electronics

Article

Challenges in NoSQL-Based Distributed Data Storage: A Systematic Literature Review

Shabana Ramzan [1,*], Imran Sarwar Bajwa [2], Rafaqut Kazmi [2] and Amna [2]

1 Department of Computer Science, Govt. Sadiq College Women University, Bahawalpur, Punjab 54890, Pakistan
2 Department of Computer Science and IT, The Islamia University of Bahawalpur, Bahawalpur, Punjab 54890, Pakistan; imran.sarwar@iub.edu.pk (I.S.B.); rafaqutkazmi@gmail.com (R.K.); amnace39@gmail.com (A.)
* Correspondence: shabana@gscwu.edu.pk; Tel.: +92-30-27766829

Received: 30 March 2019; Accepted: 23 April 2019; Published: 30 April 2019

Abstract: Key-Value stores (KVSs) are the most flexible and simplest model of NoSQL databases, which have become highly popular over the last few years due to their salient features such as availability, portability, reliability, and low operational cost. From the perspective of software engineering, the chief obstacle for KVSs is to achieve software quality attributes (consistency, throughput, latency, security, performance, load balancing, and query processing) to ensure quality. The presented research is a Systematic Literature Review (SLR) to find the state-of-the-art research in the KVS domain, and through doing so determine the major challenges and solutions. This work reviews the 45 papers between 2010–2018 that were found to be closely relevant to our study area. The results show that performance is addressed in 31% of the studies, consistency is addressed in 20% of the studies, latency and throughput are addressed in 16% of the studies, query processing is addressed in 13% of studies, security is addressed in 11% of the studies, and load balancing is addressed in 9% of the studies. Different models are used for execution. The indexing technique was used in 20% of the studies, the hashing technique was used in 13% of the studies, the caching and security techniques were used together in 9% of the studies, the batching technique was used in 5% of the studies, the encoding techniques and Paxos technique were used together in 4% of the studies, and 36% of the studies used other techniques. This systematic review will enable researchers to design key-value stores as efficient storage. Regarding future collaborations, trust and privacy are the quality attributes that can be addressed; KVS is an emerging facet due to its widespread popularity, opening the way to deploy it with proper protection.

Keywords: key-value store; NoSQL; systematic literature review; relational database

1. Introduction

In recent years, the generation of massive data applications requires new data storage techniques other than traditional databases. Relational databases and non-relational databases are different generations for data storage. They are different from each other with regards to data storage, their building method, and the type of information that they store. Extensive research has compared relational and NoSQL databases. There is no survey focusing on the quality attributes of NoSQL databases. From the software engineering perspective, software engineers and architects need a reference for selecting the right NoSQL database regarding quality attributes. The quality attributes of databases are the main concern, whether developing an enterprise system or big data applications. This systematic literature review fills this gap by clearly identifying the techniques that can be employed to achieve a particular quality attribute in NoSQL key-value store (KVS).

Relational databases are traditional databases that are based on the relational model that deals with structured data only, as developed by IBM in 1970 [1]. The Relational Database Management System (RDBMS) is software that is used to maintain relational databases. Relational databases use query language called Structured Query Language (SQL). SQL commands are used for data gathering as well as for interactive queries to manipulate databases. They organize data in relations consisting of rows and columns, where each row represents a unique instance, and each column has specific information. Rows are also called tuples or records, and columns are also called fields or attributes. Each table has a unique key, which is called a primary key, and every row is accessed using this primary key. The tables are linked to each other through a foreign key, which is a key that is a primary key of another table. Relational databases are rigid, as their schema is pre-defined and unchangeable. The schema defines the relationship of tables and column types. Relational databases are ACID (Atomicity, Consistency, Isolation and Durability) compliant. Relational databases only provide vertical scalability, i.e., they only allow the vertical growth of a data structure by adding new records at run-time, but no horizontal scalability, i.e., they do not allow the horizontal growth of a data structure by adding fields at run-time.

The non-relational databases are NoSQL databases. The term NoSQL was introduced by Strozzi Carlo in 1998 [2] as the name given to his relational database solution that does not use SQL. The idea of NoSQL was redefined in 2009 and became the competitor of relational databases, as they have dynamic schema or no schema, and provide horizontal scalability. The NoSQL databases are the best choice for cloud computing and big data, as they require horizontal scalability. NoSQL databases present themselves as a substitute of relational databases, because they are scalable and able to manage this massive amount of data efficiently. NoSQL databases have become the backbones of big enterprises and web services such as Google, Twitter, Facebook and Amazon, due to their high availability, efficient performance, and linear scalability [3].

NoSQL databases have no structured query language interface [4]: the syntax of queries varies from database to database. NoSQL databases lie in the spectrum between ACID and BASE (Basically Available, Soft state, and Eventually consistent) [5]. The NoSQL databases can distribute and replicate data over multiple servers [6]. NoSQL databases are the only databases that allow data distribution and persistency across different computing nodes [7]. NoSQL databases are categorized based on their storage type, key-value store, column store, document store, and graph store [6–8]. Initially, the term NoSQL stood for "No SQL", but it has recently been redefined as "Not only SQL", meaning that the system complements the relational database management system [9,10]. Relational and non-relational databases have their own data models, as shown in Figure 1.

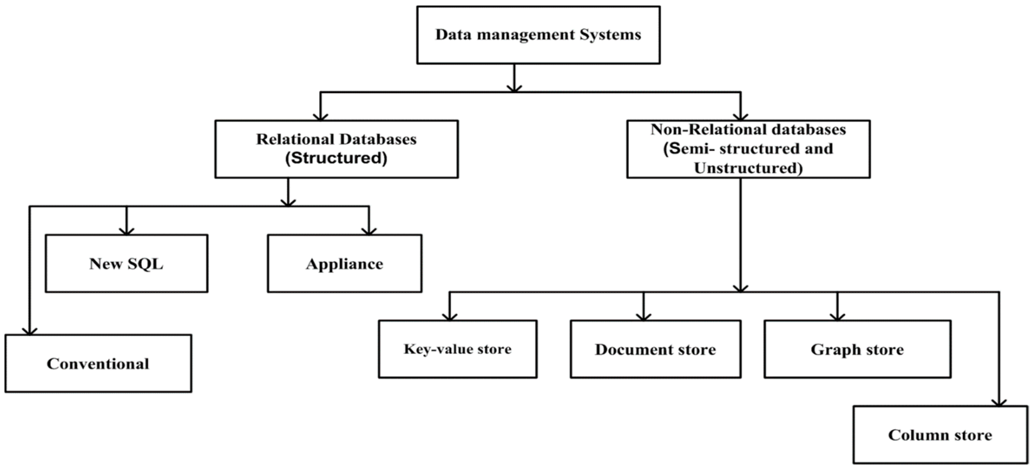


Figure 1. Data collection and storage.

Relational databases are further categorized as Conventional, NewSQL, and Appliance, etc. [11]. "Conventional database (relational, network, hierarchical) consist of records of many different record types (database looks like a collection of files)", and NewSQL is a relational database that offers the same scalability as NoSQL databases but maintains the ACID properties [11].

Non-relational databases (NoSQL) are document oriented; they can store structured, semi-structured, and unstructured data. The simplest data model of NoSQL is KVS, where each key has one value. A document store is an extended KVS, which stores more complex data compared to KVS. Column store represents the data in tabular form containing rows and columns. The foundation of the graph store is graph theory, and graph stores are used to store graph-oriented datasets in an efficient manner.

1.1. NoSQL Databases

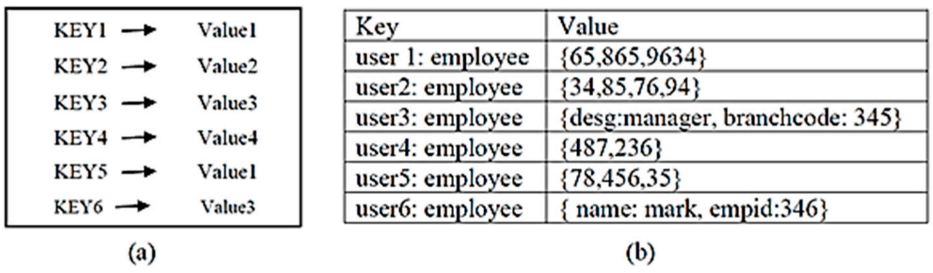
NoSQL databases that are designed to provide services such as massive data storage, handling heterogenous data, supporting horizontal scaling, and the parallel processing of data across several servers. In recent times, the need of data storage has changed due to evolutions of big data and cloud computing. The term "big data" referred to the exponential growth of data (structured, semi-structured, and unstructured) that demands the new infrastructure of storage [12,13]. Big data introduced challenges of data capturing and data storage. However, handling big data is difficult using relational databases. Relational databases require high proficiency and a lot of effort to accommodate the storage requirements of big data. Simultaneously, cloud applications impose the need of a high-performance and a scalable storage system. Relational databases are unsuccessful in handling this massive growth of semi-structured and unstructured data. The database research community is inclined to establish scalable and distributed solutions for the management of the gigantic volumes of data [14]. NoSQL has emerged as a revolutionary solution that provides the horizontal scaling, replicating, and partitioning of data over many servers. It meets the needs of the current application, as it allows number of simple read/write operations per unit time.

NoSQL can support a variety of activities such as non mission-critical OLTP (Online Transaction Processing), ETL (Extract, Transform, Load)-style data transformation, exploratory and predictive analytics [3], etc. Each NoSQL store is specialized according to the needs of current applications [15]. NoSQL databases are inherently available, and scalable databases such as Amazon's Dynamo (KVS) guarantees its availability for almost 99.9% of the year [16]. They also have the ability to add new records dynamically. The NoSQL database can be used in a framework for streaming network analysis [17]. These databases provide the new mechanism to create and manipulate data stores. The most flexible data model of NoSQL databases is the KVS, which has no schema, and its data values are opaque. KVS has a limited set of operations and requires no joins, lock, or other operations while dealing with objects. The requirement to scale out (horizontal scalability) a large number of commodity machines and different data access patterns requires the KVSs [18,19].

1.2. Key-Value Store

KVS databases are currently the main area of research due to persistent growth in data. Cloud environments and the Web 2.0 service rely on KVSs because of its exceptional features and easy adoption [20]. These databases use entirely different mechanisms of data storage and retrieval compared to relational databases [8,21]. KVS manages the unstructured data as a value of arbitrary length, which is associated with the unique key [22]; its data model and data are shown in Figure 2. These are high-performance databases, since the values are stored as an opaque BLOB (Binary Large Object). The BLOB is stored as a single entity in a database that is a collection of binary data that does not need to index data.

The simplest model of NoSQL databases is KVS; it has a pair of unique key and data values, which is helpful to handle a huge data volume. A value is an arbitrary size, structure, and type (e.g., a document, a string, or an image). KVS databases are becoming popular because of their high-level features such as scalability, availability, and efficiency handling unstructured data [23]. The principle of these databases is similar to the distributed memory cached object in a caching system [24]. KVSs use ‘get’, ‘put’, and ‘delete’ commands to retrieve, store, and update data, as it has no query language. These databases provide a platform for distributed storage services [25]. KVS databases enhanced the universal communication technology due to their ability to handle the large amount of unstructured data quite efficiently [26]. KVS has no constraints and no foreign key. It is usually used for parallel lookups and intensive read operations. KVS is mainly based on key attributes that are well-suited for simple operations. It is usually used as a distributed KVS, which is schema-free, and it provides high availability. The most popular KVS databases are Redis, Membase, Level DB, Voldemort, and Berkeley DB.



Key	Value
user 1: employee	{65,865,9634}
user2: employee	{34,85,76,94}
user3: employee	{desg:manager, branchcode: 345}
user4: employee	{487,236}
user5: employee	{78,456,35}
user6: employee	{ name: mark, empid:346}

Figure 2. (**a**) Key-value store (KVS) data model, (**b**) Data in KVS.

