Our model relies on a distributed time series database, strategically built on Apache Cassandra, a renowned NoSQL database recognized for its scalability and high availability. This database architecture is meticulously designed to efficiently store and manage time-stamped data points, making it particularly applicable to scenarios demanding the handling of large volumes of time-series data. This component serves as the cornerstone for our dataset monitoring system.

b Edge Computing with Embedded Measuring Boards

To facilitate real-time monitoring at the source of data generation, our framework embraces edge computing principles. Agents are deployed on edge computing nodes, each equipped with embedded measuring boards. These agents are designed to monitor a spectrum of parameters, including power consumption, performance metrics, and utilization. This approach proves invaluable in scenarios where centralizing all monitoring processes is impractical or resource-intensive.

c Monitoring Agents and Hardware Sensors

Agents running on computing nodes play a pivotal role in our proposed model. They are entrusted with the responsibility of monitoring key metrics, employing a combination of software commands and hardware sensors. This comprehensive approach ensures accurate measurements and provides deep insights into the behavior of computing nodes. The detailed monitoring of hardware-related metrics is particularly critical for performance optimization and anomaly detection within the dataset.

d Data Management Backbone and MQTT Protocol

Measured values obtained from monitoring agents are seamlessly transmitted to a central data management backbone. This communication is facilitated by the MQTT (MQ Telemetry Transport) protocol, chosen for its suitability in low bandwidth, high latency networks with minimal resource demands. The MQTT protocol emerges as a crucial element in our architecture, ensuring efficient and reliable communication between distributed components. This aligns seamlessly with applications where real-time data transfer with minimal network resources is of paramount importance.

B Comparison with Historical Data

The monitored data, having undergone real-time processing, is subsequently directed towards historical data repositories, represented by yellow folders. This step involves a comparison between the current dataset and historical data to discern patterns or anomalies. The integration of historical data enriches the analysis, providing context for identifying deviations from established patterns. The monitored data, continuously collected in real-time, undergoes processing to extract relevant information and insights. This processing step can involve various algorithms, statistical techniques, or machine learning models depending on the nature of the data and the analysis goals. The processed real-time data is then compared with historical data repositories. This comparison aims to discern patterns or anomalies in the data. The integration of historical data enriches the analysis by providing context for identifying deviations from established patterns, helping in the detection of anomalies. Anomalies may manifest as unexpected spikes, drops, or shifts in the data compared to historical norms.

C RNN Model

In our study, we employed recurrent neural networks (RNNs) for each virtual machine within the infrastructure, as detailed in Section 4.3. This choice was motivated by the finding that dedicated models tailored for specific nodes outperformed a generic, universally applied model. The training data for each RNN was sourced from node-related information collected by our monitoring infrastructure. During the training phase, only data corresponding to the normal state was utilized. The dataset comprises a diverse range of metrics, including core load, frequency, temperature, virtual machine power consumption, room temperature, GPU usage, and more. These metrics, also referred to as features, constituted the input set for the RNNs. Each virtual machine had a dedicated training set corresponding to two months of normal behavior, obtained in collaboration with system administrators. Due to storage constraints, fine-grained monitoring data was retained for no more than a week. Therefore, for training purposes, we employed coarse-grained data, aggregated in five-minute intervals. Following the collection of raw data, a preprocessing step was implemented, which involved tasks such as removing data corresponding to periods when the monitoring system malfunctioned and normalization. This preparatory phase took approximately 30 seconds per virtual machine. The final set of features amounted to 166. A consistent network topology was adopted for each virtual machine, employing an RNN model for sequence-based learning. We chose this approach after empirical evaluation, considering its suitability for capturing temporal dependencies within the data. The network comprised three layers: an input layer, an output layer, and a hidden layer with recurrent connections. We used Rectified Linear Units (ReLU) as the activation function for neurons.

The input and output layers contained as many neurons as the number of features (166), while the hidden layer was appropriately sized to capture temporal patterns. Each RNN was trained with data specific to its corresponding virtual machine, utilizing available computational resources. The training employed the Adam algorithm, with the mean absolute error as the target loss. After preliminary experiments, a batch size of 32 and 100 epochs were chosen. The training duration for each RNN was approximately 20 seconds. It is noteworthy that training times and overhead were not considered critical concerns, given that the training phase occurred infrequently (once every few months) and could be scheduled during maintenance periods. The Keras framework with TensorFlow as a backend facilitated the design and training of the RNNs.

