RESEARCH ARTICLE OPEN ACCESS

A Novel Big Data Migration Framework With Reduced Database Downtime

V. Rathika[1] , Dr. L. Arockiam[2]

[1] *Research Scholar, Department of Computer Science, Mother Teresa Women's University, Kodaikanal, Tamil Nadu, India*

[2] *Associate Professor, Department of Computer Science, St. Joseph's College, Trichy, Tamil Nadu, India.*
Corresponding Author: V. Rathika

ABSTRACT
This paper introduces novel big data migration architecture with its data fundamentals description, processes and an interface related to big data migration. It describes a homogeneous novel framework for implementing the big data migration between relational databases and non-relational databases on the on-premise server using map reduce. The main goal of this big data migration framework is to migrate the massive data stored in relational database to NoSQL database parallelly with minimal downtime. Another objective is to increase the post-migration process after performing live migration. The proposed framework helps migration of big data from a relational database to a distributed platform quickly and easily.

Keywords-NoSQL Databases, Big Data Migration, Denormalization, Association Rule Mining and Map Reduce.

Date Of Submission: 25-07-2019 Date Of Acceptance: 06-08-2019

I. INTRODUCTION

With the extensive spread of Cloud Computing, Internet of Things and Big Data Technologies, the amount of data is growing rapidly. Due to this data deluge, the requirement of distributed environment is not substantial for the computing technologies. For many years, a majority of corporations and organizations have typically used Relational Databases to accumulate and process their data. Even though, in this digital era, still the structured data exists and accumulate with the business development in settlements [2]. With the rapid growth of data size, the storage becomes a bottleneck. Due to its huge size, the analysis and processing have led to performance degradation. These issues are needed to be solved in organization level applications.

To solve the above-mentioned issues, distributed platform is to be developed for storing and computation technologies. Generally, this distributed platform denotes to a software platform that builds data storage, data analysis, and computations on a cluster of several hosts. The main objective of this distributed platform is revolving around the distributed storage and distributed computing. In terms of storage, it is theoretically possible to expand capacity indefinitely, and storage can be dynamically expanded horizontally with the increasing data. In terms of computing, some key computing frameworks such as Map Reduce can be used to perform parallel computing on large-scale datasets to improve the efficiency of massive data processing. Therefore, when the data size exceeds the storage capacity of a single-system or the computation exceeds the computing capacity of a stand-alone system, massive data can be migrated to a distributed platform. The ability of resource sharing and collaborative computing provided by a distributed platform can well solve large-scale data processing problems. This paper focuses on putting forward a standard for implementing a big data migration framework through web access via Internet and considering how to help users more easily and quickly migrate the massive data from a traditional relational database to a cloud platform from multiple requirements.

There is a lack of standards for NoSQL databases, especially in terms of data migration from the SQL database to NoSQL database. This deficiency causes needs for a generic migration framework that includes all stages of data migration, this research focuses on it. Many frameworks have developed to solve the problem with a different solution, characteristics and properties [3, 4, and 5], the current implementation of the framework for data migration supports three different data stores: document, columnar and graph [4].

The outline of this paper is as follows: the second section describes a brief study of the research literature by underliningthe relatedcontributions of this study. The third section describes the objective of the proposed framework. The framework methodology and technique are described in the fourth and fifth section respectively. The summary of this contribution is concluded in the sixth section.

II. LITERATURE REVIEW

The problem of data migration from SQL databases to NoSQL databases is already being studied by different authors, but with the many NoSQLdata store variants currently available, it is almost impossible to migrate data between each NoSQL database. The most widely used classification method is based on how they store and manipulate data. This categorization produces two categories: core NoSQL systems (i.e. key-value, document, column, and graph) and softcoreNoSQL systems (i.e. object, multivalued, XML, grid, and cloud, etc.) [5].