The rapid growth of web data and cloud computing highlighted the problems of storage systems’ quality, which affects the user’s behavior and system design. KVS databases can handle such problems efficiently and better than RDBMSs (Rational Database Management Systems). On the other hand, a KVS database arises with multiple quality attributes that are needed such as scalability, consistency, throughput, latency, security, and efficient query processing. In the recent past, researchers have been working to deal with these challenges in key-value stores and trying hard to provide sustainable solutions. As it is a significant domain of research, this literature is heading upward. According to our knowledge, there are no systematic literature review studies on the key-value stores that thoroughly analyze and identify the quality attributes of key-value stores, as well as the challenges and solutions in the domain. To fill this gap, this SLR (Systematic Literature Review) has raised some research questions in order to systematically present the available literature to target these quality attributes. This study is based on journal papers, conference papers, workshops, and symposium articles as the relevant literature. This review includes the research work published between 2010–2018.

This paper is organized as follows. Section 2 discusses the research goals and methodology used to identify the relevant literature and synthesize the findings of the conducted research. Section 3 depicts the selected literature, categorizes it into different classes, and tests the applicability of the literature against the research goals. Section 4 demonstrates the research challenges and corresponding solutions in detail, and Section 5 presents the discussion. Section 6 describes the research biasness and limitations, and Section 7 concludes the review.

2. Related Work

In the available literature, there is no systematic literature review in the domain of key-value stores (KVSs). However, there are a few surveys available in the domain of NoSQL databases; nevertheless, there are no systematic literature reviews. However, a survey bases study disused six KVSs and described the important features of each store [27]. This study focused on the prominent features of the following key-value stores: Berkeley DB, Project-Voldemort, LevelDB, Dynamo, Hyperdex, and SILT. This survey enables the user to choose the best key-value store according to the needs of his or her application. There is another survey that discusses an up-to-date and concise comparison of NoSQL databases on the basis of quality attributes [21]. Since then, the software engineers and architects make design-oriented decisions and develop software using quality attributes, and this survey evaluates the NoSQL databases from the perspective of quality attributes. Katarina et al. carried out a survey that presented the comparison of NoSQL (non-relational databases) and NewSQL (relational databases) [28]. The authors reviewed these databases with respect to three perspectives: providing the details of database solutions, guidelines that assist the researchers to choose the right data store, and the challenges and future opportunities of the domain. The survey concluded the challenges in the domain of cloud environments that are using NoSQL and NewSQL, including the non-existence of a standard query language terminology, sparse comparison and benchmarking criteria, limited documentation, an occasional immaturity of solutions and lack of support, diversity, and inconsistency. Another survey reflects the NoSQL solutions with respect to four design principles: a consistency model, CAP (Consistency, Availability and Partition Tolerance) theorem, a data model, and data partitioning [23]. The available features, strategies, drawbacks, and strengths for each design aspect are discussed in detail. It also highlights the challenges that are faced in developing the most efficient NoSQL databases. This survey also provides an in-depth understanding of these databases that enables the architects and designers to successfully transform existing classical database to the NoSQL databases.

The survey of NoSQL databases [29] that discussed the current needs (Web 2.0 applications) and the causes of evolution of the new generation of databases is named NoSQL. This study focused on the shortcomings of relational databases and the challenges to meet the current needs. It is recommended in the study that it is necessary to select each NoSQL database according to its claim and client's system requirements. It is also reported that how NoSQL databases achieve the scalability and availability by compromising on ACID properties. These modern databases follow the CAP theorem, which states that any system can only achieve two out of three properties: partition tolerance, availability, and consistency, each data store use different techniques to achieve these properties. Similarly, a survey conducted on NoSQL databases focuses on the literature that covered all the NoSQL solutions [30]. It also discusses a decision tree-based approach that is very helpful in selecting a database according to the requirement of the application, whereas the leaf nodes of the tree presented the range of applications from simple caching of data. These surveys concluded [29,30] the limitations of RDBMS for the current era, merits, and demerits of NoSQL databases and the classification of these databases. The summary of related studies in the field of KVS and NoSQL databases is given in Table 1.

Table 1. Summary of related studies in non-relational Structured Query Language (NoSQL) databases.

Study	Study Reference	Study Objective	Publication Year
Survey	[27]	A Survey of Modern KVSs	2012
Survey	[21]	NoSQL Quality Attribute Evaluation	2015
Survey	[28]	NoSQL and NewSQL	2013
Survey	[23]	Challenges in developing NoSQL	2018
Survey	[29]	Evolution of NoSQL	2012
Survey	[30]	NoSQL Databases	2017

The study [29,30] reviewed the evolution of NoSQL databases and important NoSQL solutions in detail. However, this SLR-based study reviewed different and important aspects, challenges, and prospects of the NoSQL key-value store. This study targets the literature that provided the aid to software engineers for the right selection of key-value stores in identifying the techniques to achieve the particular quality attribute.

3. Systematic Literature Review (SLR)

Brereton et al. described the process of a systematic review study that has three main phases: review planning, conduction, and the documentation of results as conclusion [31], as shown in Figure 3. The first phase highlights the purpose of this review, which is described in Sections 1.1 and 1.2. The targeted research questions are presented in Section 3.1, and the process of selection of primary studies discussed in Section 3.4. Section 3.5 describes the inclusion and exclusion criterion. The detail of data extraction and synthesis is given in Section 3.6. Finally, the results of the study are documented in Section 4.

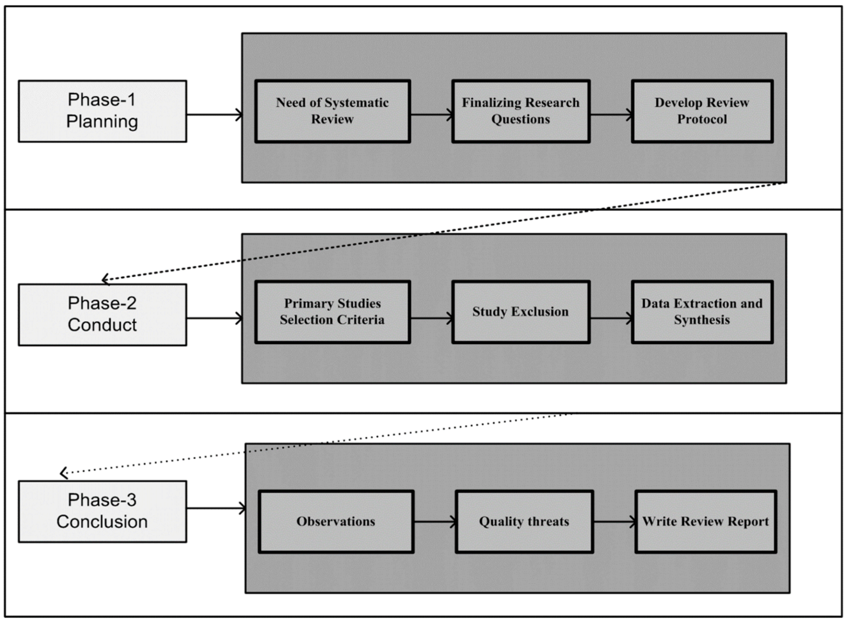


Figure 3. Systematic Literature Review (SLR) protocol.

3.1. Research Questions

The research questions are formulated after discussions with the proficient researchers and the authors in the domain. The objective of these research questions is to highlight the challenges in achieving the quality attributes of KVS-based systems and figure out the best techniques that address these challenges. This review will be helpful in synthesizing the existing research respecting a specific research question. The research questions are given in Table 2.

Table 2. Research questions and their bases.

No	Research Questions	Bases
RQ1	What are the main quality attribute challenges faced by key-value store, and what are their solutions?	This question determines the major challenges for key-value store and their solutions from the literature studied in this review.
RQ2	Which models and techniques are deployed for the execution and assessment of the presented solutions?	The main objective of this question is to play up the models and techniques applied by researches for the execution and testing of their proposed solutions.
RQ3	What is the reliability status of the presented approaches, and what is the quality status of included publications?	This question finds out the reliability and quality of the selected publications.
RQ4	What are the forums of publication and line of development in studies on the key-value store?	The main objective of this research question is to identify trends of publication and publication forums.

3.2. Searching Strategy

This is the initial phase of a typical SLR study that selects relevant literature and provides criteria to include and exclude works from an SLR. The selection of primary studies matters to the quality of the SLR results. Our searching strategy has three steps:

(1) Selection of data sources
(2) Identification of search terms
(3) Scrutiny of studies based on inclusion or exclusion criteria

3.3. Data Sources

We searched through all the major databases to gain a broad perspective. The database searching is more beneficial than searching through a limited set of journals and conference proceedings [32]. The completeness of the results of the review is based on the accurate selection of data sources and search strings. The result completeness is the factor that influences the SLR quality. Table 3 provides the list of the four databases that were searched.

The rationale to select these databases was the inclusion of the relevant journals, workshop proceedings, conferences, and symposiums within the domain of key-value stores. We excluded the grey literature (work in progress, workshop reports, and technical reports) that helped us select the quality research studies. The quality assessment of grey literature is very difficult. The searches through selected sources resulted in a repetition of studies that excluded through manual screening. This section provides the rationale to select databases for SLR studies, and the next section presents the process to identify the search terms.

Table 3. Data sources.

Data Origins	Web Sites
IEEE Xplore	http://ieeexplore.ieee.org/Xplore/
The ACM Digital Library	http://dl.acm.org/
Springer	http://www.springerlink.com/
Elsevier	http://www.elsevier.com/

3.4. Identification of Search Terms

We formulated the research keywords through the systematic method that follows the following steps:

- On the basis of selected terms that are derived from formulated research questions.
- Try alternative terms, if relevant data on selected terms is not found.
- Search through target databases with the use of Boolean operators; "OR" and "AND" and "NOT" link the selected terms.

Here, the main focus was to explore the studies that have key-value store-based storage systems that addresses the issues to achieve one of the attributes (consistency, scalability, performance, security, latency and throughput, query processing, load balancing). We selected the key terms that were used in most of the studies, so we avoided using these attributes as key terms. At the beginning, we selected the key terms from the available literature of the domain, and then we used these key terms to design the search query. Secondly, we selected key terms with the prior experience in the domain of NoSQL databases.

We used the search strings on all the selected databases, as shown in Figure 4. To get the most relevant studies, we used the terms: <KVS> and <NoSQL>. We used the OR operator to cover article titles, author keywords, and publication year of studies from 2010 to 2018.

When we use these queries to search through selected databases, there was repetition in the results that was manually screened out. Each database has its own search criteria, so we tried to refine our selection as needed. As a result, the databases returned the most relevant studies; otherwise, it was very difficult to perform analysis on the large number of returned studies. In the upcoming section, we define the inclusion and exclusion criteria to select primary studies from the returned results of selected databases.

3.5. Inclusion and Exclusion Criteria for Study Selection:

Previously, the KVS databases were used with many different objectives in different data-intensive applications. At the same time, these are also used for big data management and cloud computing, which may have different objectives other than the topic of the study. The focus of this SLR is to identify the challenges related to quality attributes, and investigate the techniques and methods that were proposed by the researchers to address these challenges. Thus, it is required to set some parameters for the selection of the most relevant studies. Here, the inclusion and exclusion criteria is defined and explained to select the most relevant primary studies, and the whole study selection process is also explained here as well. We searched through selected databases using our search query as mentioned below. Consequently, 1911 prospective studies were found from 2010 to 2018 after eliminating duplication. For selection of studies of this SLR, we followed a two-phase process as shown in Figure 4, and each phase has sub-phases. In the first phase, we follow the following criteria for the selection of study:

(1) Initially, studies were selected on the basis of title and abstracts.
(2) Selection on the basis of reading the abstract.
(3) Further selection based on reading the text of the article.

In the first phase, there were 211 selected studies; if there was confusion regarding the selection of any study, we brought this study into the second phase of selection. For the selection of studies in the second phase, we defined the inclusion/exclusion criteria. We thoroughly studied them one by one, and included/excluded the studies on the following criteria.