V RESULT ANALYSIS

We present a detailed analysis of our comprehensive evaluation of VoltDB (v4.x) as a NewSQL database, with a specific focus on investigating the impact of varying the replication degree (*k*). Our evaluation centered around the widely recognized TPC-

C benchmark, tailored for assessing Online Transaction Processing (OLTP) workloads. Some of the key indexes te evaluate the study is given in Table 1.

A Anomaly detection key indexes

The Table 1, a list of key indexes for anomaly detection in network behavior and memory and storage metrics. These indexes are important for identifying unusual patterns in network traffic and system resource usage that may indicate malicious activity or other types of anomalies. The table is organized into two categories: network behavior and memory and storage. Within each category, the table lists specific metrics that are commonly used to detect anomalies, along with a brief description of each metric. By monitoring these metrics and identifying deviations from expected behavior, security teams can quickly identify and respond to potential security threats.

Table 1 Network Behavior and Memory and Storage Metrics

Network Behavior	Memory and Storage
TCP Syn Recv Category: Connection Establishment	IO Writes Category: Data Storage
TCP MD5 Unexpected Category: Security	Memory Mapped Category: Memory Management
TCP SACK (Selective Acknowledgment) Category: Network Acknowledgment	Memory Free Category: Memory Management
TCP SACK Shifted Category: Network Acknowledgment	
TCP Resets Category: Connection Termination	
TCP Connection Active Category: Connection State	

TCP PAWS (Protect Against Wrapped Sequence numbers) Category: Network Security

This section provides a thorough examination of the results obtained from our experiments, shedding light on VoltDB's performance and scalability in different replication scenarios [14], [18].

B Impact of Replication Degree (k)

To assess the performance and scalability of VoltDB, we systematically varied the replication degree (*k*) across our experiments. The replication degree is a crucial parameter that influences the fault tolerance and consistency of distributed databases.

a Throughput

The throughput of the system was measured under different replication degrees. Figure 4 illustrates the impact of *k* on the overall throughput. The result shows the impact of repli-

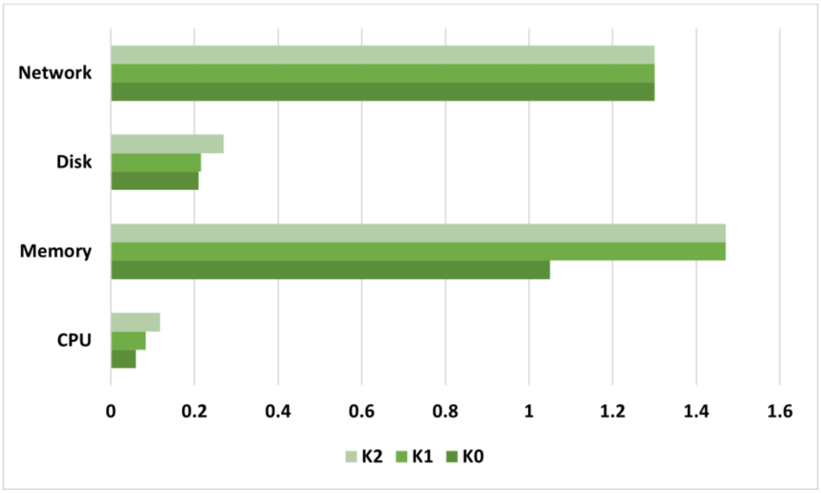


Fig. 4 Impact of Replication Degree (*k*) on Throughput

cation degree (k) on the throughput of VoltDB, a distributed database system that supports high-performance transactions. The result is based on an experiment where the replication degree was varied from 1 to 3, and the throughput was measured in transactions per second. The result indicates that increasing the replication degree reduces the throughput of VoltDB, but not significantly. This is because higher replication degree provides stronger consistency and fault tolerance, but also introduces more overhead for writing and propagating data to multiple replicas. The result also shows that the throughput reduction is more noticeable when k is doubled from 1 to 2, than when it is increased from 2 to 3. This suggests that the marginal impact of replication degree diminishes with higher k. The result also demonstrates that VoltDB can achieve very high throughput even with high replication degree. For example, at k=3, VoltDB can handle over 100K transactions per second, which is impressive for a distributed database system. This implies that VoltDB can balance performance and reliability effectively, and can support applications that require both high speed and high availability.

