Volume: 11 Issue: 04 Pages: 4321-4329 (2020) ISSN: 0975-0290

Recovery and Concurrency Challenging in Big Data and NoSQL Database Systems

Amira Hassan Abed
Department of Information Systems center
Egyptian Organization for Standardization & Quality, Egypt
mirahassan61286@gmail.com

ABSTRACT

Big data is becoming a very important concept nowadays as it can handle data in different formats and structures, velocity, and huge volume. NOSQL databases are used for handling the data with these characteristics as traditional database can't be used in managing this type of data. NoSQL database design is based on horizontal scalability with the concept of BASE which supports eventual consistence and data is considered in a soft state and basically available. Although NoSQL has a lot to offer when used in big data it is still not mature enough and faces some challenges including low join performance, concurrency control and recovery. Not only this but also it is very challenging for organizations to know which NoSQL data model to use and how does it fit with its organizational needs. This paper mainly displays the different NOSQL data models and the opportunities and challenges alongside with some techniques for handling these challenges.

Keywords**: Big data, NOSQL Data Model, Undo Recovery techniques, Disaster Recovery, Holistic Disaster recovery approach, Concurrency Control, synergy Systems techniques**.

Date of Submission: Jan 24, 2020 Date of Acceptance: Feb 01, 2020

I. Introduction

Big data is a new term that describes enormous volume of data which is generated with different formats and is considered data in motion and basically produced from everything around us [1]. It is more of a perception for collecting, consolidating and investigating the data and storing it [2]. Big data doesn't have a standard definition so experts in the field decided to use the expression Three Vs to express it where these Vs stand for Volume, velocity and variety [2]. The emerging of big data was a result of the limited capabilities of traditional database in managing the datasets with these characteristics [3]. The data involved in big data can either be structured, unstructured or even semi-structured, not only that but is also be accessed on daily basis from organizations and distinct users. Managing data with the velocity, volume, and variety existing in big data cannot be handled through the traditional databases way making NOSQL database a brilliant management substitute in big data.

Basically, NoSQL is a non-relational database management system that neither uses SQL query language for operation data nor is based on tabular relations that are extremely good in dealing with the large amount of data involved in big data. The main concept upon which NoSQL is based on is the notion of distributed storage of data alongside to the handling of parallel processing [1]. NoSQL is based mainly on horizontal scalability and there are a lot of different implementations, different systems and techniques in building a NoSQL database system. NoSQL databases mainly differ in the way data is stored and accessed they can be classified into many different types for example, wide-column store, document store, and value store each of which has its own characters and these three categories cover most of the techniques involved [2]. NOSQL doesn't account for ACID transaction which stands for Atomicity, Consistency, Isolation, and Durability. The concept of ACID is not working very well with distributed systems and as a result, it doesn't work well with the big data due to its 3Vs characteristics [3].

There are a lot of technical challenges in NoSQL in big data that could be addressed including query processing and handling complex queries which are compromised on the account of scalability, concurrency and recovery challenges. Not only that but also other challenges exist due to the fact that it is still considered a new technology and it is not mature enough. Another challenge is the ability of an organization to choose and decide on which data model to follow according it its own needs [2].

Organizations now a day are mainly founded on the NoSQL database and this lead to the need of protecting the data used and highlighted that there is a huge gap in data fortification. As for concurrency in the NoSQL systems it has been highlighted that the existing concepts of concurrency control don't work well with scaling even though the traditional techniques can be used it still reduces the performance of NoSQL systems [4]. Even though the concept of replication in big data NoSQL helps a lot in the process of recovery but yet a more robust recovery and backup approach is still needed. This is due to the fact that replication is creating a copy of the original data that could be corrupted before replication leading to loss of data. Backups is a very good strategy to accompany the replication process as it the process of taking copies of data at certain time where these copies could be restored to gain back the lost or corrupted data [5]. In order to understand the different concurrency, recovery and backup techniques we must first understand the different NoSQL databases.

Volume: 11 Issue: 04 Pages: 4321-4329 (2020) ISSN: 0975-0290

This paper is divided to many sections; first the work illustrates the different NoSQL models, their characteristics main functionalities and comparisons between different models. Section 2 addresses the state of art of NOSQL databases. In section 3 opportunities that NoSQL provide and challenges that are still many in NOSQL are introduced. The next section will cover different techniques, methodologies and approaches in NOSQL in different two very important aspects which is the recovery and concurrency

II. NOSQL DATA MODELS

Although there are various NoSQL DBs, but four data models have been addressed to be the most important, they are introduced below. Each model hold its specific properties although the differences between the introduced data models. In fact, all NoSQL databases were developed to support distribution features and scaling horizontally features. The key- value model will be introduced first.