There are two main methodologies identified that are used to deal with data migration problems which focuses on the representation or the translation. For the translation, there are two general approaches when trying to migrate data: direct mapping and intermediate mapping. Direct mapping, that is, data from the source database are translated into the data structures of the target database, while intermediate mapping, that is, data are translated into an intermediate format and then from this into the final one [6].

This paper focuses on the translation. Scavuzzo et al [3] introduced solutions to data migration by maintaining strong consistency and secondary indexes in migrating data, but still limited to the NoSQL column (column to column). Shirazi et al [5] propose a design pattern model in migrating data for columns to a graphical database (column to graph) and vice versa. Bansel et al [4] develop a framework for migrating three data stores databases: column, graph, and document using both approaches (direct and intermediate mapping).

Cloud portability means the ability to seamlessly move data and application services from one cloud provider to another [5]. Portability for heterogeneous public clouds has long been considered an open problem, since distinct cloud providers model their data structures and implementations differently [3] and data-intensive applications typically require sustainable performance and scalability [6].

Thalheim and Wang [7] state that in order to migrate the data, one need to have a thorough understanding of the data source such as data availability and data constraints since different data sources are designed using different modeling semantics. Furthermore, the data source may have inconsistent, duplicate or inaccurate data. Therefore, it is essential to migrate the data from one data store to another to appreciate the growth and requirements of the deployed application [8].

Furthermore, the framework includes migration algorithms, migration models, and migration schemes. The framework is tested by migrating a certain amount of data from one data store to another data store (i.e. Document to Column) in the actual environment. In addition, evaluation is also performed by comparing the proposed framework with the current existing framework using certain parameters [4].

NoSQL data stores have long been considered highly available, scalable, and flexible to handle large amounts of data and transactions over many distributed servers [9]. NoSQL implementations arguably provide better performance than RDBMS, particularly for simple read/write operations [10]. In addition, unlike the traditional RDBMS, NoSQL does not require fixed structures for storage [1]. While some authors have recently recognized the importance of automatic translation of SQL environments to NoSQL [11], scant research has been devoted to enable the seamless interoperability between diverse NoSQL implementations.

Consequently, most of the large-scale websites and cloud computing-based enterprises applications such as eBay, Twitter, Amazon, Google and Facebook are adopting proprietary NoSQL-based data stores [12].

Further, we present the various NoSQL types which model the Consistency, Availability, and Partition Tolerance (CAP) theorem [13]:

- **Key-value Stores:** a data model based on keys-values which is easy to implement, but inefficient in updating and querying the part of a value.
- **Document-oriented databases:** semi-structured documents which are stored in JSON format. They support efficient querying and manage the nested values with associated keys.
- **Column family databases:** an efficient data model to store and process large amounts of distributed data over multiple machines.
- **Graph databases:** enable the scalability across multiple machines and allow data-model specific queries.

It is critical to study the heterogeneity among the three essential NoSQL solutions: document, columnar and graph. Most of the document databases exhibit Consistency and Partition Tolerance (CP) and master-slave replication. The columnar NoSQL databases exhibit Partition Tolerance and Availability (PA) and use peer-to-peer replication to easily handle large amounts of data [14]. In contrast, the graph database is the only NoSQL solution which fits with modern workload growth and is highly performance-oriented [15]. The graph databases enable a fast traversal among nodes, and also can easily model complex relationships [16]. Therefore, companies such as Google, Facebook and Twitter adopt graph databases as they particularly fit to model social networks interaction as the graph data model can describe both static and dynamic relationships. Furthermore, the graph databases are used across a wide range of applications domains such as airlines, health-care, gaming, and retail [2]. Ideally, a data migration mechanism should require little assistance from a user and should be automatic. The authors [3] had introduced a solution for data migration between NoSQL columnar databases, in particular from Google App Engine to Azure Table and vice versa.

Shiraxietetal. [5] had proposeddesign pattern to migrate data from column databases to graph database. Wang et a. [7] had identified three stages for data migration: extraction, transformation and loading. First, the data is extracted from the source data model. Then, the extracted data source is transformed into the new data structure. This stage may require validation, mapping and cleansing of data. Lastly, the transformed data is loaded into the target data source.