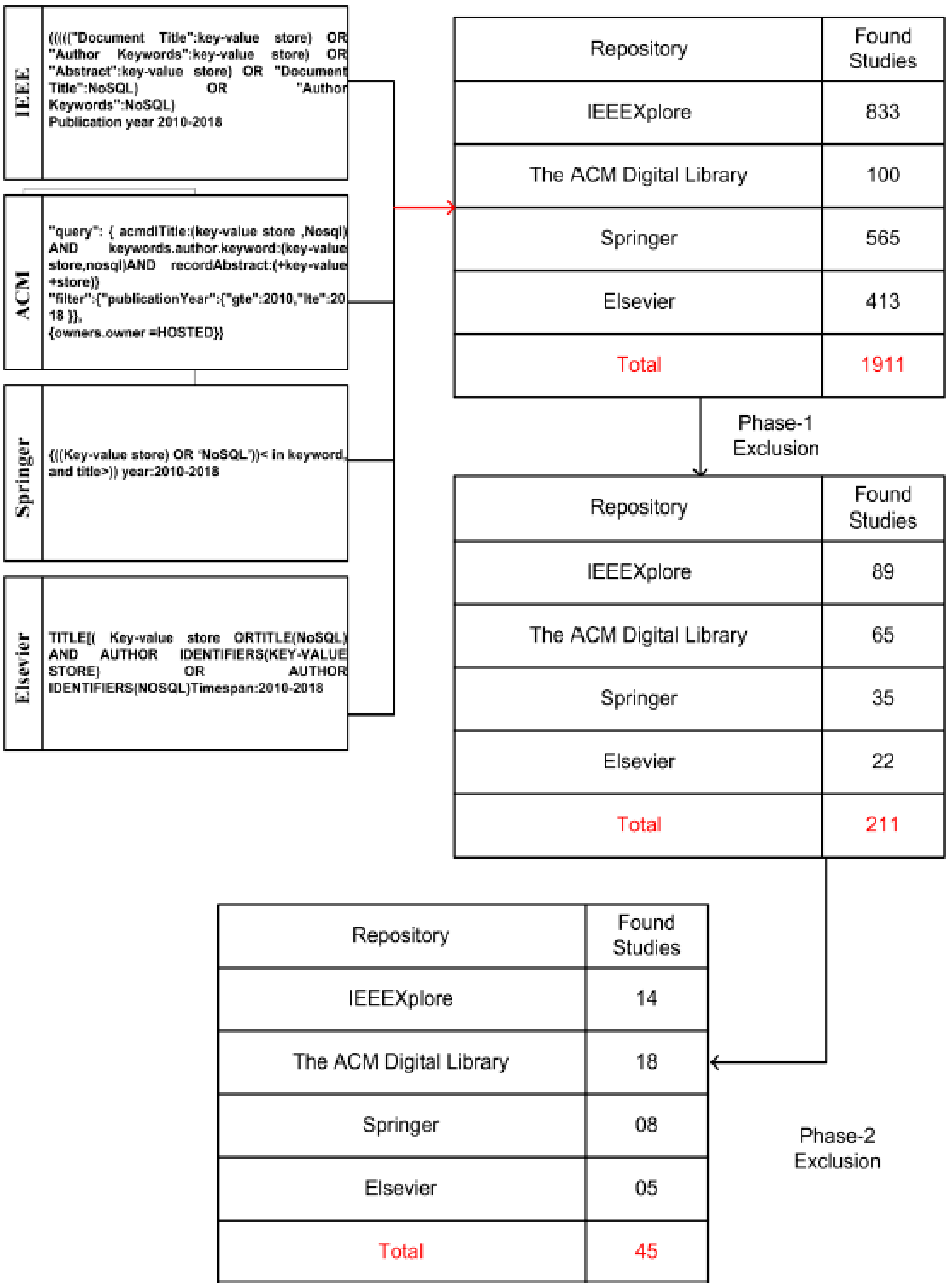
Only those studies that met the following inclusion criteria are included:

- Selected material that addresses the quality attribute challenges of the key-value store.
- All articles from peer-reviewed publication forums.
- Highly relevant papers that were published from 2010 to 2018.
- Research papers that are in the English language.

Studies were rejected that met the following exclusion criteria:

- Material that is not related to research questions.
- PhD dissertations, technical reports, posters, and studies that had less than five pages were excluded. The aim of this study is to extract data from peer-reviewed studies that have sufficient technical details.
- Articles that were not from peer-reviewed publication forums.
- Papers that do not have information related to the publication date, such as issue and volume number, were excluded.
- Research papers that were not in the English language.
- Duplicates were included; we selected the best one that was more recent, complete, and improved. The other was excluded.
- All articles that could answer at least one research question

In the second phase, we had confusion regarding whether to include a few studies or not in this SLR. That was a very important decision to be made, so we gathered opinions from other experts and researchers for making decisions on that consensus. The inclusion/exclusion criteria of the second phase refined our selection, and at the end, we were left with 45 studies.



IEEE:

```
((((("Document Title":key-value store) OR
"Author Keywords":key-value store) OR
"Abstract":key-value store) OR "Document
Title":NoSQL) OR "Author
Keywords":NoSQL)
Publication year 2010-2018
```

ACM:

```
"query": { acmdlTitle:(key-value store ,Nosql)
AND keywords.author.keyword:(key-value
store,nosql)AND recordAbstract:(+key-value
+store)}
"filter":{"publicationYear":{"gte":2010,"lte":20
18 }},
{owners.owner =HOSTED}}
```

Springer:

```
{((Key-value store) OR 'NoSQL'))< in keyword,
and title>)) year:2010-2018
```

Elsevier:

```
TITLE[( Key-value store ORTITLE(NoSQL)
AND AUTHOR IDENTIFIERS(KEY-VALUE
STORE) OR AUTHOR
IDENTIFIERS(NOSQL)Timespan:2010-2018
```

Repository	Found Studies
IEEEXplore	833
The ACM Digital Library	100
Springer	565
Elsevier	413
Total	1911

Repository	Found Studies
IEEEXplore	89
The ACM Digital Library	65
Springer	35
Elsevier	22
Total	211

Repository	Found Studies
IEEEXplore	14
The ACM Digital Library	18
Springer	08
Elsevier	05
Total	45

Figure 4. Study selection process.

3.6. Data Extraction

The data is collected on the basis of data collection terms, as shown in Table 4. Then, data is organized into different classes on the basis of the main concepts in the domain of this study (see Table 5). We presented the data integration regarding the quality attributes, research challenges of key-value stores, and the related results. The techniques and strategies used by various studies are listed in Table 6. The algorithms used by researchers to provide the solutions of challenges are also extracted. The algorithm-based solutions are organized in the form of a table, as shown in Table 7. We perform analysis in order to categorize the collected data. The classes were extracted from the reviewed literature on the basis of our research questions. If one study addressed more than one challenge, then we mentioned that study in only one class. Figure 5 shows the classification of reviewed literature on KVS.

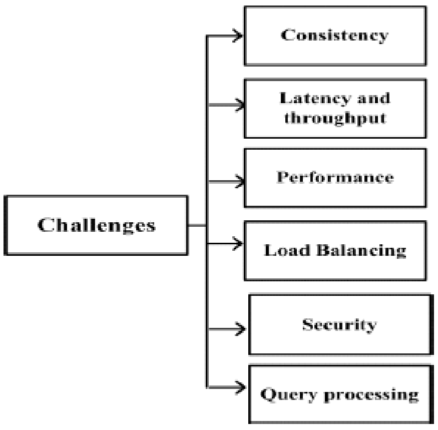


Figure 5. Classification of research on key-value store.

We did this classification on the basis of concepts of selected studies. Each class is formed with respect to the challenges and solutions to obtain quality attributes, as shown in Table 5. We discussed the challenges and corresponding solutions in detail. This review enables the researchers to have detail sight of the studied approaches.

Table 4. Terminology for data collection.

Term	Explanation
Study Area	The focus of the study.
Date of Review	Data collection date
Aims	Objective of the study
Impetus	Motivation for the study.
Algorithm	The algorithm presented in the study as a solution.
Problem	The issue addressed by the study.
Solutions	The approach presented to solve problem.
Improvements	Research directions for future work in the area of study.

4. Results and Discussions

This section presents the results of this SLR in the form of different categories that are classified from the reviewed studies, challenges, solutions, execution techniques, evaluation techniques, and studies' distribution with respect to publication forums and publication year.

4.1. RQ 1: What Are the Main Challenges Faced by Key-Value Store, and What Are Their Solutions?

The primary challenges that are associated with a KVS-based system to achieve the quality attribute are classified into six main classes. A few studies were relevant to many classes; to avoid replication, we discussed one study under one class. The studies that addressed the challenges and provided solutions are listed in Table 5.

4.1.1. Challenges:

Latency and Throughput: There are various performance metrics for a system, but in this class, we included the studies that present the solutions to achieve high throughput and low latency in KVS. To achieve the performance of a server side applications needs to have high-throughput persistent

KVS [33]. We have few studies in this class that addressed the challenge of achieving high throughput with low latency, a few studies addressed of achieving high throughput, and a few of them proposed solutions for low latency.

Consistency: The studies under this class provided different techniques and approaches to achieve consistency in KVS. In principle, eventual consistency is guaranteed by NoSQL KVS [34]. Generally, all the NoSQL databases are eventually consistent, but we have included different studies of KVS that provide strong consistency.

Performance: We include studies in this class that proposed different techniques and algorithms to achieve a high level of performance in KVS.

Scalability: NoSQL databases provide scalability; we include studies in this class that proposed approaches, techniques, and algorithms to enhance the scalability feature of KVS. Scalable systems can handle large workloads without affecting the performance of the system.

Security: We included the studies that were introducing different techniques, algorithms, and approaches to achieve the security of KVS. NoSQL databases are efficient storage for big data, but are lacking in security [35]. Secure systems are those systems that are protected from malicious or accidental actions.

Query Processing: We include the studies that proposed different approaches and techniques to do query processing. KVS provides scalability advantages over relational databases, but limits object retrieval, and does not allow data retrieval on secondary attributes [S27]. Quality-based query processing contributes to the performance of storage systems.

Table 5. Distribution of primary studies over classes.

Research Classes and Relevant Research Papers	
Latency and Throughput	[S06][S11][16][S22][S31][S32][S39]
Consistency	[S01] [S02][S05][S14][S18][S21][S26] [S33][S34]
Security	[S03][S08][S17][S20][S25]
Performance	[S07][S09][S10][S12][S15][S19][S23] [S29][S36][S38][S41][S42][S44][S45]
Query Processing	[S27][S30][S35][S37][S40][S43]
Load Balancing	[S04][S13] [S24][S28]

4.1.2. Solutions

This section provides a detailed analysis of the solutions. The studies that provided solutions for more than one challenge are classified into one class to avoid repetition. Figure 6 shows the percentage of solutions in each class.

Latency and Throughput: Cho et al. [S16] achieve high throughput using a bloom filter with FPGA (Field-Programmable Gate Array) implementation. The bloom filter enhanced the performance of hash tables by controlling the unnecessary access to it. The modified cuckoo hashing algorithm is implemented to utilize the hash table properly. Yue et al. [S22] designed and implemented a KVS that controls the write amplification and achieves high throughput. The experiment showed that the proposed system is highly suitable for put-intensive and scan-intensive workloads. Debnath et al. [S11] designed a persistent KVS that used flash memory and provided high throughput. It used a log structure to organize a key-value pair on flash to achieve high write performance. An in-memory KVS deals with data-intensive applications but does not provide massive data parallelism. Raju et al. presented a high-performance KVS design that modified HyperlevelDB [S06]. Zhang et al. [S32] designed and implemented a system Mega-KV, in-memory KVS that provided high throughput. Li et al. proposed architecture to achieve maximum throughput on single KVS platform [S31]. It is a best choice for massive data applications. This study also identified the software/hardware efforts to achieve high throughput KVS. Li et al. [S39] presented a system ZHT/Q (a flexible QoS (Quality of Service) fortified distributed key-value storage system), which is a KVS that provides high throughput and satisfies a wide range of latency requirements for the cloud environment.