A-Key-Value Store DB: Although it is considered as a simple NoSQL database but it also efficient and powerful DB. The data in this model is stored in two dimension, a string that act as the key and the actual data that act as the value, that result in generating a "key-value" pair. This led to values that are indexed using own specific keys for query processes; this concept is identical to concept for hash table. In other words, the store enables the users to retrieve the requested values based on the key specified to them. The key value model has the ability to process structured or unstructured data. It grants advanced concurrency and scalability in addition to rapid scans, however that it provides low consistency. The Key-Value store DBs have been used in building online shopping carts and high number commercial sites which it allow the feature of storing the requested users sessions. The well-known examples for this category are "Apache Cassandra", "Azure Table Storage (ATS)", and "Basho Technologies (Riak)". One of the **advantages** provided by Key-Value store database is its increasing the insert and read averages in comparison to traditional SQL database. This is fulfilled and obtained depend on provision many entries to the store as presented in the following example:

```
@db.bulk_save([
{"hot" => "and spicy"},
{"cold" => "yet loving"},
{"other" => ["set","of","keys"]}
])
```

B -Column Oriented/wide-column Store DBs – In this category of NoSQL database, the columns are realized and determined in relevant to each row in state of predefined by the table organization owned uniform sized columns for each tuple. Such these stores introduce a two-dimension gross/aggregate organization, a key and a row gross that is defined as a set of columns. This allows any column to be added to any particular row, and in this case the rows can own a lot of various columns. In other words, each row possesses a number of different columns that were maintained and stored. It also is able to maintain data in tables like segments of data columns. Data is presented like "row-oriented" where each row is a gross or as "column-oriented where in this state every column segment introduced a specific record type. Each key is typically connected to one or more specific numbers of columns and a key for every column segments is used for its capability to fast and high data retrieval processing with little input or output activities thus it offering and obtaining rapid extensive a performance. These databases moreover support the feature of high scalability as well as mainly store data items in significantly distributed architectures. These Wide column DBs is appropriate to be used and have important role when they included within applications specified to data mining and analytic tasks with Big Data. The well-known examples of these column oriented stores providers are "Facebook advanced performance Cassandra", "Apache Hbase", and "Hyper Table".

C- Document Store Databases – this type of NoSQL DB typically spins the main key-value DB category concept and stores complex data structure included and stored in different document form such as XML / JSON. A document store DB is considered schema-less in which every specific document possess ability to stores various needed fields under any defined length. Documents are accessed and also identified based on a particular unique key that it possible to be simple string form, URI string type or also a path string. Document databases are more complex databases but grant advanced features and capabilities like raising performance, horizontal scalability and schema flexibility these all properties help in storing virtual manner any organization requested by any particular application. Document oriented DBs are considered proper for many systems such as content management and blog system. Many examples for document oriented DBs; the common ones are "10generation MongoDB", "Apache CouchDB", and also "AWS DynamoDB".

D- Graph Store or as known "Graph database" this type based mainly on the defined relationships between the existing data items. It follows moreover the graph theory technique to maintain the data and optimizes/enhances the search processes through enables an ***index free adjacency*** mechanism. It is developed typically for data whose relationships are exactly full structured by graph organizations composed of sets of property, node, and also edge. A node act as a particular object, an edge determined the explicit relationship between the objects and the property is the node hold over the other end side of specific relationship. According to the index free adjacency approach, each node mainly involved a pointer where it directly and forthright refers to the specific adjacent node as presented in the supported **Figure 1.**

These stores provide extensive capabilities such as advanced performance, compliance to ACID base and supporting rollback property. These databases are typically convenient and fit to building applications specified to social networks, bioinformatics, as well as the cloud services. The common applications for Graph DBs are Orient DB and Apache Giraph.

Volume: 11 Issue: 04 Pages: 4321-4329 (2020) ISSN: 0975-0290

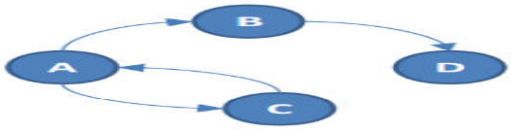

Figure1: Graph algorithm.