The authors [3] had proposed two main strategies for data migration from source database to a target database such as Direct Migration: the translation of source database into the target database without any intermediate stage and Intermediate Migration: first, the data from the source database is translated into an intermediate format, and then the intermediate data is transformed into the data structure supported by target database.

III. LARA FRAMEWORK

The proposed LARA framework has six main phases which are presented in the Fig.1 are as follows:

1. Select:

This is the initial phase which identifies the source and target database for the big data migration to be performed. For this research work, MySQL database is selected as the source database on behalf of relational database and MongoDB is selected as the target database on behalf-of the document-oriented database.

2. Prepare:

The second phase of the proposed LARA framework is preparing the huge amount of homogeneous data which are to be migrated. The proposed ABS_DET$_{GB}$algorithm analyzes the source database attributes using the denormalization and association aware techniques. This technique is explained in the succeeding section. Based on the analysis, it prepares the collections and fields in the target database.

3. Extract:

The third phase is initial stage of the big data migration. It extracts the big data from the source database based on the information received from the ABS_DET$_{GB}$algorithm.

4. Migrate:

This is the key phase of the proposed LARA framework. The data extracted from the source database is migrated to the target database using the MapReduce technique. It is described in the section 6.5.5.

5. Load:

The fifth phase of the proposed LARA framework is to load the transferred data from the source database into the target database using the key-value pair technique which is explained in the section 6.5.5.

6. Validate:

The final phase is the validation phase which is to validate the overall framework and ensure the performance effectiveness and quality of service of the big data migration.

Each phase may have different actions reliant on the data migration that will be executed. Moreover, the framework comprises of query translators, query logs and big data migration algorithms. The proposed LARA framework is evaluated by migrating the big data from row-oriented database to the document-oriented database in the actual environment. Furthermore, evaluation is also performed by comparing the LARA framework with the existing framework by using certain constraints.

The massive data migration migrates all the data based on the Associate Aware Technique and Query Translator. The database migration is performed lively, where the users are able to access the data from the target database instantly. MongoDB does not support the join operation in the data retrieval. This affects the post-migration process and becomes a big challenge for the researchers and users. To increase the post-migration, the proposed graph based denormalization method is hybridized along with the proposed associate rule mining technique.

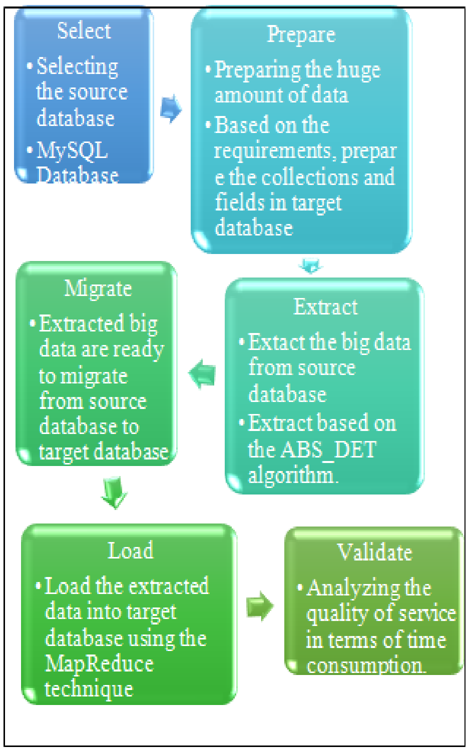
Select
• Selecting the source database
• MySQL Database
Prepare
• Preparing the huge amount of data
• Based on the requirements, prepare the collections and fields in target database
Migrate
• Extracted big data are ready to migrate from source database to target database
Extract
• Exact the big data from source database
• Extract based on the ABS_DET algorithm.
Load
• Load the extracted data into target database using the MapReduce technique
Validate
• Analyzing the quality of service in terms of time consumption

Figure1. Process of LARA Framework

The associate aware technique is built inside the proposed framework. Fig. 2 represents the LARA framework.