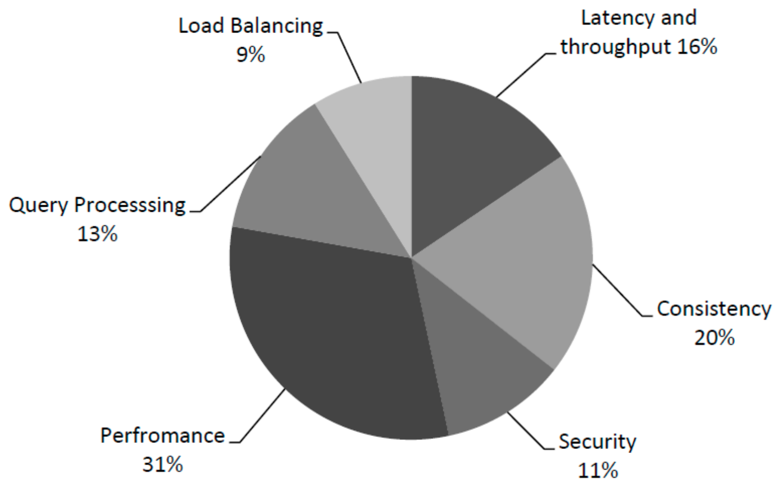


Figure 6. Solutions provided in each class.

Consistency: Bailey et al. [S01] presented Echo, which deployed two types of memory, DRAM (Dynamic Random Access Memory) and SCM (storage-class memory), and also used snapshot isolation to provide consistency, versioning, and concurrency in a KVS-based system. Garefalakis et al. [S02] designed and implemented a system named ACaZoo, which is a KVS that provides high availability and strong consistency. Its compaction operation resolved the issue of file overlaps, which occurs under the high-intensive workloads. Rahman et al. [S05] designed and implemented a system that provided consistency and optimizing operation latency. Most of the Web-based applications use KVS for data storage; they achieved high performance and scalability, but compromised consistency. Xu et al. [S14] proposed a system that improves the consistency of KVS and ensures a balance between consistency and latency. Glendenning et al. [S34] provided the design of a system that offered a high level of consistency in KVSs. It was based on the modified hashing mechanism of Cassandra and had three primary goals: scalability, consistency, and adaptability. Garefalakis et al. [S33] used different components to redesign the popular KVS Cassandra, which provided strong consistency, scalability, and adaptability. To cope with the goals, a group of independent nodes was made that used a replicated state machine. Zhou et al. [S18] proposed a library over KVS for non-volatile memory that provided high performance, consistency (using a Log-based mechanism), and durability. Tan et al. [S21] presented a design of consistent KVS that is highly available, consistent, and scalable for distributed systems. It provided strong consistency, but with a little increase in overhead. Tarnavo et al. proposed the design of an in-memory KVS that provides strong consistency [S26] and allows the user to set the level of resilience on a key-value pair.

Security: Pattuk et al. [S17] presented a system that is highly secure and efficient. This system enabled outsourced encrypted data over public KVS. Key-value pairs can be easily outsourced, and for this purpose, two access control modes were used i.e., an attribute-based mode and a multi-level mode. Yuan et al. [S03] presented the encrypted KVS that allows data retrieval with the help of multiple secondary attributes. Searchable order-revealing encryption and symmetric encryption were used to tackle the security issues. Tang et al. [S08] proposed a system that ensured the data authentication and freshness. It is an efficient system that has low computing power but achieves high throughput. Waye et al. presented the design of a secure KVS [S20] that enforced confidentiality. Most of the web-based applications rely on KVSs due to their simplicity and popularity, which leads to deploying unprotected KVS. While enabling the internet on KVS-based applications, it suffers a range of security attacks, as shown in Figure 7.

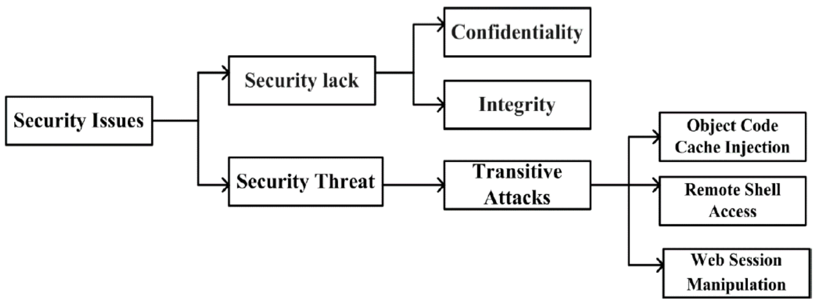


Figure 7. Security issues in KVSs.

Asadulla et al. [S25] presented the design and implementation of a secure Redis (an open source (BSD licensed), in-memory data structure store). This system provided encryption, authentication, and security for BLOB and persistent data. It provided strong security for user-specific sessions and data.

Performance: Son et al. [S23] proposed a system that introduced a new scheme to improve the performance of KVS by combining the old flushing scheme with consolidated array and group commit. This improved system executed the requests with high efficiency. Shim et al. [S19] presented PHash, which is a KVS that provided high performance at minimum memory cost. The system is able to handle on-demand workloads in data-intensive applications. Lim et al. presented SILT (Small Index Large Table), which is a key-value store that used flash storage to improve the performance [S09]. This system combines entropy-coded and cuckoo hashing, which reduced the requirement of memory as compared to existing systems. Lemire et al. [S07] proposed a compression technique for B-trees that made queries operation fast and reduced the storage requirement. This approach employed single instruction multiple data (SIMD) architecture. Lu et al. [S45] proposed a persistent KVS that is faster than RocksDB and LevelDB. The LSM tree indexing technique is used, which improves the performance of solid state disk (SSD) optimized KVS. Thongprasit et al. [S10] proposed a KVS that employed two mechanisms to enhance the performance of a system. This design is highly scalable and faster. Zhang et al. proposed a new compaction method for LSMT (log structure merge tree)-based KVS to improve the write performance [S12]. Wang et al. proposed an approach that provided fast access on massive data [S15]. However, access to the main database was reduced, because the data is stored in the cache, and consequently, the system performance is improved. Li et al. presented a design of KVS that used multiple techniques to develop a mechanism to achieve high performance [S29]. Hong et al. proposed a system that partitioned the data of KVS in such a way that each node had multiple groups of memory [S36]. This system allowed non-uniform memory access (NUMA) aware allocation and provided high throughput. Chang et al. presented an SSD-based system to achieve high performance [S38]. Wu et al. proposed a design of high-performance KVS for distributed systems [S41] that was implemented and evaluated. The result showed that the system performance is retained, even if the number of servers increased. Zhao et al. presented a design that enhanced the Input-Output (I/O) performance of KVS in a distributed environment [S42]. Ahn et al. presented a high-performance KVS system [S44] whose performance was outstanding, even if the keys were very long and randomly distributed in key space.

Query Processing: Escriva et al. [S27] proposed HyperDex, which enabled searches by using the primary key and secondary attributes on distributed key-value store. This system provided strong consistency and availability without compromising on performance. It achieved high throughput for read and write operations. Termehchy et al. [S35] presented a large-scale key-value store that allowed data retrieval by range query instead of a single key. It is capable of handling a large number of nodes and data. This KVS was implemented through a range-key skip graph, which is scalable. Hu et al. [S37] proposed a solution to resolve the issue of privacy in query processing; it maintains the privacy of a query at the client end and the privacy of data at the server end. Ge at al. [S40] implemented a system, CinHBa, that enabled the HBase to operate queries on non-rowkey columns in a very efficient manner. Sun et al. [S43] presented a system named SPIKE that supported complex queries on multi-dimensional datasets. SPIKE implemented and evaluated on Cassandra KVS. Zhou et al. [S30] proposed an approach for similarity searches in KVS that reduced the search time. Cloud applications do a similarity search in high-dimensional data space, but it took a long time, and this approach allows high-dimensional data space similarity searches in a very short time.

Load Balancing: Qin et al. [S04] proposed a system, ElastiCat, for load balancing in cloud-based KVS that ensured the availability and auto balancing of the storage system at minimum cost. Trajano et al. [S13] proposed a system for load balancing; fast response is the main requirement of cloud-based applications. This requirement is achieved via in-memory KVS, but created a problem regarding load balancing. This paper proposed an architecture for in-memory KVS to provide the services for load balancing. Gavrielatos et al. presented a system that controlled the skew problem and provided a load-balanced system [S24]. The results showed that the proposed KVS system achieves high throughput and also guaranteed strong consistency. Jin et al. presented a system for balancing KVS [S28] that cached the network data through new programmable switches.

4.2. RQ2: Which Models and Techniques are Deployed for the Execution and Assessment of the Presented Solutions?

The objective of this question is to investigate the models and techniques applied by researchers for the execution and testing of their proposed solutions. The researchers have used different techniques to achieve the solutions against the challenges, and evaluated these solutions using different techniques.

4.2.1. Models and Techniques for Execution

This research question (RQ) focused on the techniques/strategies that were used to achieve the quality attributes of KVS. To answer this question, we collected data from the primary studies through SLR protocol, as shown in Figure 3. Figure 8 shows the percentage of reported techniques in primary studies.

The results show that the most utilized technique was indexing, which was reported in 20% of the primary studies [S02, S06, S09, S22, S36, S40, S43, S44, S45]. It provides the fast retrieval of data and shows efficient performance when used in queries. This technique also allows data sorting. LSM trees, T-trees, and B$^+$-trees are popular indexing techniques in NoSQL databases. LSM tree indexing is an attractive technique that is used in various NoSQL systems [S02]. LSM-based KVS uses batch writing to store data in backing storage, but this batch writing does not overwrite data. Therefore, data is accumulated in many places. It improves write performance but decreases read performance. The T-structure provides high read performance. The B$^+$-tree is the most popular index structure in modern KVSs [S44].

The second-largest technique from 2010 to 2018 was the hashing scheme. This technique was used in six studies of 45 (13%). The hashing scheme is used in the studies [S11, S16, S19, S27, S32, S41] for different problems. The hashing scheme used a hash table in KVS for the key to value mapping. Its constant time performance for size, add, and remove operations makes it attractive for researchers. However, there is a chance of collision in a hash table, so it needs some mechanism or technique to handle it.

Security techniques were used in 9% [S03, S17, S20, S25] of the primary studies. Security is major concern in KVS because of its widespread use in web-based and cloud applications [36]. Cryptography (encryption) is the most popular technique that is used for the security of publicly accessible KVS. It provides a stronger protection of data from intruders to control data breaches.

The caching technique was used in 9% [S15, S24, S28, S38] of the primary studies. It is the popular technique in KVS to improve the data lookup performance. The most frequently accessed data is placed in a cache to reduce the data processing time.

Paxos is a replication technique (protocol and algorithm) that was used in 4% [S21, S34] of the selected studies. This is a technique that provides strong consistency in KVS, and its protocol [37] is used to provide replica consistency in distributed KVSs with low protocol overhead.

The batching strategy is a technique that was used in 4% [S31, S39] of the primary studies. This strategy is used to reduce the number of messages [S39]. Batching for the client side is a common technique in KVSs that improves the throughput of a system.

The encoding technique was used in 4% [S26, S42] of the primary studies. Locality-aware encoding is used to directly assign the job to the node where the corresponding data reside [S42], and as a result increases the speed and accuracy of the data entry. It takes up less storage space and allows fast data searching. Figure 8 shows the percentage of execution techniques in the primary studies.

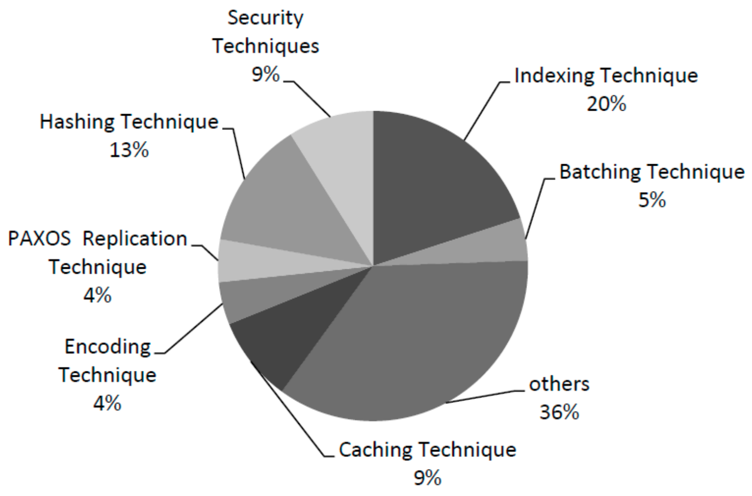


Figure 8. Techniques used in primary studies.