The work now present a significant comparison around a number of the most prominent key-value, document and wide-column stores. The proposed comparison show how SQL and NoSQL DBs are developed to support many different requirements: RDBMSs provide an unmatched level of functionality whereas NoSQL databases excel on the non-functional side through scalability, availability, low latency and/or high throughput. However, there are also large differences among the NoSQL databases. Riak and Cassandra, for example, can be configured to fulfill many non-functional requirements, but are only eventually consistent and do not feature many functional capabilities apart from data analytics and, in case of Cassandra, conditional updates. MongoDB and HBase, also they grant stronger consistency and high advanced functional features such as peruse queries and filter queries, however they not enable read and write availability during partitions operation and resort to offer optimizing read latencies. Redis, which considered the only not partitioned system included within this comparison apart from MySQL, it represent an explicit group of trade-offs focused on the ability to maintain high throughput at low-latency supported with in-memory data structures and asynchronous master-slave replication. An extra detailed and wordy comparison around each prime system's capabilities is introduced in the shown **Table 1**.

III. State Of Art

NoSQL databases shifts from the ACID properties which are the base of relational databases and adopts the concept of BASE (Basically Available, Soft state, and Eventually Consistent), along with adopting the CAP theorem [6]. The CAP Theorem which stands for Consistency, Availability, and Partitioning was introduced by Eric Brewer, where he stated that a system can only account for two of these three characteristics either availability and partitioning (AP), or consistency and availability(CA) or availability and partitioning (AP) [7].

Table1: the comparison between Mongo DB, HBase, Cassandra, &Ria

Dimension	MongoDB	HBase	Cassandra	Riak
Model	Document	Wide-column	Wide-column	Key-value
CAP	CP	CP	AP	AP
scan performance	High(with appropriate shard key)	High (only on row key)	High(using compound index)	N/A
Network latency	Configurable: nearest slave, master	Designated region server	Configurable: R replicas contacted	Configurable: R replicas contacted
Durability	Configurable: none, WAL, replicated (”write concern”)	WAL, row-level versioning	WAL, W replicas written	Configurable: writes, durable writes, W replicas written
Replication	Master-slave, synchronicity configurable	File-system-level (HDFS)	Consistent hashing	Consistent hashing
Sharding	Hash- or range-based on attribute(s)	Range-based (row key)	Consistent hashing	Consistent hashing

Volume: 11 Issue: 04 Pages: 4321-4329 (2020) ISSN: 0975-0290

Consistency	**Master writes, with quorum reads linearizable and else eventual**	**linearizable**	**Eventual, optional linearizable updates ("lightweight transactions")**	**Eventual, client-side conflict resolution**
Atomicity	**Single document**	**Single row, or explicit locking**	**Single column (multi-column updates may cause dirty writes)**	**Single key/value pair**
License	**GPL 3.0**	**Apache 2**	**Apache 2**	**Apache 2**

Unlike the ACID, BASE supports the concept of eventually consistent which allows improved performs for NoSQL as all the replicas of the data will reflect the performed transaction as time propagates it doesn't have to be reflected instantly [7]. This will lead us to explore the different techniques used in the NoSQL database that make them what they are; these techniques are Sharding, replication and query processing [2].

Sharding technique helps NoSQL databases to achieve excessive scalability by partitioning or sharding the data thru separate nodes either by following the range-sharding, hash-sharding, or group-sharding. As for the replication it is creating copies of the data that can be accessed at any site to avoid any failures that might result for the large volume and velocity of data in the system. Replication can be classified into synchronous, asynchronous, primary copy, or even update anywhere all of this classification reflects the technique by which all the replicas will eventually be the same reflecting the same copies after transactions have been performed [2].

Last but not lease is the Query processing technique in NoSQL, it is very similar to query processing in distributed database systems and it required a "Query planning" task to help in decreasing the execution cost as there are many replicas [2]. The query planning is very vital especially when executing aggregation and joins. Deciding which NoSQL to choose within an organization is a very tricky and challenging task. A lot or researches and surveys have been conducted to try and create a road map for organizations on how to find the best NoSQL database to use. NoSQL databases are categorized into different types: key-value, document, wide-column store and graph database [8].

Each data type uses a combination of different sharding, replication and query processing. The key- value store is the simplest form and considered assembles model, where it is basically a prearranged collection of unique keys with key-value pairs only executing create, read, update, and delete (also known as CRUD) operations [2]. The Document store is based on the key concept of the key-value store but also only works with semi-structured arrangements such as Jason and it allows us to work on parts of the document and work on it including databases like Couch Band MongoDB. While the wide-column stores model is based on rows and column keys and can be found in Hbase, Cassandra and Big table databases. The row column has to be coupled with the column key for a successful retrieving process [2]. The Graph NoSQL database like Giraph and Neo4j represent where the representation of objects and relations is in the format of a graph with nodes and objects and relations as edges [8].