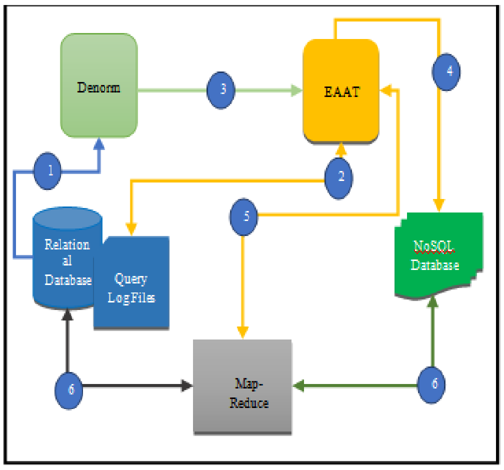

Figure 2. LARA Framework

Note:

- DET_{GB} – Graph based Denormalization Technique
- AAT – Associate Aware Technique
- ABS_DET_{GB} – Attribute and Graph Based Denormalization Selector

The following is the process of the proposed LARA framework:

Step 1: De-normalize the relational database; the information of the de-normalized attributes is stored in a temporary file using the DET_{GB}.
Step 2: Read the query log files; frequently accessed queries are evaluated and ranked the frequently appeared attributes using AAT technique.
Step 3: ABS_DET_{GB} technique is used to filter the de-normalized attribute using the association aware technique.
Step 4: The filtered attributes and tables are created as fields and collections respectively in the NoSQL database based on its ranking.
Step 5: The ABS_DET_{GB} passed the Key to the Map-Reduce function; the keys are attributes of the data (value) to be migrated.
Step 6: The map-reduce function sends request to the source database based on the keys received from the ABS_DET_{GB} technique. Later, the data (values) are migrated parallel to the NoSQL database.
Most of the existing framework has used the Database Layers for migrating the data from the relational database to non-relational database. The novel feature of adding the denormalization technique along with the associate-aware technique makes the big data migration more effective. This feature resulted with more accuracy in time management and increased the performance of the post-migration process. The techniques used in the proposed framework are described in succeeding section.

IV. LARA TECHNIQUE

As it is presented and discussed in the preceding section the LARA framework is built by using the hybridization of the proposed graph based denormalization method along with the proposed associate rule mining technique. The following sections describe the procedures and algorithms used in the LARA framework.

4.1 Algorithm for LARA Technique

The overall process of the LARA technique is given below in the algorithm 1. Initially the source database (MySQL) is started. The inputs are collected from the source database using denormalization and associate-aware techniques to create collection in the target database (MongoDB). Python API is used to collect, migrate and create instances and collections in the databases.

Algorithm 1: LARA Algorithm

```
Input:  Database on MySQLDB must be started
Output:    Collection on MongoDB must be createdand Data Migrated
1:    ConnectToMySQLDB ()
```

```
2:  ConnectToMongoDB ()
3:  ForExist_MySQLDB:
4:  For exist MySQL_TABLE:
5:  ABS_DET_GB():          // call Association based denormalization
6:  Create new_COLLECTION_MongoDB ():
7:  End for
8:   End for
9:  MapReduce_Migrate()    // call mapreduce function for migrating the data
```

In LARA algorithm, the function ConnectToMySQLDB() establish a valid connection with the MySQL database. The target database MongoDB is connected using the ConnectToMongoDB() function. The source and target database are connected through the Python API. It is essential to check the entities and attributes of the source database and also it is vital to create the relevant collections and fields in the target database to process the data migration effectively and efficiently. Exist_MySQLDB() is the function to check the databases of the MySQL database. If the database exists, then it will check the entities in the MySQL database using the Exist_MySQLTABLE() function.