Other individual techniques made up 37% of the primary studies, including soft data partitioning probabilistic models, MAPE (Monitor-Analyze-Plan-Execute) loop model, consistency model, search scheme, optimizing flushing scheme, log-based mechanism, compression technique, programmable Network Interface Cards (NICs), Intel DPDK, and freshness/tardiness mechanisms. The list of techniques and their description are given in Table 6.

Table 6. List and description of execution techniques.

Technique/Strategy	Description	Study ID
Batching technique	Batching strategies are used to combine similar tasks together to make a group.	[S31][S39]
Hashing technique	"A randomized algorithm that selects a hashing function from a list of hash functions, so that the collision probability between two distinct keys is 1/n, where n is the number of distinct hash values desired independently of the two keys."	[S11][S16][S19][S27] [S32][S41]
Indexing technique	Indexing scheme is a technique of data structure to improve the data retrieval speed.	[S02][S06][S09][S22] [S36][S40][S43][S44]
Security techniques	These techniques are used to secure data of KVSs that are publicly accessible.	[S03][S17][S20][S25]
Caching technique	Caching is a technique that helps achieve performance, affordability and scalability against data accuracy and consistency.	[S15][S24][S28][S38]
Paxos technique	Paxos protocol and Paxos algorithm. They are replication techniques. They based on consensus; they elect one node to select a value, send it to all the nodes, and a decision is made by consensus.	[S21][S34]
Encoding technique	Encoding techniques are used to encode information into signals.	[S26][S42]

4.2.2. Algorithm-Based Solutions

The selected studies proposed different solutions to achieve quality attributes. We listed the studies in Table 7 that provide algorithm-based solutions.

The secure data partition algorithm [S03] is used to evenly distribute data across the nodes. The data records are encrypted, so it is called a secure data partition algorithm. The cost-aware rebalancing algorithm [S04] is used to select the rebalancing actions to make a rebalancing plan by utilizing the balance rate and cost model. A control loop algorithm is used for the control loop process [S05], which is a process management system that is used to maintain the variables of the process at a desired set point. The pointer-free buddy allocator (PFBA) algorithm has the characteristics of friendly recovery and also minimizes internal fragmentation [S18]. It manages the large memory region that is divided into small chunks.

Table 7. Algorithms used in selected studies.

Algorithm	Study ID
Secure Data Partition Algorithm	[S03]
Cost-Aware Rebalancing Algorithm	[S04]
Control Loop Algorithm	[S05]
Pointer-Free Buddy Allocator (PFBA).	[S18]
Modified Earliest-Deadline First (EDF) Algorithm.	[S39]
Hotscore Algorithm	[S40]

For dynamic batching mechanisms, the modified earliest deadline first (EDF) algorithm is used [S39]. If the user does not specify the batch size, then the system automatically handled it with the EDF algorithm by considering the following parameters: physical batch size in bytes, logical batch size (number of requests in a batch), and batch latency limit (deadline). The hotscore algorithm [S40] is used for caching policy to insert data into a cache space. It follows the principle of "if-visit-then-insert" to insert data. When a data item is visited, it should be inserted at the head of the cache queue, and the least visited item should be inserted at the tail of cache queue. Hotscore evaluation selects the victim and then kicks out from caching when the cache space is full. In order to balance the load, the key constraint is the cost of balancing actions.

4.2.3. Techniques for Assessment

The solutions proposed by selected studies to address the above-mentioned challenges are evaluated through different techniques. Figure 9 shows the evaluation techniques map with respect to the challenges that have been collected from this systematic literature review (SLR). In the figure, evaluation techniques are represented on the X-axis, and challenges are represented on the Y-axis. The nine different evaluation techniques were used in primary studies.

The most utilized evaluation technique is the microbenchmark, which was used in the 20% of the primary studies. It is designed to test a small unit of a larger system and is often used to evaluate the performance of a critical code. The major advantage of this benchmark is that its runtime is usually small; as a result, it answers the questions quickly.

The second-largest evaluation technique is the Yahoo Cloud Serving Benchmark (YCSB), which was used in 18% of the studies. YCSB is an open-source framework that is used as the de facto standard to evaluate the performance of NoSQL databases. Its main advantage is its capability to evaluate the retrieval performance of multiple types of data-serving systems for a given workload. YCSB has two components: a workload generator as a client, and a desired set of workloads that will be executed by the generator.

The extensive experiment evaluation is performed in 23% of the primary studies. The performance of 21% of the primary studies is compared with the performance of existing system and 14% other evaluation techniques (statistical model, measuring the miss rate, benchmarking software, RDMA (Remote direct memory access)-based evaluation, test-based benchmark) were used in the primary studies.

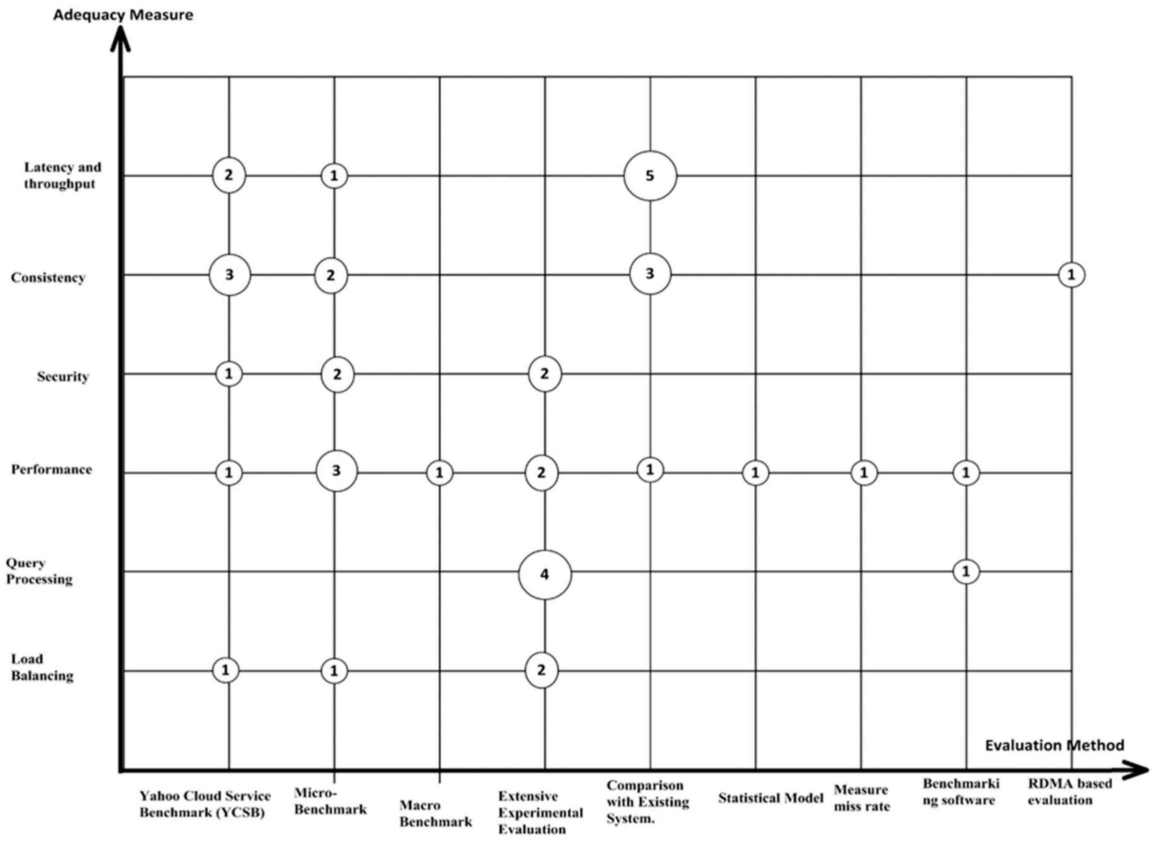


Figure 9. Evaluation techniques perception map with respect to challenges.

One study that is 4% of the primary studies used two of the evaluation techniques i.e., microbenchmark and macro benchmark. The evaluation techniques that were used in the primary studies are listed in Table 8.

Table 8. List and description of evaluation techniques.

Evaluation Technique	Description	Study ID
Yahoo Cloud Serving Benchmark (YCSB)	YCSB is the standard used to evaluate the performance of NoSQL databases. It is a suite that evaluates the retrieval as well as maintenance performance of NoSQL databases for a set of desired workloads	[S02][S04][S06] [S08][S14][S19] [S27][S33]
Microbenchmark	It is a tiny routine to evaluate the performance of language feature or specific hardware.	[S05][S10][S20] [S23][S25][S28] [S29][S34][S39] [S45]
Extensive experimental evaluation	The extensive experiments performed to evaluate the proposed system.	[S03][S13][S17] [S24][S30][S37] [S38][S40][S42] [S43]

Table 8. *Cont.*

Evaluation Technique	Description	Study ID
Comparison with existing system.	Performance of the proposed system is compared with the performance of an already existing system.	[S01][S11][S12] [S16][S18][S21] [S22][S31][S32]
Statistical model.	The system is evaluated by a statistical model of user behavior. It is concluded by Facebook for key-value storage.	[S41]
Measure miss rate	The system performance is evaluated in terms of the miss rate over six different reuse patterns.	[S36]
Benchmarking software	They compile benchmarking software for system evaluation.	[S07][S35]
RDMA-based evaluation	The proposed approach is evaluated on an RDMA network.	[S26]
Test-based benchmark	Benchmark for testing is the access time in milliseconds.	[S15]

4.3. *RQ3: What Is the Reliability Status of the Presented Approaches, And What Is the Quality Status of the Included Publications?*

The reliability status of presented approaches is ensured by the selection of data sources. We selected predominantly related studies from peer-reviewed forums, including IEEE, ACM, Springer, and Elsevier. The reviewed studies are selected from high-impact conferences and journals, symposiums, and workshops.

The most important and critical step of this literature review is to assess the studies that we have studied. We assessed the primary studies on the basis of assessment criteria [11,12]. For this purpose, we have formulated a certain criteria as shown in Table 9, and checked all the studies against these criteria to ensure that they are highly pertinent to our research questions. Our research criteria assessed the studies in the following way: whether the studies displayed formal research methods or were only based upon the judgment of experts, and whether the studies clearly depicted the research objectives or not. It also analyzed whether the study proved their findings by experiments, and whether the studies clearly mention directions for future research?

Table 9. Assessment criteria for primary studies.

	Assessment Criteria
C1	Has the study included formal research, or was it just based upon the judgment of expert?
C2	Are the research aims clearly depicted and addressed in the studies?
C3	Does the paper mention the limitations for future work?
C4	Is there any description about the evaluation of the proposed solution?
C5	Were the experiments performed on adequate datasets?

To assess the studies, we chose a scale for scoring: 1 for yes, 0 for no, and 0.5 for partly. We calculated the score of each study according to the designed criteria and the used scale is shown in Table 10.

Table 11 shows the average score of C1, C2, C3, C4, and C5, the average score of C1 and C2 shows whether the research paper has done a formal search, and clearly expressed the research objectives. The average scores for C3, C4, and C5 were 0.33, 0.94, and 0.86.

Table 10. Assessment scores of reviewed literature.

Assessment Scores of Reviewed Literature													
Paper ID	**Formal Research C1**	**Objectives C2**	**Limitations C3**	**Findings C4**	**Experiment C5**	**Total Score**	**Paper ID**	**Formal Research C1**	**Objectives C2**	**Limitations C3**	**Findings C4**	**Experiment C5**	**Total Score**
[S02]	1	1	1	1	1	5	[S23]	1	1	1	1	1	5
[S07]	1	1	1	1	1	5	[S33]	1	1	1	1	1	5
[S13]	1	1	1	1	1	5	[S39]	1	1	1	1	1	5
[S16]	1	1	1	1	1	5	[S40]	1	1	1	1	1	5
[S17]	1	1	1	1	1	5	[S43]	1	1	1	1	1	5
[S10]	1	1	0.5	1	1	4.5	[S42]	1	1	0.5	1	1	4.5
[S04]	1	1	0.5	1	1	4.5	[S24]	1	1	0.5	1	1	4.5
[S31]	1	1	0.5	1	1	4.5	[S36]	1	1	0.5	1	1	4.5
[S37]	1	1	0.5	1	1	4.5	[S44]	1	1	1	1	0.5	4.5
[S03]	1	1	0	1	1	4	[S41]	1	1	0	1	1	4
[S01]	1	1	0	1	1	4	[S38]	1	1	0	1	1	4
[S05]	1	1	0	1	1	4	[S32]	1	1	0	1	1	4
[S08]	1	1	0	1	1	4	[S06]	1	1	0	1	1	4
[S09]	1	1	0	1	1	4	[S26]	1	1	0	1	1	4
[S11]	1	1	0	1	1	4	[S28]	1	1	0	1	1	4
[S14]	1	1	0	1	1	4	[S34]	1	1	0	1	1	4
[S15]	1	1	0	1	1	4	[S21]	1	1	0	1	1	4
[S18]	1	1	0	1	1	4	[S22]	1	1	0	1	1	4
[S19]	1	1	0	1	1	4	[S27]	1	1	0	1	1	4
[S30]	1	1	0	1	1	4	[S45]	1	1	0	1	1	4
[S12]	1	1	0	1	0.5	3.5	[S29]	1	1	0	1	0.5	3.5
[S25]	1	1	0.5	0	1	3.5	[S20]	1	1	0	0	1	3
[S35]	1	1	0	0	0.5	2.5							

Table 11. Average of assessment scores.