On the side, for recovery and backup techniques Kathpal & Sehgal [9] indicated that as NoSQL DBs are receiving high significance in business applications industry, but they suffer from unique limitations in the areas related to recovery and backup. The authors identified these drawbacks as, "cluster consistent backup and emphazing on resolve free restoring, efficient backup space/storage, and finally, topology oblivious backup and recovery." As a result, the authors attempted to introduce the **BARNS** as the solution to face the above identified recovery and backup drawbacks in the NoSQL DBs resident on shared storages. Their BARNS solution is leveraging "light-weight" snapshots that generating copy-on-write and writable snapshot capabilities related to shared storage. They leverage NoSQL DBs approaches by providing cluster consistency through backup process, ridding of identical replica copies and are resilient to topology modifications and changes through recovery and backup processes. BARNS save about 66% of storage spaces by reducing two replica copies for MongoDB (master- slaves) and Cassandra (master-less). But according to [10], they have reported that the NoSQL databases have developed as respond to continuous increasing in datasets volumes. However, these NoSQL databases solutions do not support out of the box the necessary backup and recovery features needed by many big data modes. So they address this aspect through introducing an extensive disaster recovery method proposed for big data NoSQL tasks. This approach is able to be employed for least recovery point and time aims at minimal costs. Their approach is highly scalable, according to open sources combinations and significantly convenient to every document-based NoSQL, as allowed it to be more attractive for various big data workloads.

Volume: 11 Issue: 04 Pages: 4321-4329 (2020) ISSN: 0975-0290

Another novelty in this proposed backup and recovery approach represented through the backup and recovery tier that mainly prolong bidirectional replicating, in which the replication of data performed by replicating the data from the database into the steady storages. Also, David & Miguel [5] point that according to the NoSQL databases simplicity and flexibility these databases solutions become very popular among web application developers. But in general these NoSQL databases usually provide basic backup and recovery mechanisms that enable restoring databases from crash actions, but not to remove undesired operations caused by accidental or malicious actions. As a result, the authors proposed NOSQL UNDO as a recovery method and tool in which it enable database administrators to delete any undesirable effect caused during the above actions by "undoing operations", resulting in a consistent system mode. NOSQL UNDO enhances the log in addition to snapshot tasks built-in NoSQL databases, and is allowed to undo processes as they are present in the logs. This is, as far as we know, the prime recovery service grants these capabilities for NoSQL databases. The experimental findings for this study with MongoDB reveal that it is possible to undo an individual operation in a log with 1,000,000 entries in around one second and to undo 10,000 incorrect operations in less than 200 seconds.

IV. Opportunities and Challenges

NOSQL Systems added tremendous value to different type of systems especially systems provide real time services or near real time services; it represents the storage layer for huge amount of various types of data in those systems and helps them to provide low latency to the requests of the user with high throughput against massive distributions and high fault tolerance. NoSQL DB store data with their raw state of these data without making any transformation over that data, unlike it handled in relational Database which gives us the power to discover hidden pattern, modules and increasing information business value.

NoSQL databases ended the complexity of dealing with SQL queries or nested queries instead provide very simple ways to retrieve the data. NoSQL uses cheap commodity servers and hardware to deal with these data volumes as a result the Cost is quite less compared to relational database which need very expensive hardware to prevent the bottleneck and support fault tolerance. NoSQL data base depends on Base which states for "basically available soft state eventually consistent" the management of the change had always been a struggle with RDBMS which request high consistency that need excessive management and reducing of the provided service levels and unwanted downtime. NoSQL however is more relaxed "soft state" and can alter itself easily to suit these changes [2].

Regardless of the tremendous opportunities that the NoSQL provides for business in the area of big data it still faces a lot of challenges appeared in recovery, consistency and query processing. Achieving any level of consistency in NoSQL DBMS accounting for the ACID properties is considered one of the challenges that faced alongside with acquiring speedy performance when accessing distributed data stores. Even though NoSQL supports relaxed consistency still the existence and handling of different replicas lead us to the challenge of concurrency control and which replica is to be accessed and how can we handle the concurrency. Applying concurrency control mechanisms in the NoSQL databases shouldn't come on the expense of the query processing performance and performance.