The proposed novelty of this work is to reduce the downtime of the database during the data migration by denormalizing the database using Graph based method along with associationranked attributes. ABS_DET$_{GB}$ () is the function used to filter the entities for the above statement. This function is discussed in the following sections. After performing the ABS_DET$_{GB}$(), the fields and collections are created to the relevant entities and attributes in the target MongoDB database using the new_COLLECTION_MongoDB () function. At last the MapReduce_Migrate () function migrate the big data from the source MySQL database to the target MongoDB database using the map reduce algorithm which is discussed in the following section.

.2 Graph Based Denormalization Technique (DET$_{GB}$)

The process of denormalization is to reduce the complexity of the usage of JOIN operations and increase the query-response performance. In the proposed LARA framework, the RDBMS is denormalized using the following algorithm 2. The graph-based algorithm is used to denormalize the normalized database. This could find the constraints efficiently based on the attributes and relationships among the attributes in different tables. The relationship between the primary key and foreign key are established using the graph and eliminated the complexity of the constraints and keeps the remaining entities with minimal constraints. The aim of this DET$_{GB}$ is to join the entity in to a single collection. As mentioned in the LARA technique, step 2 is initiated by the AAT technique. The stored information in the DET$_{GB}$is then passed to the AAT technique.

Algorithm 2: Graph Based Denormalization (DET$_{GB}$)

```
Input: SQL schema
Output: NoSQL schema
1:      Create Relational Schema Graph G
2:      Make G as acyclic graph
3:      Convert G into a set of $ST_i$
4:      for (each schema tree T ∈ ST) {
5:          create a collection for the root of T
6:              for (each non-root vertex v of T) {
7:      set v into the parent node vp of v
8:                  remove the foreign key in vp that refers to v
9:                      for (each query q ∈ Q) {
10:                         build a transaction-query graph G (V, K) for q
11:                         add the columns of the vertex d to the entity e
12:                     }
13:                 }
14:         }
```

4.3 Association Aware Technique (AAT)

The association aware technique has been proposed to decrease the downtime of the database access during the big data migration process. The brief summary of the functions of AAT technique is given below:

- Ranking the query based on its frequency of attribute accesses
- Filtered and ranked the attribute based on association rule
- Prioritize the attribute based on its ranking
- Convert the entity and attribute as collection and field respectively

The overall process of AAT is explained in the algorithm 3. The step 6 'association rule' is applied on the results of DET$_{GB}$.The step 6 in the algorithm 3 is used for filter and rank the attribute with entity, which is evolved as ABS_DET$_{GB}$ in this AAT and explained in the succeeding section.

Algorithm 3: Associate Aware Technique (AAT)

Notations used in the algorithms are described as follows:
q $\rightarrow$query
E $\rightarrow$ entity
A $\rightarrow$ Attribute

```
Input: Sorted query from the query log files
Output: Frequently Accessed Entities (FAE)
1:      Start
```

```
2:      Read Sort($q_j$)
                // i = 1,2,3…n
3:      Tokenize entities $E_i$ and attributes $A_j$
                //i =1,2,3…n and j = 1,2,3…m
4:      Find the frequent of E
5:      Find the frequent of A
6:      Apply ABS_DET$_{GB}$ for E and A
7:      Apply the Associated Attribute-Entity Ranker
8:      Stop
```

4.4 Association based Denormalization Technique (ABS_DET$_{GB}$)

Algorithm 4: Association Rule Generator

Notations used in the algorithms are described as follows:

Q → Query
C → Confidence
S → Support
Max → Maximum
Min → Minimum
T → Transaction
e→ entity
a→ attribute

```
1.  Start
2.  Function ARG (Q, T, $S_{min}$, $C_{min}$, $e_{max}$, $a_{max}$)
3.  $q \in Q$, where q = {$q_1$, $q_2$, $q_3$…...$q_m$)
4.  Let e and a = non-empty set
5.  $C_{e|a} := S_{i\in I}\{i\}$;
6.  While $F_{e|a} \neq \emptyset$ and e and a are maximum
7.  Check the $C_i$ of e and a where i = {1,2, 3…. n}
8.  MAX($C_i$) = MAX($C_{i+1}$)        /* Check up to n */
9.  Find the $S_{max}$
10. If S and C are below minimum, remove it from the T
11. Generate R using tf-idf
        /* explained in AER, Algorithm 5 */
12. Find most associated entity-attribute
13. Stop
```

