International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Navigating Key-Value Stores in Cloud Edge Computing and IoT

Shyam Burkule[1], Ramprasad Chinthekindi[2], Senthilbharanidhar Boganavijayakumar[3]

[3]Email: *shyam.burkule[at]gmail.com*
[3]Email: *ramprasad.ch[at]gmail.com*
[3]Email: *bvsenthil[at]gmail.com*

Abstract: ***Embedded key-value stores play a crucial role in enabling efficient data management in Internet of Things(IoT) and edge computing contexts, where local processing and storage capabilities are vital. The primary objective of this study is to investigate the management of key-value stores in cloud edge computing and the IoT. The study utilized the Systematic Literature Review technique. This study conducted a thorough analysis of various embedded key-value stores, such as RocksDB, LevelDB, and LMDB, to assess their applicability and performance in limited contexts. In addition, the study examined optimization tactics specifically designed for embedded systems, including compression algorithms, wear levelling approaches for non-volatile memory, and energy-efficient data operations that are crucial for battery-powered devices. The results demonstrated clear trade-offs between performance and resource utilization among the examined key-value stores. This provides valuable guidance for developers and system architects when choosing and optimizing embedded databases based on specific application needs and environmental limitations. This study enhanced the field by offering a comprehensive analysis of the performance characteristics of embedded key-value stores in IoT and edge environments. This analysis lays the foundation for developing data management solutions that are optimized, resilient, and resource conscious. Consequently, in future, it could help in creating more intelligent and efficient IoT ecosystems.***

Keywords: IoT; Key Value Stores; Cloud Edge Computing; Data Storage; Data Management

1. Introduction

Key-value stores are crucial in the realm of cloud edge computing and IoT systems, since they provide efficient mechanisms for storing and retrieving data. Successfully navigating these businesses requires a comprehensive grasp of their underlying ideas. Key-value stores function as NoSQL databases, where data is structured into key-value pairs, similar to dictionaries in computer languages [1]. Creating a proficient data model is crucial, customizing it to meet the precise needs of the application while taking into account scalability and performance. This entails the selection of suitable keys that are distinct and readily recognizable, typically representing device identifiers or sensor categories in the context of the IoT [2]. The implementation of security measures, such as encryption and access control, effectively protect sensitive data during transmission and storage [1]. Monitoring instruments are utilized to monitor essential metrics and guarantee optimal performance and dependability, while proactive management tactics aid in preserving system health. In the end, the choice of the appropriate key-value store, such as Redis, Amazon DynamoDB, Apache Cassandra [3], or others, depends on factors like scalability, performance, consistency, and ease of management [4]. By implementing these methods, the process of accessing key-value stores in cloud edge computing and IoT contexts becomes more efficient and organized, enabling smooth data storage, retrieval, and management. A representation of the key-value storage in the cloud computing system is shown in the figure below.

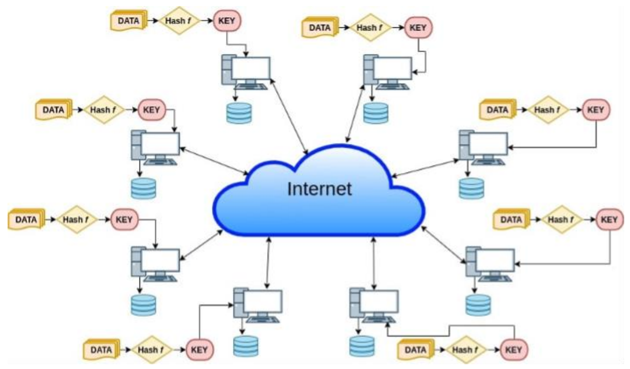

Figure 1: An overview of the use of Key – value storage in cloud environment [5]

A. Scope of the study

This study explores the use of key-value stores in cloud edge computing and IoT, with a focus on effective ways for storing and retrieving data. The objective is to determine the most effective methods for navigating these systems, while considering scalability, performance, and security issues in distributed and resource-limited situations.