There are different concurrency control techniques allowing read and write including optimistic concurrency control, multi-version, snapshot isolation and others but the most important factor to account for is how will they affect our performance. In addition to that the nature of NoSQL leads the researchers to an extremely vital aspect which is the recovery. Even though the existence of replicas might mistakenly represent the illusion that data will not be lost but that's not the case as they don't account for out of the box capabilities for restoring, recovering and necessary backup techniques to account for different catastrophic events [5].

Recovery techniques used in most NoSQL database are considered modest mechanisms and are basically only based on logs whether local or global logs and snapshots that come in use when server crashes for instance [10]. The existing techniques are acceptable but they don't account for the effect of faulty operations, whereas if an executed operation lead to tainting the database the restoring of old version from the snapshots will help get back the data but will also result in losing operations that have been performed after it was taken leading the data to be in an incorrect state and not reflecting the actual correct version [10].

V. Recovery and concurrency techniques

In this section the study discuss different techniques of some of the addressed challenges that discussed in the previous section, that focusing on the NoSQL recovery and concurrency challenges. Although the difference in NoSQL model systems but at the end they have the same system architecture which consists basically of minimum two shards, then data are sliced to smaller parts at theses shards which are a set of servers with the same copy of data "replicas". Replica can provide the basic recovery to the systems. It works as a fault tolerance. The replication servers differ in functionality. The primary server keeps data consistent inside a shard where routers servers split data correctly and divide records in shards. Routers servers are the main part to communicate with the applications. The more routers servers the more the availability of the systems are increased. Database couldn't be accessible of the all the routers are dead; it affects the performance and availability [5].

Recovery in NOSQL systems depends as well as on the" logging mechanisms which records database requests and take periodic snapshot, permitting the recovery of the system in case of failure." [5]. Local logs which also called "diary" used to fix server from the unexpected

Volume: 11 Issue: 04 Pages: 4321-4329 (2020) ISSN: 0975-0290

failure which writes every operations to disk, after the failure, fail- over should be applied to return the different replica to work state In some times this log is not enough for recovery since it only have recent operations, so it is efficient to use a snapshot (a full copy of the server in a previous time). Each individual server or replica has it is own logs so in case of all servers failures, it is very hard to recover the entire database using local logs of each server or replica. The global logs are providing very important rule to keep data consistency across different servers by delivering the requests in the same orders to all servers. Global logs provides guaranties cause of saving each request they receives that in case of some requests lost or couldn't be delivered in the same orders the ability to check the global log and execute that request.

NoSQL Undo recovery technique is one of the very important useful techniques & the first recovery service that offers these Capabilities for NoSQL databases with its architecture. It is a client side that only accesses a NoSQL database instance when the database administrator needs to delete the influence of some operations from the database, because they are malicious for example. No need for the client to be connected to the server in run time since it uses the database built-in logs to do recovery. It also does not need an extra server to act as proxy where it only uses the built-in log and snapshots of the database to achieve recovery. No need also for extra meta-data or modifications to the database distribution or to the application using the database. NOSQL UNDO is also the first that supports such a replicated and shared architecture system type database which the study discussed in earlier sections. In opposite the Rollback technique is removing of the detected incorrect operation and revert the entire database to a previous point before the execution of the incorrect operations which cause a challenging drawback that each correct operation executed after the point in time to recover is completely lost. Two main methods of NOSQL UNDO to delete the effects of incorrect operations return back the database into correct state. Full recovery and Focused recovery. Both methods take input as list with operations to be undoing." [5]

- Full recovery: The full recovery method is much simpler then focused recovery method and very effective. It loads the most recent snapshot of the database, and then updating the state by implementing the rest of the operations, which were previously recorded in a log. It is better to use the full recovery method with big set of operations to undo, where it requires executing every correct operation in the log after the snapshot.
- Focused recovery: The main concept behind Focused Recovery is instead of recovering the entire database just to delete the influence of a small set of incorrect operations, only compensation operations are implemented. A compensation operation is an operation that corrects the effects of a faulty operation. Efficient more with the small number set of the incorrect operations.

Another huge challenge addressed in NOSQL systems is recovery from disaster failure [10]. Replication and different backup capabilities can only support recovery in local storage only against crashes not human errors or malicious security issues [9] in this section, the study provides the Holistic backup/restore approach to handle the disaster failure inside the NOSQL systems. This disaster recovery approach can be applied to any document-based NoSQL database. It is very highly scalable and fixable and based on open source components. "The holistic approach achieved regarding these main concepts The *Recovery Time Objective (RTO) that* defines specifically the maximum allowable period until application service is restored after a failure event [11] and The *Recovery Time Objective (RTO)* define specifically the maximum allowable period until application service is restored after a failure event. So this approach can provide low recovery point and time objectives at low costs [11].