4.5 Map Reduce Technique

The document-oriented database is practically undistinguishable to the RDBMS database, only storing data using a document-oriented model unlike RDBMS that storing data using a row-oriented model [10]. The database used for the document type is MongoDB, while the structure of the row database is as follows:

- Table: Similar the table in RDBMS.
- Row: A Primary Key.
- Column: Field in RDBMS.
- Value: Where information is stored.

The target database used for the document type is MongoDB, the document database has Embedded Documents and Reference Documents. Reference Documents such as Foreign Keys in an RDBMS, while Embedded Documents are like documents in a document. The structure of the MongoDB document is as follows:

- Collection: Same as the table in RDBMS.
- _id: A Primary Key.
- Key: Description of stored data.
- Value: Where information is stored.

V. RESULTS AND DISCUSSIONS

In this proposed framework, table 1 represents metrics used to compare and analyze the proposed work with the existing works.

Table 1. Evaluation Metrics

S. No.	Metric	Description
1.	Total Migration time (TMT)	Time period from start to complete of the live migration.
2.	Total Amount of Migrated Data (TMD)	Total amount of data migrated from source DB to the target DB.
3.	Downtime (DT)	Time interval during which the source DB is stopped and unavailable to the end user.
4.	On-premise server CPU utilization (CPU)	Total CPU resources consumed by the LARA framework during live migration on the on-premise server.
5.	On-premise server memory utilization (MEM)	Total memory resources consumed by the LARA framework during live migration on the on-premise server.

The required resources of the big data migration are evaluated by the TMT, CPU and MEM. The quality of service (QoS) is evaluated by the DT of the proposed framework whereas in possible cases, DT may affect Service Level Agreement of the users.

The proposed technique is evaluated with the e-learning portal (www.examey.com) database with more than 1 million rows. This web portal is a Beta version which allows the public to practice themselves for competitive exams. The datasets used for live big data migration are described in the table 2. These attributes are interlinked to each other and accessed many times as mentioned in the values. Based on this frequency, ranking the source database is denormalized using the association rule mining. The frequently accessed database attributes are denormalized to increase the post-migration process. In the previous contribution MR-EAAT technique, all the attributes are ranked based on the ranking and data are migrated to the target database.