B. Research objectives

1) To conduct a comprehensive analysis of several prominent embedded key-value stores and examining their suitability and performance within these constrained environments.
2) To explore optimization methodologies for embedded systems and energy-efficient data operations for battery operated devices.
3) To examine the integration of these key-value stores with other IoT components to provide a holistic view of their deployment in an end-to end system.

C. Research Questions

1) What are the comparative performance characteristics of prominent embedded key-value stores in constrained environments, considering factors such as memory

footprint, read/write latency, and throughput under varying workload conditions?
2) How can optimization methodologies be effectively applied to enhance energy efficiency in data operations for battery-operated devices utilizing embedded key-value stores?
3) What are the trade-offs between energy consumption and performance in different optimization strategies?
4) What are the challenges and opportunities in integrating embedded key-value stores with other IoT components to create end-to-end systems?
5) How does this integration impact factors such as data consistency, scalability, and fault tolerance in diverse IoT deployment scenarios?

2. Research Background

Efficient data management has become crucial in the fast changing realm of cloud-edge computing and IoT. Key-value stores have become a fundamental technique in this field, providing a simple yet powerful way to store data. Nevertheless, the ever-changing and widely spread characteristics of cloud-edge settings provide distinct obstacles in efficiently harnessing these resources [6]. This study intends to thoroughly examine the complexities of navigating key-value stores in such situations, including their implementation, optimization, and interaction with IoT components [7]. The research aims to clarify the intricacies of this relationship in order to improve the performance, scalability, and reliability of cloud-edgeIoT systems. Ultimately, this will expand the capabilities of modern computing infrastructures.

3. Methodology

A systematic literature review was conducted on the topic of navigating key-value stores in cloud edge computing and IoT using the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) approach.

To conduct a systematic literature review on the navigating key-value stores in cloud edge computing and IoT, this study used a well-designed search strategy to locate relevant literature. This method involved using search strings that consist of appropriate keywords and Boolean operators. Search strings consist of combinations such as Key-value stores* AND cloud-edge computing* AND IoT* OR Keyvalue stores* OR NoSQL* AND cloud-edge computing* OR edge computing* AND IoT* OR Internet of Things* OR Integration* OR deployment* AND key-value stores* OR NoSQL* AND cloud-edge computing* OR edge computing* AND IoT* OR Internet of Things* OR Scalability* AND performance* OR challenges* AND key-value stores* OR NoSQL* AND cloud-edge computing* OR edge computing* AND IoT* OR Internet of Things* OR Data consistency* OR reliability* AND key-value stores* OR NoSQL* AND cloud-edge computing* OR edge computing* AND IoT* OR Internet of Things. The search keywords are tailored to the individual databases and repositories being utilized, ensuring comprehensive coverage of pertinent material. Furthermore, the incorporation of synonyms and variations of keywords is employed to incorporate a diverse array of perspectives and methodologies encountered in the literature.

A. Inclusion/Exclusion Criteria

The table II presents the potential criteria for inclusion and exclusion in this systematic literature review on the investigation of navigating key-value stores in cloud edge computing and IoT.

The data was obtained by formulating and evaluating the search query in several reputable databases. After the references were imported into Endnote, they were filtered to eliminate any instances of duplication. As a result, a total of 120 papers were retained, all of which were entirely distinct from one another. Through a thorough examination of abstracts and titles, our study found publications that included the selected keywords. After that, the created references were moved to an Excel spreadsheet for further filtering and analysis. This spreadsheet included the names of the writers, the year the work was published, the title, and the abstract. The Excel spreadsheet has 81 research publications in total. All the paper abstracts were reviewed using Excel, with a focus on those that were directly connected to the objectives of the study. At first, 52 pieces in total were found; however, some of them had to be eliminated since they were classified as grey literature, books, or parts of books. Moreover, as some of the papers could not be downloaded, they were removed from distribution. The 20 papers that made up the final shortlist. This flow diagram (Figure below) illustrates the process of selecting articles for the SLR during a literature search and is based on the PRISMA Flow Diagram.