The holistic backup store approach depends on three main features. These unique features are Trigger-based backup approach, High parallelism and High modularity. It also consists of 4 tiers or models which are a) The load-balancer module b) the backup and restore management module c) the monitoring module, implemented using ZooKeeper [12], and d) the stable storage module, implemented using Hadoop File System (HDFS) [13].

The holistic approach provides two backup modes of operation: 1) ***on-demand***; and 2) ***continuous*** (the default mode). The load balance tiers consist of two components, the main load balance which is small server sends the backup/restore REST requests to a random (stateless) worker out of the backup and restore management tier. And one or more secondary load balancer. The zookeeper chooses the main load balancer in case of failure. The main load balancer main functionality is handling all requests at certain points. The BRM is a fixable tier composed out of stateless workers that are performing backup/ restore/ change requests received from the load-balancer tier. When the BRM worker receives a request it instantly replays with positive status and serves the request. When the BRM worker ends with serving the request it deletes the request from its queue. Zookeeper module it provides higher level services, these services include the following: Group membership, Metadata storage; and Leader election. These services are used by the load-balancer and backup/restore tiers. The leader choosiness is picked by the load-balancer tier for electing the primary load balancer server. The metadata storage is serving the Queue management functionality: each backup/ restore worker has its queue keeping all the not finished requests served by this worker. Backup Version provide the holistic approach in different document level versions with different period time base, daily, weekly, monthly. Recall that we influence the version number attached to each database document. This number reflects the state of the document in a certain time.

Volume: 11 Issue: 04 Pages: 4321-4329 (2020) ISSN: 0975-0290

The first backup is asking to takes a full backup of the entire database. The versions and the states of the documents are maintained by the database. The first backup is treated as the 'baseline' for other changes. In the continuous backup mode, upon every database change, the primary load balancer is informed with nonfiction contains the new/changed document attached with a new (increasing) version number. After this point, the system saves and backups only a changed document since the last 'baseline'. These changes from the last 'baseline' are taken as 'baseline delta'. The system configuration counts how many 'baseline delta' to preserve before creating a new 'baseline'. Daily 'baselines' turned to be weekly 'baselines' and weekly 'baselines' turned to be monthly 'baselines'. This scenario is shown in Fig. 2.

On the other side, the study now discuses the concurrency control that addressed as challenged area in the NOSQL. According to [4], they handled concurrency control issue in NOSQL DBMS by applying the Synergy system technique. The main purpose of proposing the synergy system is to improve performance in NOSQL databases using Materialized views (MVs) and a specialized system for concurrency control based on the top. Not only has that but also helped offering a scalable data management offering good ACID semantics. The synergy based systems first foundation is having quicker join performance to increase the disk operation based on materialized views.

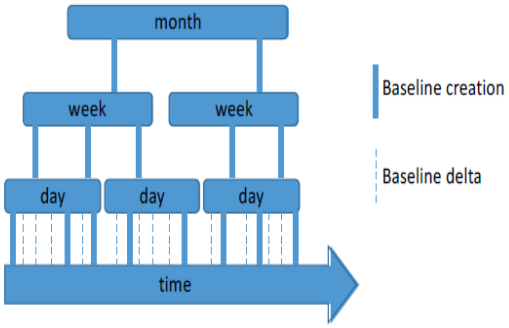

Figure 2: "backup versioning: the new baseline gather the baseline delta into a tt less granular time slice" source [10]

Generally, Materialized views (MV) is an object of the database where the query results are held and basically are locally replicated data that is remotely local located. The addition of MV to the NoSQL capabilities is purely aimed to enhance its performance but alone it will lead us to facing the consistency issues that lead the authors to account for concurrency control in their model as multi-versioning or locking techniques to help resolve this issue. They decided to resolve the consistency issue applying a layer of concurrency control above the NOSQL database with its selected MV involving only a single lock to be seized in a transaction using a hierarchal locking mechanism.

The first step in the formation of their model was how to reach the best way to integrate MVs into the NoSQL data stores without compromising its consistency. In the start, they considered the implications of materialized views, the lock number and granularity following them was the view selection challenges. They settled on using a lock-based concurrency control mechanism since it was proven that in NoSQL multi versioning the acquiring and checking of additional rows timestamps hugely reduces the performance. The lock number should be as minimized as possible as the lock acquiring process can be highly costly in the presence of the selected MV in the NoSQL leading them to choosing single lock concurrency control mechanism. They based their model on schema based workload driven views that work best with the criteria in mind. The following figure illustrates more the selection choices they followed in their design.
The synergy system architecture consists of the following: HBase layer, client and Transaction layer.