	Formal Research C1	**Objectives C2**	**Limitations C3**	**Findings C4**	**Experiment C5**	**Total Score**
Average	1	1	0.3	0.95	0.95	4.2

4.4. RQ4: What Are the Forums of Publication and Line of Development in Studies on the Key-Value Store?

Table 12 shows the overview of selected studies with the respective primary study and its publication venue of study.

Table 12. Selected studies with publication venue.

Study ID	Publication Year	Publication Type	Publication Venue
[05]	2017	Journal	ACM Transactions on Autonomous and Adaptive Systems
[07]	2017	Journal	Information Systems
[13]	2016	Journal	ACM
[14]	2014	Journal	Distributed and Parallel Databases
[19]	2017	Journal	Journal of Systems and Software
[22]	2016	Journal	IEEE on Parallel and Distributed Systems
[31]	2016	Journal	ACM Transactions on Computer Systems (TOCS), 34(2), 5.
[42]	2016	Journal	Journal of Parallel and Distributed Computing,
[43]	2017	Journal	Information Systems, Elsevier
[44]	2016	Journal	IEEE transaction
[45]	2017	Journal	ACM transaction on storage.
[03]	2017	Conference	ACM on Asia Conference on Computer and Communications Security
[04]	2012	Conference	Cluster Computing Workshops (CLUSTER WORKSHOPS), EEE International Conference
[06]	2012	Conference	Mass Storage Systems and Technologies (MSST), IEEE Symposium
[08]	2014	Conference	Concepts, Theory and Applications (ICAICTA), International Conference
[10]	2015	Conference	In Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium IEEE.
[12]	2016	Conference	Parallel and Distributed Systems (ICPADS), EEE International Conference
[15]	2015	Conference	International Conference on Intelligent Science and Big Data Engineering, Springer
[17]	2013	Conference	Cloud computing (CLOUD), IEEE international Conference.
[18]	2016	Conference	International Conference on Big Data Computing, Applications and Technologies ACM
[20]	2017	Conference	Workshop on Automated Decision Making for Active Cyber Defense.
[21]	2014	Conference	IFIP International Conference on Network and Parallel Computing, Springer Berlin Heidelberg
[24]	2015	Conference	Trustcom/BigDataSE/I SPA, 2016 IEEE (pp. 1660-1667). IEEE
[25]	2015	Conference	Electrical, Computer and Communication Technologies (ICECCT), 2015 IEEE International Conference
[26]	2018	Conference	ACM SIGPLAN workshop on ERLANG.
[27]	2012	Conference	ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication Computer Communication Review
[30]	2013	Conference	Concurrency and Computation: Practice and Experience,
[33]	2013	Conference	International Conference on Distributed Applications and Interoperable Systems Springer.
[35]	2010	Conference	In Proceedings of the 19th international conference on World wide web (pp. 1193-1194). ACM.
[36]	2016	Conference	Computer Design (ICCD), International Conference, IEEE.
[37]	2014	Conference	Data Engineering (ICDE), International Conference on (pp. 628-639). IEEE.
[38]	2014	Conference	International Conference on Database Systems for Advanced Applications, Springer.

Table 12. *Cont.*

Study ID	Publication Year	Publication Type	Publication Venue
[39]	2015	Conference	In Big Data (Big Data), IEEE International Conference. IEEE.
[40]	2014	Conference	International Conference on Advanced Data Mining and Applications, Springer.
[41]	2016	Conference	International Asia Conference on Industrial Engineering and Management Innovation, Atlantis Press.
[23]	2016	Workshop	Foundations and Applications of Self* Systems, IEEE International Workshops.
[28]	2017	Workshop	Symposium on Operating Systems Principles, ACM.
[01]	2013	Workshop	Workshop on Interactions of NVM/FLASH with Operating Systems and Workloads
[29]	2017	Symposium	Annual ACM Symposium on Applied Computing (pp. 2072-2074). ACM.
[34]	2011	Symposium	Symposium on Operating Systems Principles ACM.
[02]	2014	Symposium	Reliable Distributed Systems (SRDS), IEEE International Symposium.
[09]	2011	Symposium	ACM Symposium on Operating Systems Principles
[16]	2014	Symposium	VLSI Design, Automation and Test (VLSI-DAT), International Symposium, IEEE.
[11]	2010	VLDB Endowment	ACM
[32]	2015	VLDB Endowment	ACM

4.4.1. Overview of Studies

We have selected and reviewed literature from different publication forums. The most leading publication forum is ACM, which had 40% (18) of the primary studies; it is followed by IEEE, which contained 31% (14) of the primary studies. There were 18% (eight) primary studies from Springer studies and 11% (five) of primary studies from Elsevier. Figure 10 shows the publications distribution over data sources.

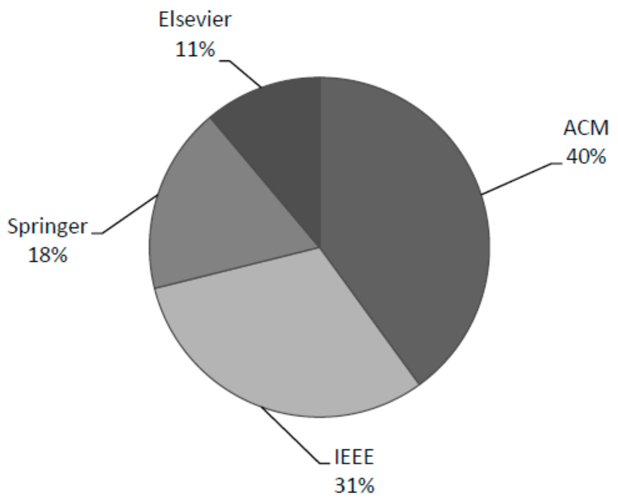


Figure 10. Publications distribution over publication forums.

4.4.2. Publication Over Years

We have studied the literature from 2010 to 2018, and then distributed the literature on the basis of the year of publication, as shown in Figure 11. After applying the screening process, there were only a few papers in 2010 and 2011 and then a gradual increase in subsequent years: three papers in 2012, four papers in 2013, eight papers in 2014, six papers in 2015, 10 papers in 2016, nine papers in 2017, and one by April 2018.

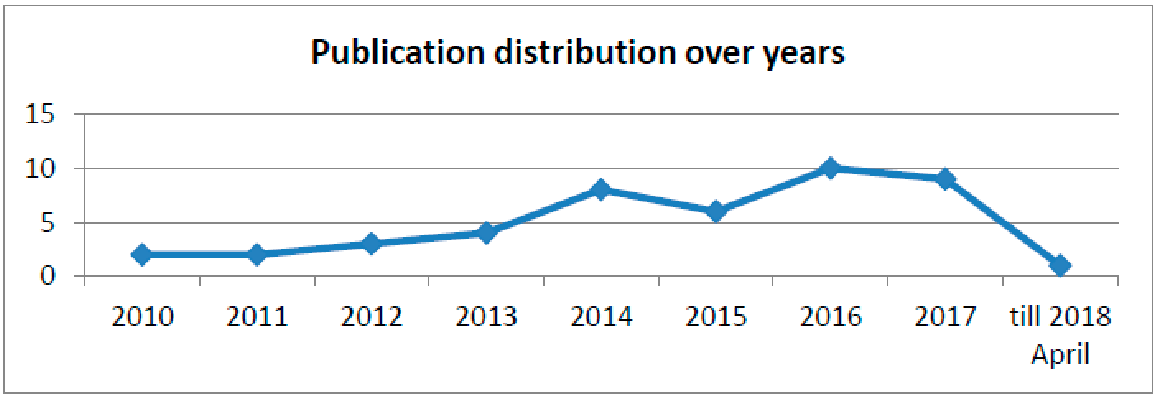


Figure 11. Publications distribution over years.

5. Discussion

We have been attempting to provide a SLR from the software engineering perspective, and looked at software engineering quality attributes for KVSs. The overall goal of this study was to identify the challenges for KVSs to achieve quality attributes, and their corresponding solutions. In particular, we look at consistency, throughput and latency, security, performance, load balancing, and query processing. A broad set of distinct software engineering scenarios is covered by these attributes, and these attributes have been used as the target to pick primary studies.

NoSQL databases replace relational databases by providing availability at the cost of consistency. In this SLR, we discussed different solutions that provide strong consistency in NoSQL databases. The consistent hashing on replica groups is an effective technique to ensure strong consistency. Consistent hashing provides elastic scaling through dynamically adding or removing a database server. When a node is added or removed, it requires the minimum amount of data movement between servers. It also facilitates the data partitioning and replication across the servers.

To improve the throughput of KVS systems, different tree data structures were proposed by the researchers. The balance between throughput and latency is very important, because the KVS is the backbone for distributed systems and data-intensive systems; the performance of KVS is also an important concern. There is a need to design KVS that can achieve high write throughput with a low latency of searches. LSMT-based KVS provide good read performance, but poor write performance at the same time. The use of bloom filters with LSM trees improves the write performance. For range queries, bloom filters do not provide the required latency and throughput. Data structures and an advanced index design improve the performance of KVSs with minimum memory usage. A novel tree data structure FLSM (fragmented log-structured merge trees) is the most efficient data structure that provides a considerable increase in write throughput and low latency for range queries. FLSM exploit the non-volatile memory storage to achieve high parallelism.

Load balancing is required for cloud-based KVSs. These stores use consistent hashing, which results in a network load imbalance. The long response time problem is due to consistent hashing techniques. The key length and key distribution are the main factors that cause the load-balancing problem. These factors (key length and key distribution) and query volume can also impact the cluster response times. ElastiCat and network function virtualization architecture are the efficient solutions that control these load imbalance factors and balance the load for KVSs.

KVSs suffered security issues because they are usually deployed without proper protection mechanisms. The increasing popularity of KVS raises the user concern for the security of data, such as the confidentiality and privacy of the data. KVSs have no security feature that can embed security in the database. Security is difficult in these stores due to unstructured data. Since KVS uses a distributed database, there is a great risk of security attacks across the distributed nodes. The main security challenges are authorization and access control, data integrity, and data protection. Data at rest can be protected with OS-level security. An encrypted key-value pair with secure query support is an effective security solution.

For efficient query, KVSs must use search primitives that allow queries on secondary attributes to improve the throughput on get/put operations. Cloud computing and big data usually rely on KVSs, but these stores do not efficiently support the range queries on multi-dimensional datasets. The kernel indexing scheme is required to provide efficient multi-dimensional complex query processing. An SP index-based solution called SPIKE is providing efficient support to range queries.

6. Threats to Validity

We have selected studies for this SLR according to the research protocol presented in Section 3. The research protocol has different stages to select relevant studies. Firstly, we formulated research questions; then, we have done a search on the basis of title and string according to the defined strategy. In the end, we have applied screening criteria to select the most relevant studies. Research bias arises when there are only positive search results; our research protocol is very helpful for us to reduce research biasness in this SLR.