4. Results and Discussion

Efficiently handling data in dispersed situations is essential when navigating key-value stores in cloud edge computing and IoT. Key-value stores hold data in the form of key-value pairs, offering efficient retrieval and storing capabilities [7]. Efficient data management is essential for performance and scalability in cloud edge computing and IoT, where resources are restricted and data is generated at the edge. This involves implementing techniques for dividing data, duplicating it, and ensuring consistent updates between edge devices and cloud servers. Efficiently navigating key-value stores in these situations improves data retrieval, boosts system speed, and enables immediate decision-making in IoT applications [8].

A. Comprehensive Analysis of various embedded key-value stores

Embedded key-value stores such as RocksDB, LevelDB, and LMDB are crucial in cloud edge computing and IoT as they effectively handle data in contexts with limited resources [13]. The following section examine each of them in detail:
ROCKSDB: RocksDB, created by Facebook, is an integrated key-value store that is specifically designed to efficiently store and retrieve data on flash drives and RAM. This is built around

Table I: Relevant publications were identified from internet repositories in this study

Digital Library	URL
Scopus	https://www.scopus.com/sources.uri?zone=TopNavBar&amp;origin=searchbasic
Semantic Scholar	https://www.semanticscholar.org/
IEEE Xplore	https://ieeexplore.ieee.org/Xplore/home.jsp
Science Direct	https://www.sciencedirect.com/
Springer	https://link.springer.com/
Web of Science	https://wosjournal.com/
PubMed	https://pubmed.ncbi.nlm.nih.gov/

Table II: Inclusion/Exclusion Criteria

Criteria	Inclusion	Exclusion
Type of Study	Academic Research papers and Review articles	Editorials and opinions
Topic Relevance	Studies that investigate the navigating keyvalue stores	Studies unrelated to navigating key-value stores in cloud edge computing and IoT
Publication date	Studies published between 2018 – 2024	Studies beyond and after 2018-2024
Subject Area	Key – Store Values, Cloud Edge Computing, IoT, Data Storage	Irrelevant Subjects
Access Availability	Open access studies	Studies behind paywalls or lacking access
Peer – Review	Peer-reviewed studies	Non-peer-reviewed studies

Table III: Architectural traits, data handling capabilities, and resource efficiency to assess IoT task performance [9] [10] [11]

Characteristics/DB	ROCKSDB	LEVELDB	LMDB
Architectural Characteristics	Log-structured merge-tree (LSMtree) architecture optimizes SSDs and flash storage in RocksDB. It efficiently organizes small and large datasets into many levels of sorted files	The log-structured merge-tree (LSM-tree) in LevelDB is simpler and has less functionality than RocksDB. It sorts data into keyvalue pairs in files	B+ tree design and memorymapped I/O optimize data storage and retrieval in LMDB. High parallelism and low latency make it ideal for IoT applications
Data Handling Capabilities	Supports several data types, transactions, snapshots, and secondary indexes. It allows workload-based performance tweaking with customizable configuration.	supports basic key-value operations and sequential readings efficiently. It lacks Rocks DB's advanced capabilities like transactions and secondary indexes.	ACID transactions, cursor-based traversal, and multi-threaded access are supported. Simple and reliable, it performs well.
Resource Efficiency	optimised for memory and I/O efficiency. Write amplification is reduced and compression is available, saving storage space. It may use a lot of memory, especially during compression.	its memory-efficient and lightweight design makes it suited for resource-constrained IoT devices. Its simplicity reduces memory expense compared to RocksDB, however large write workloads may impair performance.	resource-efficient, with low memory overhead and disk space consumption. It reduces disk I/O operations with memory-mapped I/O, boosting performance and resource consumption.
Handling of IoT specific Workloads	IoT workloads needing high write throughput and complicated data processing work well with RocksDB. Secondary indexes and quick compaction make it appropriate for multiple data formats in IoT applications.	Simple IoT workloads with readheavy access patterns may benefit from LevelDB. Lightweight and efficient sequential read performance make it suited for edge devices with limited resources.	IoT applications requiring low latency access, high concurrency, and ACID transactions benefit from LMDB. It's ideal for mission critical IoT applications that require dependability and performance.