- **HBase layer:** This layer includes HDFS, Zookeeper and the HBase components and it acts like a layer of distributed data storage [14].

Volume: 11 Issue: 04 Pages: 4321-4329 (2020) ISSN: 0975-0290

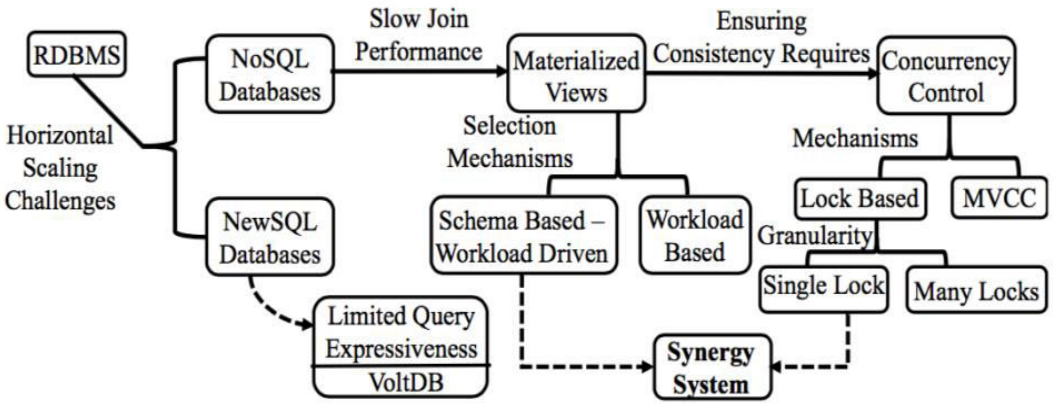

Figure 3: Design choices and decisions in the Synergy System [4]

- **Client layer**: The read and write statements are executed in the workload where a read request is directly sent to the Hbase layer and the write request to the other layer which is the transaction layer and then a synchronized response is received.
- **Transaction layer:** The main purpose for including this layer is accounting for the ACID transaction support along with the HBase layer. It consists of a Master node and 1 or more slave nodes for handling concurrency in the layer which is a scalable and distributed layer accounting for fault tolerance. In this layer a manager transaction is assigned to each slave node for handling recovery and durability by implementing a write a head log which is stored in the HDFS in the Hbase layer. When it receives the write transaction request the first step is to assign an id and add both the statement and the id in the write a head log then when the transaction is executed the response is sent back to the client. The existence of the Master node is solely for the purpose of spotting any failures within a slave nod and assigns a new slave node to perform and execute the task of the failed one.

The synergy system accounts for light weighted lock technique by using a single lock per its write operation by using a logical lock and a physical lock through the implementation of a lock table. In the system the base tables, indexes and views are automatically updated for each transaction procedure. Finally, the system is set to provide a transaction isolation level for both the read committed and the ACID semantics.

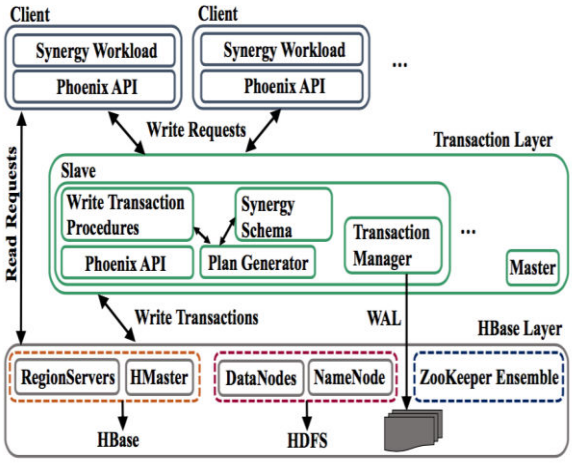

Figure 4: Synergy System Architecture Overview [4]

The authors of the paper conducted experimental comparisons with other systems in terms of the concurrency control mechanism and the selection of Materialized views and it was obvious that the concurrency control in the proposed synergy system results in a much better output in terms of writes and read response time and performance. Not only that but also the response time when using them is significantly higher than other systems except for when it comes to the join the performance is slightly slower than the VoltDB which doesn't use any MV and is based on single threaded partition processing. They also created a qualitative comparison of NoSQL, NewSQL, and Synergy in terms of scalability, query expressiveness, transaction support and disk utilization displayed in the next table. The synergy system yet still has some limitations as they are limited to single SQL transaction statements and key-foreign key

Volume: 11 Issue: 04 Pages: 4321-4329 (2020) ISSN: 0975-0290

equijoins. Also the level of separation is only limited to the read commits.