Table 2. Dataset Description

S. No	Database	No. of Tables	No. of Columns
1.	Before Denormalization	84	549
2.	After Denormalization	47	416

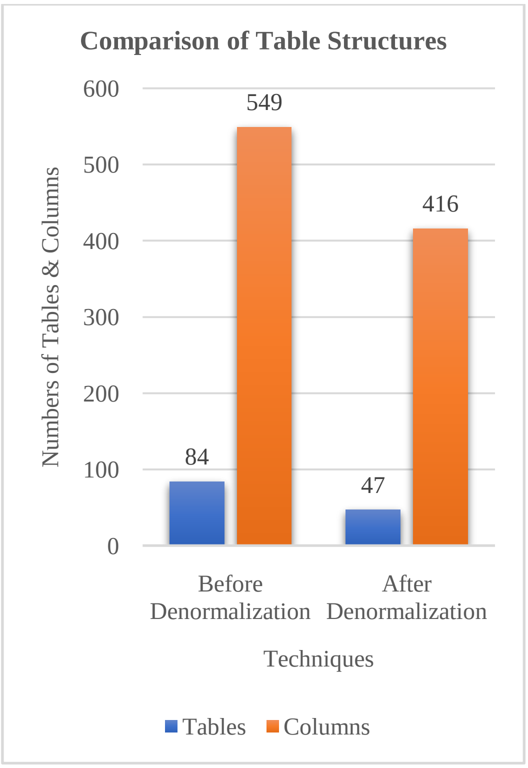

Figure 3. Comparison of Table Structures

The comparative results of database size after migrating the big data from the RDBMS to the NoSQL database is presented in the table 5. Generally, the NoSQL database consumes more space when compared to the RDBMS database. The actual size of the database is 355 MB. The MR_AAT technique which is discussed in the previous paper occupied 651.5 MB of the disk space. The proposed MR_ABS_DET$_{GB}$ technique occupied 991.9 MB of the disk space, i.e. it is 2.78 times greater than the source database. This is due to the denormalization technique. It generates more redundant data and reduce collections and fields in the target database.

Table 3. Table Structure before Denormalization

Table_Name	S. No	Attribute Name
Admin	1.	Admin_ID
	2.	User_Id
	3.	Admin_Name
	4.	Admin_Password
Faculty	1.	Faculty_Id
	2.	User_Id
	3.	Faculty_Name
	4.	Faculty_Password
User	1.	User_Id
	2.	User_First_Name
	3.	User_Middle_Name
	4.	User_Last_Name
	5.	User_Father_Name
	6.	User_Mother_Name
	7.	User_Name
	8.	User_Password
	9.	User_DOB
	10.	User_Sex
	11.	User_Address
	12.	User_Contact_Num
	13.	User_Email
	14.	User_Aadhar
	15.	User_Qualification

Table 4. Table Structure after Denormalization

S. No	Attribute Name
1.	User_Id
2.	User_First_Name
3.	User_Middle_Name
4.	User_Last_Name
5.	User_Father_Name
6.	User_Mother_Name
7.	User_Name
8.	User_Password
9.	User_DOB
10.	User_Sex
11.	User_Address
12.	User_Contact_Num
13.	User_Email
14.	User_Aadhar
15.	User_Qualification
16.	Roles

Table 5. Database Size Comparison after Data Migration

S. No	Techniques	Database Size (MB)
1.	MR_ART	620
2.	MR_ABS_DET$_{GB}$	991.9

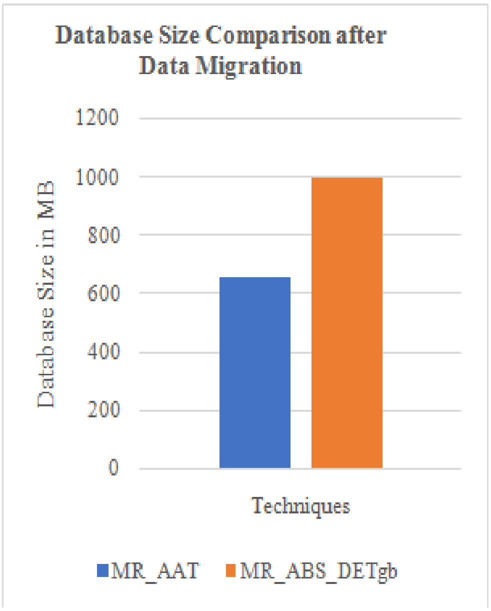

Figure 4. Comparison of Database Size after Data Migration

Table 6. Comparison of Database Downtime

S. No	Framework	Database Downtime (Time in Secs)
1.	MR_ART	140
2.	MR_ABS_DET$_{GB}$	129

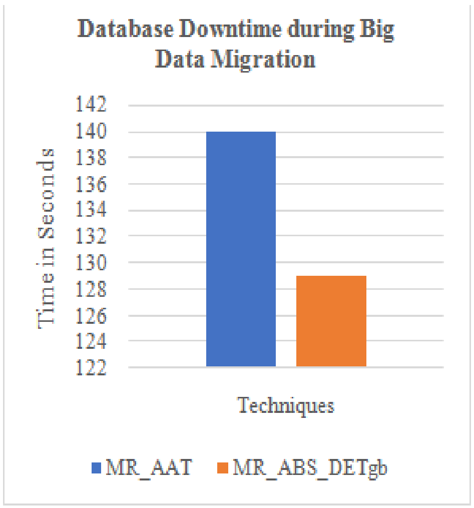

Figure 5. Comparison of Database Downtime during Big Data Migration

Table 7. Comparison of Select Statement

S. No	Number of Records	Framework (time in seconds)	
		NoSQLayer	LARA
1.	20000	14.29	11.17
2.	40000	25.11	21.27
3.	60000	37.51	30.83
4.	80000	46.89	39.64
5.	100000	52.59	42.56