[12]

Google's LevelDB; however, it has been enhanced to take use of the latest hardware advancements [14]. RocksDB is well suited for cloud edge situations because it optimizes resource use and delivers exceptional performance. It is especially suitable for situations that require efficient and fast data access, which is often the case in Internet of Things (IoT) applications. It demonstrates exceptional performance in both reading and writing operations, making it well-suited for use cases that include frequent changes and queries of data. The software's optimal utilization of memory and its capability to utilize SSDs and flash drives enhance its performance in contexts with limited resources [9]. LEVELDB: LevelDB, a key-value store developed by Google, is designed to be embedded and is highly optimized for simplicity, exceptional performance, and dependability. The program is implemented in the C++ programming language and offers a structured arrangement of string keys and their corresponding string values. It is particularly suitable for cloud edge scenarios where simplicity and reliability are crucial prerequisites. Due to its lightweight nature and ease of integration, it is well-suited for edge devices that have limited resources. It provides excellent performance for tasks that involve reading a lot of data and following a sequential order. Nevertheless, the performance of the system may deteriorate when subjected to heavy write loads because it

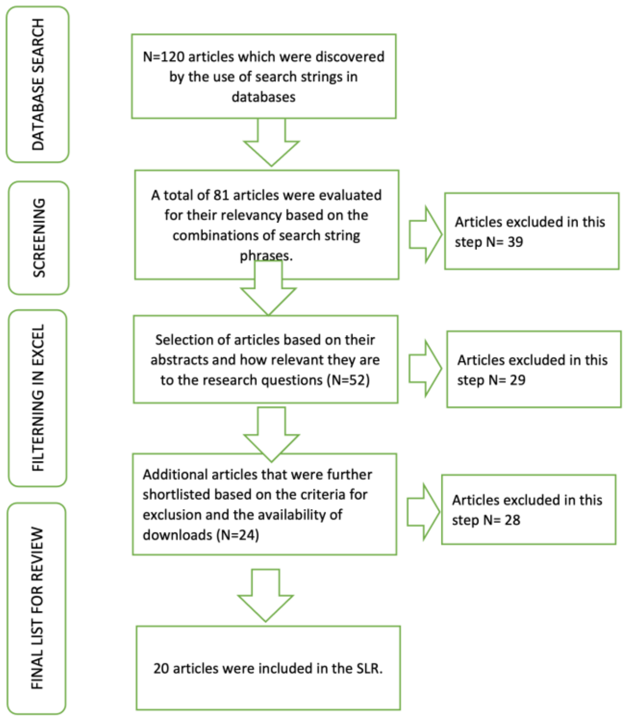
DATABASE SEARCH
N=120 articles which were discovered by the use of search strings in databases
SCREENING
A total of 81 articles were evaluated for their relevancy based on the combinations of search string phrases.
Articles excluded in this step N= 39
FILTERNING IN EXCEL
Selection of articles based on their abstracts and how relevant they are to the research questions (N=52)
Articles excluded in this step N= 29
FINAL LIST FOR REVIEW
Additional articles that were further shortlisted based on the criteria for exclusion and the availability of downloads (N=24)
Articles excluded in this step N= 28
20 articles were included in the SLR.

Figure 2: Literature search for SLR publications (based on PRISMA flow diagram).

relies on single-threaded compactions. However, for numerous IoT applications, its performance is sufficient, particularly when combined with effective data management tactics [10]. LMDB (Lightning Memory-Mapped Database): LMDB is a compact key-value store specifically developed for efficient and lightweight data storage with excellent performance. It is renowned for its straightforwardness, dependability, and effective utilization of memory. Its small overhead and efficient memory-mapped architecture make it particularly suitable for cloud edge environments and IoT devices. It is especially suitable for situations where low response time and fast data transfer rate are crucial. It provides exceptional performance for both reading and writing activities. The memory-mapped design of the system reduces disk input/output and utilizes the caching capabilities of the operating system to achieve optimal performance. LMDB has stable performance even when subjected to high workloads, making it well-suited for mission-critical IoT applications [15]. The table III evaluates the architectural features, data management capabilities, and resource efficiency of several platforms to determine their usefulness in managing workloads specific to the Internet of Things (IoT).