Although primary studies are selected according to the guidelines of systematic strategy, there is still a possibility of missing some relevant studies. The reason is the exclusion of the PhD theses and technical reports that may be relevant. In this SLR, we excluded PhD theses and technical reports because most of the time, the authors only briefly reported on these studies. Another factor that can lead to overlooking some relevant publications is to find the appropriate search string. The complete justification is given in Section 3.5 regarding the selection of repositories and the use of search strings. We refine our search strings until all the relevant studies are included that were initially missed, but are available in the reference of some studies. Data is extracted from selected databases that have maximum probability to return the most relevant studies.

In this SLR, incomplete data extraction can be the reason for inaccurate results. We try to handle this problem in two ways. First, we identified the data items that should be extracted, and afterwards, all the extracted data was reviewed several times.

We have not taken studies from sources other than the selected databases, which may have answered our research questions in a better way. Another important issue raised is that we only included research paper from the English published material; as a result, we may have missed the relevant studies that were published in any other language. However, we selected studies published from 2010 to 2018, so works in process or unpublished papers were not included, which may have answered our research questions. Consequently, we may have missed some existing techniques and processes in this review.

7. Conclusions

The aim of this review is to organize the available literature on quality attribute challenges and their corresponding KVS solutions. According to my knowledge, there are no reviews on key-value store that thoroughly analyze and identify the challenges and solutions. This research identifies the challenges that can be addressed in a better way with the help of current studies that were reported in this review. This research provides a protocol and criteria to evaluate the selected studies. The protocol defined steps to scrutinize the studies, data collection and synchronization, and quality assessment to achieve research goals. The most relevant studies were selected; then, we evaluated these studies against quality criteria. Data was collected from these research studies and then synchronized. The essence of this research is that challenges can be solved in an effective manner to enhance the quality attributes in KVS. The SLR tries to identify the key-value store challenges in NoSQL databases such as consistency, latency, throughput, performance, load balancing, security, and query processing. The SLR also identifies the trending research; for example, performance is the most researched issue in the domain right now, while the indexing technique has obtained more focus as a solution in the NoSQL databases, while in the evaluation of current solutions, comparisons with the existing systems are mostly used by the researcher to justify their results.

(RQ1) The 45 most relevant studies were chosen for this SLR, which are addressing quality challenges for KVS-based systems. There are various solutions for each challenge (consistency, throughput and latency, security, performance, load balancing, and query processing).

(RQ2) Various techniques were used to achieve the quality attributes; namely, these included the indexing technique, hashing scheme, security techniques, caching technique, batching strategy, and encoding techniques. These techniques differ from each other on the basis of their input and output methods, but are used to achieve quality attributes. The indexing scheme was used in 21% of the studies, hashing schemes were used in 15% of the studies, the caching technique was used in 13% of the studies, security techniques were used in 10% of the studies, and the batching strategy, encoding technique, and Paxos family were each used in 5% of the studies. Algorithm-based solutions were also proposed to address these quality attributes challenges such as the secure data partition algorithm, control loop algorithm, pointer-free buddy allocator (PFBA), modified earliest-deadline first (EDF) algorithm, and the hotscore algorithm.

These proposed solutions were evaluated through different techniques, which included the Yahoo Cloud Serving Benchmark (YCSB), microbenchmark, macrobenchmark, extensive experimental evaluation, comparison with existing systems, statistical model, the measure miss rate, benchmarking software, RDMA-based evaluation, and test-based benchmark.

(RQ3) The quality assessment criterion is defined to ensure the reliability status of the presented approaches and quality status of included publications. To evaluate the studies, we choose a scale for scoring: 1 for yes, 0 for no, and 0.5 for partly. We have calculated the score of each study according to the designed criteria and scale. The average score of C1 and C2 shows that all of the research paper included in the study are selected after formal searches and had clearly expressed their research objectives. The average scores for C3, C4, and C5 are 0.33, 0.94, and 0.86. We selected predominantly related studies from peer-reviewed forums such as IEEE, ACM, Springer, and Elsevier. The reviewed studies are selected from high-impact conference forums and journals, symposium, and workshops.

(RQ4) The forum of publication of each study is clearly mentioned, and the line of development in studies on key-value store is depicted by showing the detail of the primary studies' publication venue and primary studies distribution over publication forums and each year.

In future studies, we focus on the challenges that have a strong impact on the user's confidence to deploy the KVS in their systems, such as privacy and trust. Although these are very important questions, they have not yet been addressed in any review. Another important aspect is to raise the questions that can improve user experience regarding simplicity, comfort, and ease of NoSQL key-value stores.

Author Contributions: S.R. contributed in design, implementation, and experimentation of this research and writing this manuscript. I.S.B. supervised this work and also edited this manuscript. R.K. contributed in visualization and evaluation of this research. A. contributed in devising research questions and proof-reading the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

List of the studies that have been reviewed in this SLR.

[S01]. Bailey, K.A.; Hornyack, P.; Ceze, L.; Gribble, S.D.; Levy, H.M. Exploring storage class memory with key value stores. In Proceedings of the 1st Workshop on Interactions of NVM/FLASH with Operating Systems and Workloads, Farmington, PA, USA, 3 November 2013; p. 4.

[S02]. Garefalakis, P.; Papadopoulos, P.; Magoutis, K. Acazoo: A Distributed key-value store based on replicated lsm-trees. In Proceedings of the 2014 IEEE 33rd International Symposium on Reliable Distributed Systems (SRDS), Nara, Japan, 6–9 October 2014; pp. 211–220).

[S03]. Yuan, X.; Guo, Y.; Wang, X.; Wang, C.; Li, B.; Jia, X. Enckv: An encrypted KVS with rich queries. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, Abu Dhabi, United Arab Emirates, 2–6 April 2017; pp. 423–435.

[S04]. Qin, X.; Wang, W.; Zhang, W.; Wei, J.; Zhao, X.; Huang, T. Elasticat: A load rebalancing framework for cloud-based KVSs. In Proceedings of the 2012 19th International Conference on High Performance Computing (HiPC), Pune, India, 18–21 December 2012; pp. 1–10.

[S05]. Rahman, M.R.; Tseng, L.; Nguyen, S.; Gupta, I.; Vaidya, N. Characterizingand adapting the consistency-latency tradeoff in distributed KVSs. *ACM Trans. Auton. Adapt. Syst. (TAAS)*, **2017**, *11*, 20.

[S06]. Raju, P.; Kadekodi, R.; Chidambaram, V.; Abraham, I. PebblesDB: Building KVSs using Fragmented Log-Structured Merge Trees. In Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, 28 October 2017; pp. 497–514.

[S07]. Lemire, D.; Rupp, C. Upscaledb: Efficient integer-key compression in a KVS using SIMD instructions. *Inf. Syst.* **2017**, *66*, 13–23.

[S08]. Tang, Y.; Wang, T.; Liu, L.; Hu, X.; Jang, J. Lightweight authentication of freshness in outsourced KVSs. In Proceedings of the 30th Annual Computer Security Applications Conference, Los Angeles, CA, USA, 7–11 December 2014; pp. 176–185.

[S09]. Lim, H.; Fan, B.; Andersen, D.G.; Kaminsky, M. SILT: A memoryefficient, high- performance KVS. In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, Cascais, Portugal, 23–26 October 2011; pp. 1–13).

[S010]. Thongprasit, S.; Visoottiviseth, V.; Takano, R. Toward fast and scalable KVSs based on user space TCP/ip stack. In Proceedings of the Asian Internet Engineering Conference, Bangkok, Thailand, 18–20 November 2015; pp. 40–47.

[S011]. Debnath, B.; Sengupta, S.; Li, J. FlashStore: High throughput persistent KVS. *Proc. VLDB Endow.* **2010**, *3*, 1414–1425.

[S012]. Zhang, W.; Xu, Y.; Li, Y.; Li, D. Improving Write Performance of LSMT-Based KeyValueStore. In Proceedings of the 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), Wuhan, China, 13–16 December 2016; pp. 553–560).

[S013]. Trajano, A.F.; Fernandez, M.P. Two-phase load balancing of In-Memor Key-Value Storages using Network Functions Virtualization (NFV). *J. Netw. Comput. Appl.* **2016**, *69*, 1–13.

[S014]. Xu, C.; Sharaf, M.A.; Zhou, X.; Zhou, A. Quality-aware schedulers for weak consistency key-value data stores. *Distrib. Parallel Databases* **2014**, *32*, 535–581.

[S015]. Wang, L.; Chen, G.; Wang, K. Research of Massive Data Caching Strategy Based on Key-Value Storage Model. In *Proceedings of the International Conference on Intelligent Science and Big Data Engineering, Suzhou, China, 14–16 June 2015*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 443–453.

[S016]. Cho, J.M.; Choi, K. An FPGA implementation of high-throughput KVS using Bloom filter. In Proceedings of the 2014 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), Hsinchu, Taiwan, 25–27 April 2014; pp. 1–4.

[S017]. Pattuk, E.; Kantarcioglu, M.; Khadilkar, V.; Ulusoy, H.; Mehrotra, S. Bigsecret: A secure data management framework for KVSs. In Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing (CLOUD), Santa Clara, CA, USA, 28 June–3 July 2013; pp. 147–154.

[S018]. Zhou, J.; Shen, Y.; Li, S.; Huang, L. NVHT: An efficient key-value storage library for non-volatile memory. In Proceedings of the 3rd IEEE/ACM International Conference on

Big Data Computing, Applications and Technologies, Shanghai, China, 6–9 December 2016; pp. 227–236.

[S019]. Shim, H. PHash: A memory-efficient, high-performance KVS for large-scale data intensive applications. *J. Syst. Softw.* **2017**, *123*, 33–44.

[S020]. Waye, L.; Buiras, P.; Arden, O.; Russo, A.; Chong, S. Cryptographically Secure Information Flow Control on KVSs. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1893–1907.

[S021 .] Tan, Z.; Dang, Y.; Sun, J.; Zhou, W.; Feng, D. PaxStore: A Distributed Key Value Storage System. In Proceedings of the IFIP International Conference on Network and Parallel Computing, Ilan, Taiwan, 18–20 September 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 471–484.

[S022]. Yue, Y.; He, B.; Li, Y.; Wang, W. Building an Efficient Put-Intensive KVS with Skip-tree. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *28*, 961–973.

[S023]. Son, Y.; Kang, H.; Han, H.; Yeom, H.Y. Improving Performance of Cloud Key-Value Storage Using Flushing Optimization. In Proceedings of the IEEE International Workshops on Foundations and Applications of Self* Systems, Augsburg, Germany, 12–16 September 2016; pp. 42–47).

[S024]. Gavrielatos, V.; Katsarakis, A.; Joshi, A.; Oswald, N.; Grot, B.; Nagarajan, V. Scale-out ccNUMA: Exploiting skew with strongly consistent caching. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 21–23 April 2018, p. 21.

[S025]. Zaki, A.K.; Indiramma, M. A novel redis security extension for NoSQL database using authentication and encryption. In Proceedings of the 2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 5–7 March 2015; pp. 1–6.

[S026]. Taranov, K.; Alonso, G.; Hoefler, T. Fast and strongly-consistent per-item resilience in KVSs. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018; p. 39.

[S027]. Escriva, R.; Wong, B.; Sirer, E.G. HyperDex: A distributed, searchable key-value store. In Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Helsinki, Finland, 13–17 August 2012; pp. 25–36.

[S028]. Jin, X.; Li, X.; Zhang, H.; Soulé; R; Lee, J.; Foster, N.; Stoica, I. NetCache: Balancing KVSs with Fast In-Network Caching. In Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, 28 October 2017; pp. 121–136.

[S029]. Li, B.; Ruan, Z.; Xiao, W.; Lu, Y.; Xiong, Y.; Putnam, A.; Zhang, L. KV-Direct: High-Performance In-Memory KVS with Programmable NIC. In In Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, 28 October 2017; pp. 137–152.

[S030]. Najaran, M.T.; Hutchinson, N.C. Innesto: A searchable key/value store for highly dimensional data. In Proceedings of the 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), Bristol, UK, 2–5 December 2013; Volume 1, pp. 411–420.

[S031]. Li, S.; Lim, H.; Lee, V.W.; Ahn, J.H.; Kalia, A.; Kaminsky, M.; Dubey, P. Full-Stack Architecting to Achieve a Billion-Requests-Per-Second Throughput on a Single KVS Server Platform. *ACM Trans. Comput. Syst. (TOCS)* **2016**, *34*, 5.