b Latency

Latency is a critical metric in OLTP systems. Table 2 provides insights into the effect of *k* on transaction latency. Latency varied across nodes, with Node33 exhibiting a higher 97th percentile (0.95) but lower 99th percentile (0.93), suggesting potential spikes in response times. Fig. 5 illustrates the impact of replication degree (*k*) on transaction latency, generally showing a trend of latency reduction with higher *k*, aligning with expectations of improved fault tolerance and consistency. However, the trade-off between consistency and performance is evident, as higher replication may impact latency, but not uniformly across nodes. The Table 2 Percentile Results for Different Nodes

Node	97th Percentile	99th Percentile
node10	0.88	0.91
node12	0.90	0.92
node27	0.91	0.95
node33	0.95	0.93
node40	0.90	0.89
node41	0.87	0.83

choice of replication degree should therefore carefully consider application needs, potentially tailoring *k* configurations for different nodes to balance performance and consistency requirements.

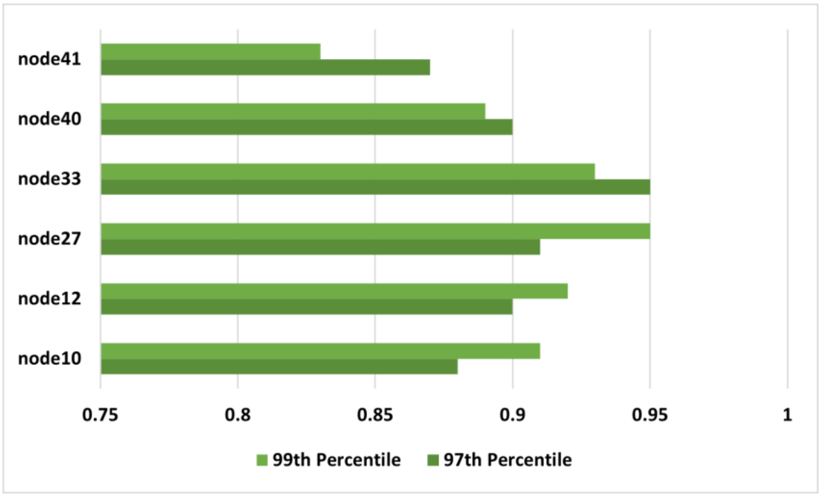


Fig. 5 Impact of Replication Degree (*k*) on Latency

VI CONCLUSION

The increasing prevalence of NoSQL databases, while offering unparalleled flexibility for handling diverse and unstructured data, also presents challenges in terms of data consistency and security. The security incident at MongoDB underscores the importance of robust anomaly detection techniques for maintaining data accuracy and safeguarding against potential threats. In this paper, we explored the application of machine learning, particularly recurrent neural networks (RNNs), for automated anomaly detection directly within NoSQL databases. Our proposed model, incorporating distributed time series databases, edge computing, and historical data comparison, leverages the strengths of NoSQL databases while addressing the evolving landscape of security threats. The RNNbased approach, tailored for specific virtual machines, demonstrated superior performance in capturing temporal dependencies within the data. The evaluation of VoltDB's performance and scalability, focusing on the impact of replication degree (*k*), revealed insightful results. The trade-off between consistency and performance was evident, with higher replication degrees influencing throughput and latency. However, even at high replication degrees, VoltDB demonstrated impressive throughput, balancing performance and reliability effectively.

While our study has made improvement in the integration of machine learning for anomaly detection in NoSQL databases and understanding the performance nuances of distributed databases like VoltDB, several avenues for future exploration and improvement exist. Future research should explore deeper into refining anomaly detection models, exploring advanced techniques beyond recurrent neural networks (RNNs). Investigating the applicability of deep learning architectures, ensemble methods, or hybrid models could contribute to more robust and accurate anomaly detection systems. Additionally, research efforts should focus on adapting models to dynamic changes in data schema, a common feature in NoSQL databases. Exploring dynamic replication strategies in distributed databases is a promising avenue. Adaptive replication mechanisms that adjust the replication degree (k) based on workload characteristics, system performance, and anomaly detection results could optimize both consistency and performance. This dynamic approach can offer an intelligent trade-off, providing higher replication during critical periods and scaling down during periods of lower demand.