The performance of databases such as RocksDB, LevelDB, and LMDB is thoroughly evaluated across multiple crucial aspects using a comprehensive assessment that combines synthetic benchmarks with real-world IoT simulation scenarios [16]. Read/Write Throughput: Synthetic workloads are specifically created to imitate the usual read and write operations observed in IoT applications. Throughput measures quantify the speed at which the databases can process these operations. Simulated IoT scenarios encompass situations in which IoT devices generate, process, and store data. Throughput is assessed in situations that closely resemble real-world deployment environments. Latency: Latency benchmarks replicate real-time interactions with the database, quantifying the duration required for a request to be executed and completed. Latency is assessed in simulated IoT environments to accurately represent the delays encountered by devices when transferring data and receiving responses from the database. Storage Footprint: IoT application storage footprints are determined by data volume, kind, and retention regulations. Power Consumption: To understand how database activities affect IoT device energy consumption, especially battery-powered ones, simulated IoT scenarios are used.

B. Optimization techniques

Specific optimization solutions for embedded systems, particularly for IoT devices powered by batteries, are essential for maximizing resource efficiency and extending the lifespan of the devices [11] [17]. Here are a few of essential optimization strategies: Compression algorithms: Minimize the storage and transmission requirements of data in order to save energy. Employ lightweight compression methods like LZ4, Snappy, or zlib to compress data prior to storing or transmitting it. These algorithms provide an optimal trade-off between compression ratio and computational complexity, rendering them wellsuited for embedded systems with limited resources. Wear Levelling for Non – Volatile Memory: Prolong the durability of flash memory by uniformly dispersing write and erase operations across memory blocks. Incorporate wear-levelling methods into the program or utilize hardware features offered by the flash memory controller. The techniques employed to assure uniform usage of memory cells and prevent premature wear-out of flash memory include dynamic wear levelling, static wear levelling, and wear-aware data insertion. Energy efficient Data – Operations: Optimize energy efficiency in data processing, storage, and transmission to enhance battery longevity. Batch processing involves consolidating several data activities into batches in order to minimize CPU wake-ups and idle time, resulting in energy conservation. During idle periods, low-power modes will be employed to conserve power consumption by utilizing low-power sleep modes. Utilize dynamic voltage and frequency scaling (DVFS) approaches to modify CPU performance according to the requirements of the workload.

C. Integration of key-value stores with other IoT components

By incorporating key-value stores like RocksDB, LevelDB, and LMDB with other IoT components like data analytics engines and security protocols, a full and unified ecosystem for IoT deployments may be established. These integrations allow for a comprehensive understanding of the entire system by providing smooth data transfer, effective data analysis, and strong security measures [18]. Key-value stores are utilized in this integrated arrangement as the fundamental data layer, effectively storing and managing data supplied by IoT devices.

They function as a storage facility for unprocessed sensor data, event records, and contextual details. Data analytics engines are connected with key-value stores to do analytics in real-time or in batches, deriving important insights and patterns from the stored data. This facilitates well-informed decision-making, proactive maintenance, and optimization of performance. Furthermore, the use of security standards guarantees the secrecy, consistency, and accessibility of IoT data. Key-value stores employ encryption technologies and access control regulations to ensure security [19]. Additionally, secure communication protocols such as MQTT with TLS/SSL encryption safeguard the transport of data between IoT devices and the storage layer. Intrusion detection systems and security information and event management solutions bolster the security stance of the system, facilitating immediate identification and response to potential threats. By integrating key-value stores with data analytics engines and security standards, IoT installations achieve enhanced operational efficiency, scalability, and flexibility. This comprehensive method simplifies the management of data, speeds up the processing of analytics, and enhances security measures, finally offering a complete perspective of the entire IoT system deployment.