[S032]. Zhang, K.; Wang, K.; Yuan, Y.; Guo, L.; Lee, R.; Zhang, X. Mega-KV: A case for GPUs to maximize the throughput of in-memory KVSs. *Proc. VLDB Endow.* **2015**, *8*, 1226–1237.

[S033]. Garefalakis, P.; Papadopoulos, P.; Manousakis, I.; Magoutis, K. Strengthening consistency in the cassandra distributed KVS. In *Proceedings of the IFIP International Conference on Distributed*

Applications and Interoperable Systems, June 2013; Springe: Berlin/Heidelberg, Germany, 2013; pp. 193–198.

[S034]. Glendenning, L.; Beschastnikh, I.; Krishnamurthy, A.; Anderson, T. Scalable consistency in Scatter. In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, Cascais, Portugal, 23–26 October 2011; pp. 15–28.

[S035]. Takeuchi, S.; Shinomiya, J.; Shiraki, T.; Ishi, Y.; Teranishi, Y.; Yoshida, M.; Shimojo, S. A large scale KVS based on range-key skip graph and its applications. In Proceedings of the International Conference on Database Systems for Advanced Applications, Tsukuba, Japan, 1–4 April 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 432–435.

[S036]. Hong, B.; Kwon, Y.; Ahn, J.H.; Kim, J. Adaptive and flexible KVSs through soft data partitioning. In Proceedings of the 2016 IEEE 34th International Conference on Computer Design (ICCD), Phoenix, AZ, USA, 3–5 October 2016; pp. 296–303.

[S037]. Hu, H.; Xu, J.; Xu, X.; Pei, K.; Choi, B.; Zhou, S. Private searchon KVSs with hierarchical indexes. In Proceedings of the 2014 IEEE 30th International Conference on Data Engineering (ICDE), Chicago, IL, USA, 31 March–4 April 2014; pp. 628–639.

[S038]. Chang, D.; Zhang, Y.; Yu, G. MaiterStore: A Hot-Aware, High- Performance KVS for Graph Processing. In Proceedings of the International Conference on Database Systems for Advanced Applications, Bali, Indonesia, 21–24 April 2014; Springer: Berlin/Heidelberg, Germany; 2014; pp. 117–131.

[S039]. Li, T.; Wang, K.; Zhao, D.; Qiao, K.; Sadooghi, I.; Zhou, X.; Raicu, I. A flexible qos fortified distributed key-value storage system for the cloud. In Proceedings of the 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 29 October–1 November 2015; pp. 515–522.

[S040]. Ge, W.; Huang, Y.; Zhao, D.; Luo, S.; Yuan, C.; Zhou, W.; Zhou, J. Cinhba: A secondary index with hotscore caching policy on key-value data store. In *Proceedings of the International Conference on Advanced Data Mining and Applications, December 2014*; Springer: Berlin/Heidelberg, Germany, pp. 602–615.

[S041]. Wu, H.J.; Lu, K.; Li, G. Design and Implementation of Distributed Stage DB: A High Performance Distributed Key-Value Database. In *Proceedings of the 6th International Asia Conference on Industrial Engineering and Management Innovation*; Atlantis Press: Paris, France, 2016; pp. 189–198.

[S042]. Zhao, D.; Wang, K.; Qiao, K.; Li, T.; Sadooghi, I.; Raicu, I. Toward high-performance KVSs through GPU encoding and locality-aware encoding. *J. Parallel Distrib. Comput.* **2016**, *96*, 27–37.

[S043]. Sun, H.; Tang, Y.; Wang, Q.; Liu, X. Handling multi-dimensional complex queries in key-value data stores. *Inf. Syst.* **2017**, *66*, 82–96.

[S044]. Ahn, J.S.; Seo, C.; Mayuram, R.; Yaseen, R.; Kim, J.S.; Maeng, S. ForestDB: A fast key-value storage system for variable-length string keys. *IEEE Trans. Comput.* **2016**, *65*, 902–915.

[S045]. Lu, L.; Pillai, T.S.; Gopalakrishnan, H.; Arpaci-Dusseau, A.C.; Arpaci-Dusseau, R.H. WiscKey: Separating keys from values in SSD-conscious storage. *ACM Trans. Storage (TOS)* **2017**, *13*, 5.

References

1. Codd, E.F. Relational database: A practical foundation for productivity. In *Readings in Artificial Intelligence and Databases*; Elsevier: Amsterdam, The Netherlands, 1988; pp. 60–68.
2. Strozzi, C. Nosql-a relational database management system. *Lainattu* **1998**, *5*, 2014.
3. Moniruzzaman, A.; Hossain, S.A. Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. *arXiv* **2013**, arXiv:1307.0191.

4. Lith, A.; Mattsson, J. *Investigating Storage Solutions for Large Data—A Comparison of Well Performing and Scalable Data Storage Solutions for Real Time Extraction and Batch Insertion of Data*; Chalmers University Of Technology: Göteborg, Sweden, 2010.
5. Sharma, V.; Dave, M. Sql and nosql databases. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **2012**, *2*, 20–27.
6. Cattell, R. Scalable sql and nosql data stores. *ACM SIGMOD Rec.* **2011**, *39*, 12–27. [CrossRef]
7. Leavitt, N. Will nosql databases live up to their promise? *Computer* **2010**, *43*, 12–14. [CrossRef]
8. Schram, A.; Anderson, K.M. MySQL to NoSQL: Data modeling challenges in supporting scalability. In Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity, Tucson, AR, USA, 19–26 October 2012.
9. Orend, K. Analysis and classification of nosql databases and evaluation of their ability to replace an object-relational persistence layer. *Architecture* **2010**, *1*. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.184.483&rep=rep1&type=pdf (accessed on 29 April 2019).
10. Kuznetsov, S.D.; Poskonin, A.V. Nosql Data Management Systems. *Program. Comput. Softw.* **2014**, *40*, 323–332. [CrossRef]
11. Bell, G.; Hey, T.; Szalay, A. Beyond the data deluge. *Science* **2009**, *323*, 1297–1298. [CrossRef]
12. Almassabi, A.; Bawazeer, O.; Adam, S. Top NewSQL Databases and Features Classification. *Int. J. Database Manag. Syst.* **2018**, *10*, 11–31. [CrossRef]
13. McAfee, A.; Brynjolfsson, E.; Davenport, T.H.; Patil, D.; Barton, D. Big data: The management revolution. *Harv. Bus. Rev.* **2012**, *90*, 60–68.
14. Gomez, A.; Ouanouki, R.; April, A.; Abran, A. Building an experiment baseline in migration process from sql database to column oriented no-sql databases. *J. Inf. Technol. Softw. Eng.* **2014**, *4*, 137.
15. Hecht, R.; Jablonski, S. Nosql evaluation: A use case oriented survey. In Proceedings of the 2011 International Conference on Cloud and Service Computing (CSC), Hong Kong, China, 12–14 December 2011; pp. 336–341.
16. DeCandia, G.; Hastorun, D.; Jampani, M.; Kakulapati, G.; Lakshman, A.; Pilchin, A.; Sivasubramanian, S.; Vosshall, P.; Vogels, W. Dynamo: Amazon's highly available key-value store. *ACM SIGOPS Oper. Syst. Rev.* **2007**, *41*, 205–220. [CrossRef]
17. Wylie, B.; Dunlavy, D.; Davis, W.; Baumes, J. Using nosql databases for streaming network analysis. In Proceedings of the 2012 IEEE Symposium on Large Data Analysis and Visualization (LDAV), Seattle, WA, USA, 14–15 October 2012; pp. 121–124.
18. Chang, F.; Dean, J.; Ghemawat, S.; Hsieh, W.C.; Wallach, D.A.; Burrows, M.; Chandra, T.; Fikes, A.; Gruber, R.E. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.* **2008**, *26*, 4. [CrossRef]
19. Cooper, B.F.; Ramakrishnan, R.; Srivastava, U.; Silberstein, A.; Bohannon, P.; Jacobsen, H.-A.; Puz, N.; Weaver, D.; Yerneni, R. Pnuts: Yahoo!'S hosted data serving platform. *Proc. VLDB Endow.* **2008**, *1*, 1277–1288. [CrossRef]
20. Marston, S.; Li, Z.; Bandyopadhyay, S.; Zhang, J.; Ghalsasi, A. Cloud computing—The business perspective. *Decis.Supp. Syst.* **2011**, *51*, 176–189. [CrossRef]
21. Lourenço, J.R.; Cabral, B.; Carreiro, P.; Vieira, M.; Bernardino, J. Choosing the right nosql database for the job: A quality attribute evaluation. *J. Big Data* **2015**, *2*, 18. [CrossRef]
22. Horie, H.; Asahara, M.; Yamada, H.; Kono, K. Pangaea: A single key space, inter-datacenter key-value store. In Proceedings of the 2013 International Conference on Parallel and Distributed Systems (ICPADS), Seoul, Korea, 15–18 December 2013; pp. 434–435.
23. Davoudian, A.; Chen, L.; Liu, M. A survey on nosql stores. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 40. [CrossRef]
24. Fitzpatrick, B.; Vorobey, A. Memcached: A Distributed Memory Object Caching System. 2011. Available online: http://memcached.org/ (accessed on 4 January 2019).
25. Jang, J.; Cho, Y.; Jung, J.; Jeon, G. Enhancing lookup performance of key-value stores using cuckoo hashing. In Proceedings of the 2013 Research in Adaptive and Convergent Systems, Montreal, QC, Canada, 1–4 October 2013; pp. 487–489.
26. Iwazume, M.; Iwase, T.; Tanaka, K.; Fujii, H.; Hijiya, M.; Haraguchi, H. Big data in memory: Benchimarking in memory database using the distributed key-value store for machine to machine communication. In Proceedings of the 2014 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Las Vegas, NV, USA, 30 June–2 July 2014; pp. 1–7.

27. Cronin, D. *A Survey of Modern Key-Value Stores*; Cal Poly Computer Science Department Labs: San Luis Obispo, CA, USA, 2012.
28. Grolinger, K.; Higashino, W.A.; Tiwari, A.; Capretz, M.A. Data management in cloud environments: Nosql and newsql data stores. *J. Cloud Comput. Adv. Syst. Appl.* **2013**, *2*, 49. [CrossRef]
29. Gajendran, S.K. *A Survey on Nosql Databases*; University of Illinois: Champaign, IL, USA, 2012.
30. Gessert, F.; Wingerath, W.; Friedrich, S.; Ritter, N. Nosql database systems: A survey and decision guidance. *Comput. Sci. Res. Dev.* **2017**, *32*, 353–365. [CrossRef]
31. Brereton, P.; Kitchenham, B.A.; Budgen, D.; Turner, M.; Khalil, M. Lessons from applying the systematic literature review process within the software engineering domain. *J. Syst. Softw.* **2007**, *80*, 571–583. [CrossRef]
32. Dieste, O.; Grimán, A.; Juristo, N. Developing search strategies for detecting relevant experiments. *Empir. Softw. Eng.* **2009**, *14*, 513–539. [CrossRef]
33. Debnath, B.; Sengupta, S.; Li, J. Skimpystash: Ram space skimpy key-value store on flash-based storage. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, Athens, Greece, 12–16 June 2011; pp. 25–36.
34. Liu, S.; Nguyen, S.; Ganhotra, J.; Rahman, M.R.; Gupta, I.; Meseguer, J. Quantitative analysis of consistency in nosql key-value stores. In Proceedings of the International Conference on Quantitative Evaluation of Systems, Madrid, Spain, 1–3 September 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 228–243.
35. Mohamed, M.A.; Altrafi, O.G.; Ismail, M.O. Relational vs. Nosql databases: A survey. *Int. J. Comput. Inf. Technol.* **2014**, *3*, 598–601.
36. Fiebig, T.; Feldmann, A.; Petschick, M. A one-year perspective on exposed in-memory key-value stores. In Proceedings of the 2016 ACM Workshop on Automated Decision Making for Active Cyber Defense, Vienna, Austria, 24 October 2016; pp. 17–22.
37. Lamport, L. Paxos made simple. *ACM Sigact News* **2001**, *32*, 18–25.