REFERENCES

[1] C. Sauvanaud, M. Kaâniche, K. Kanoun, K. Lazri, and G. Da Silva Silvestre, “Anomaly detection and diagnosis for cloud services: Practical experiments and lessons learned,” *Journal of Systems and Software*, vol. 139, pp. 84–106, May 2018.

[2] T. Halabi, M. Bellaiche, and A. Abusitta, “Online Allocation of Cloud Resources Based on Security Satisfaction,” in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, Aug. 2018, pp. 379–384.

[3] A. Aldribi, I. Traore, and B. Moa, “Data Sources and Datasets for Cloud Intrusion Detection Modeling and Evaluation,” in *Cloud Computing for Optimization: Foundations, Applications, and Challenges*, ser. Studies in Big Data, B. S. P. Mishra, H. Das, S. Dehuri, and A. K. Jagadev, Eds. Cham: Springer International Publishing, 2018, pp. 333–366.

[4] M. Mowbray, S. Pearson, and Y. Shen, "Enhancing privacy in cloud computing via policy-based obfuscation," *The Journal of Supercomputing*, vol. 61, no. 2, pp. 267– 291, Aug. 2012.

[5] P. Team, "MongoDB data breach: Customer data stolen in cyber attack," Jan. 2023. [Online].

[6] T. Halabi, M. Bellaiche, and A. Abusitta, "Toward Secure Resource Allocation in Mobile Cloud Computing: A Matching Game," in *2019 International Conference on Computing, Networking and Communications (ICNC)*, Feb. 2019, pp. 370–374.

[7] S. V. G. Subrahmanya, D. K. Shetty, V. Patil, B. M. Z. Hameed, R. Paul, K. Smriti, N. Naik, and B. K. Somani, "The role of data science in healthcare advancements: Applications, benefits, and future prospects," *Irish Journal of Medical Science (1971 -)*, vol. 191, no. 4, pp. 1473–1483, Aug. 2022.

[8] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009.

[9] R. A. Ariyaluran Habeeb, F. Nasaruddin, A. Gani, I. A. Targio Hashem, E. Ahmed, and M. Imran, "Real-time big data processing for anomaly detection: A Survey," *International Journal of Information Management*, vol. 45, pp. 289–307, Apr. 2019.

[10] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula, "Intrusion detection techniques in cloud environment: A survey," *Journal of Network and Computer Applications*, vol. 77, pp. 18–47, 2017. [Online]. Available: https://www.sciencedirect. com/science/article/pii/S1084804516302417

[11] A. Praseed and P. S. Thilagam, "DDoS Attacks at the Application Layer: Challenges and Research Perspectives for Safeguarding Web Applications," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 661–685, 2019.

[12] K. Petersen, M. Spreitzer, D. Terry, and M. Theimer, "Bayou: Replicated database services for world-wide applications," in *Proceedings of the 7th Workshop on ACM SIGOPS European Workshop: Systems Support for Worldwide Applications*, ser. EW 7. New York, NY, USA: Association for Computing Machinery, Sep. 1996, pp. 275– 280.

[13] M. Stonebraker, “SQL databases v. NoSQL databases,” *Communications of the ACM*, vol. 53, no. 4, pp. 10–11, Apr. 2010.

[14] “GitHub - VoltDB/voltdb: Volt Active Data.” [Online]. Available: https: //github.com/VoltDB/voltdb/tree/master

[15] A. Wang, S. Venkataraman, S. Alspaugh, R. Katz, and I. Stoica, “Cake: Enabling highlevel SLOs on shared storage systems,” in *Proceedings of the Third ACM Symposium on Cloud Computing*. San Jose California: ACM, Oct. 2012, pp. 1–14.

[16] B. Eriksson, R. Durairajan, and P. Barford, “RiskRoute: A framework for mitigating network outage threats,” in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*. Santa Barbara California USA: ACM, Dec. 2013, pp. 405–416.

[17] I. Kotenko, I. Saenko, and A. Branitskiy, “Machine Learning and Big Data Processing for Cybersecurity Data Analysis,” in *Data Science in Cybersecurity and Cyberthreat Intelligence*, L. F. Sikos and K.-K. R. Choo, Eds. Cham: Springer International Publishing, 2020, vol. 177, pp. 61–85.

[18] “TPC-C Homepage.” [Online]. Available: https://www.tpc.org/tpcc/