5. Conclusion

To summarize, the integration of key-value stores with data analytics engines and security protocols establishes a strong and effective environment for IoT deployments. This integration facilitates smooth data management, provides valuable insights, and enhances security measures. This comprehensive approach guarantees that the performance, scalability, and integrity of the entire IoT system are improved. It enables informed decision-making and proactive mana-gement of IoT assets.

References

[1] M. Szalay, P. Matray, and L. Toka, “Annabelladb: Key-value store made cloud native,” in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–5.
[2] S. Idreos and M. Callaghan, “Key-value storage engines,” in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 2667–2672.
[3] R. Chinthekindi, S. Burkule, and A. K. Chintakindi, “A comparative analysis of popular distributed key-value stores,” *International Journal of Science and Research (IJSR)*, vol. 13, pp. 1730–1734, 2024. [Online]. Available: \url{https://www.ijsr.net/getabstract.php?paperid= SR24426081530}
[4] K. Tulkinbekov and D.-H. Kim, “Casedb: Lightweight key-value store for edge computing environment,” *IEEE Access*, vol. 8, pp. 149775– 149786, 2020.
[5] G. Arora. Build your own decentralised large scale key-value cloud storage. [Online]. Available: \url{https://www.opensourceforu.com/2020/ 11/build-your-own-decentralised-large-scale-key-value-cloud-storage/}
[6] B. Confais, A. Lebre, and B. Parrein, “Performance analysis of object store systems in a fog and edge computing infrastructure,” *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXIII*, pp. 40–79, 2017.
[7] X. Lu, D. Shankar, and D. K. Panda, “Scalable and distributed key-value store-based data management using rdma-memcached.” *IEEE Data Eng. Bull.*, vol. 40, no. 1, pp. 50–61, 2017.
[8] K. Venkatram and M. A. Geetha, “Review on big data & analytics– concepts, philosophy, process and applications,” *Cybernetics and Information Technologies*, vol. 17, no. 2, pp. 3–27, 2017.
[9] Z. Cao, S. Dong, S. Vemuri, and D. H. Du, “Characterizing, modeling, and benchmarking {RocksDB}{Key-Value} workloads at facebook,” in *18th USENIX Conference on File and Storage Technologies (FAST 20)*, 2020, pp. 209–223.
[10] K. Tulkinbekov and D.-H. Kim, “Casedb: Lightweight key-value store for edge computing environment,” *IEEE Access*, vol. 8, pp. 149775– 149786, 2020.
[11] N. Cunningham, “A further performance comparison of operations in the file system and in embedded key-value databases,” 2024.
[12] C. Chen, W. Zhong, and X. Wu, “Building an efficient key-value store in a flexible address space,” in *Proceedings of the Seventeenth European Conference on Computer Systems*, 2022, pp. 51–68.
[13] F. Liang, C. Qian, W. G. Hatcher, and W. Yu, “Search engine for the internet of things: Lessons from web search, vision, and opportunities,” *IEEE Access*, vol. 7, pp. 104673–104691, 2019.
[14] S. Dong, A. Kryczka, Y. Jin, and M. Stumm, “Rocksdb: Evolution of development priorities in a key-value store serving large-scale applications,” *ACM Transactions on Storage (TOS)*, vol. 17, no. 4, pp. 1–32, 2021.
[15] M. Yin, H. Zhang, R. van Renesse, and E. G. Sirer, “Cedrusdb: Persistent key-value store with memory-mapped lazy-trie,” *arXiv preprint arXiv:2005.13762*, 2020.
[16] K. I. Duwe, “Coupling storage systems for efficient management of self-describing data formats,” 2023.
[17] Y.-S. Chang, Y.-F. Chen, and H.-S. Ko, “Weakly durable highperformance transactions,” *arXiv preprint arXiv:2110.01465*, 2021.
[18] Z. Shen, F. Chen, Y. Jia, and Z. Shao, “Didacache: an integration of device and application for flash-based key-value caching,” *ACM Transactions on Storage (TOS)*, vol. 14, no. 3, pp. 1–32, 2018.
[19] H. Huang and S. Ghandeharizadeh, “Nova-lsm: A distributed, component-based lsm-tree key-value store,” in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 749–763.