VI. Conclusion

Throughout this survey paper, NoSQL database management systems and their different data models and techniques used in building up these data models including sharding or partitioning, replication and query processing, were discussed. Upon understanding how big data NoSQL systems work a few issues were raised including recovery and concurrency control. The work discussed the techniques handling these issues like a holistic backup approach for handling the recovery issue and Synergy System which is a technique for handling concurrency control in Hbase NoSQL and NOSQL UNDO. The survey also displayed the benefits and the shortcomings for each of these approaches.

Table 2: Qualitative comparison of NoSQL, NewSQL and Synergy systems [4]

	Scalability	Query Expressiveness	Transaction Support	Disk Utilization
NoSQL (HBase)	Linear scale out	SQL	ACID with Snapshot Transaction Isolation	Higher than NewSQL
NewSQL (VoltDB)	Linear scale out	SQL with joins limited to partition keys	ACID with Serializable Transaction Isolation	Lowest
Synergy	Linear scale out	SQL with MVs limited to Key/Foreign-Key joins	ACID with Read Committed Transaction Isolation	Highest

References

[1] Amira H., Mona N., & Walaa S. (2019). The future of internet of things for anomalies detection using thermography. International Journal of Advanced Networking And Applications. 11(03). P: 4298-4304.
[2] F. Gessert, W. Wingerath, S. Friedrich & N, Ritter, NoSQL Database Systems: A Survey and Decision Guidance. Springer-Verlag Berlin Heidelberg. (Nov. 2016)
[3] M. Razu, M. Khatun, M. Asraf, & K. Sundaraj (2018) A Literature Review On NoSQL Database For Big Data Processing International Journal Of Engineering & Technology, 7 (2) pp.902-906
[4] A. Tapdiya, Y. XueAnd D. Fabbri, (2017). A comparative analysis of materialized views selection and concurrency control mechanisms in NoSQL databases. IEEE clusters conference.
[5] D. Matos & M. Correia. NoSQL Undo: Recovering NoSQL Databases by Undoing Operations, IEEE 15th International Symposium on Network Computing and Applications (Nca), (Dec.2016).
[6] A. Hamed Al Hinai. A Performance Comparison of SQL and NoSQL Databases for Large Scale Analysis Of Persistent Logs, Http://Www.Teknat.Uu.Se/Student, 2016
[7] A. Khan Zaki. (2014). NoSQL Databases: New Millennium Database for Big Data, Big Users, Cloud Computing and Its Security Challenges, IJRET: International Journal of Research In Engineering And Technology, (3)3. P: 403-409.
[8] M. Teresa, A. Ogunyadeka, M. Younas, J. Tuya & R. Casado. (2017). Transaction processing in consistency-aware user's applications deployed on NoSQL databases. Human–Centric computing and Information Science. (7)7.
[9] A.Kathpal and P. Sehgal, (2016). BARNS: Towards Building Backup and Recovery for NoSQL Databases. Proceeding, Hotstorage Proceedings Of 9thUsenix Conference On Hot Topics In Storage And File Systems.
[10] A. Abadi, A.Haib, R.Melamed, A.Nassar, A. Shribman, & H. Yasin. Holistic Disaster Recovery Approach For Big Data NoSQL workloads, IEEE International Conference On Big Data. (2016).
[11] K. KEETON, C. SANTOS, BEYER, D. J. CHASE, & J. WILKES. Designing for disasters. In Proceedings of the 3rd USENIX Conference on File and Storage Technologies (Berkeley, CA, USA, 2004), FAST '04, USENIX Association, pp. 59–62.
[12] Apache ZooKeeper - Home. https: //zookeeper.apache.org.
[13] Apache Hadoop. http://hadoop.apache.org.
[14] "HBase. [online]. available: http://hbase.apache.org/."

Author Detail:

Amira Hassan Abed received her B.Sc. degree in Information Systems from Helwan University, Egypt, in 2009 and her M.Sc. degree in Information Systems from Faculty of Computers and artificial intelligence, Helwan University, Egypt, in 2017. She has been working toward the Ph.D. degree in Information Systems, Faculty of Computers and artificial intelligence, Helwan University, since September 2017. Her main areas of research are machine learning, Internet of Things (IoT), bioinformatics, Business intelligence, Customer relationship management (CRM).