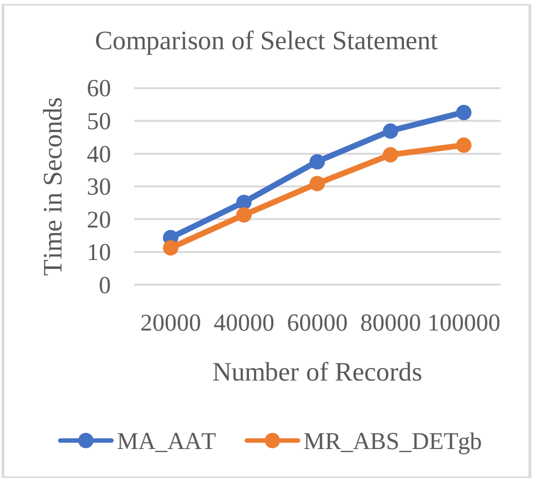

Figure 6. Comparison of Select Statement

Table 8. Comparison of Insert Statement

S. No	Number of Records	Framework (time in seconds)	
		NoSQLayer	LARA
1.	20000	140	129
2.	40000	149	133
3.	60000	261	245
4.	80000	472	457
5.	100000	784	769

Figure 7. Comparison of Insert Statement

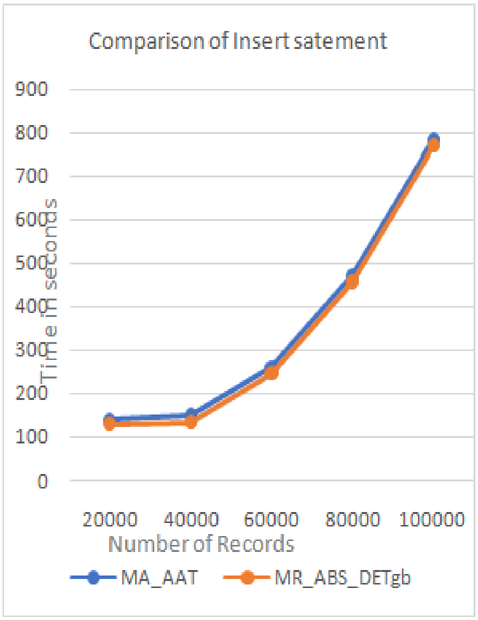

Table 9. Comparison of Delete Statement

S. No	Number of Records	Framework (time in seconds)	
		NoSQLayer	LARA
1.	20000	6.11	5.11
2.	40000	7.9	5.87
3.	60000	8.97	6.45
4.	80000	9.8	7.99
5.	100000	10.9	8.87

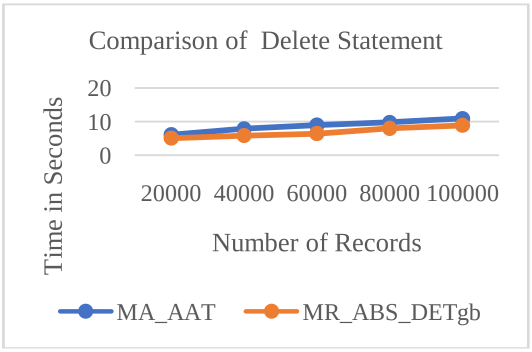

Figure 8. Comparison of Delete Statement

Table 10. Comparison of Server CPU Utilization

S. No	Size of the Data (MB)	CPU Usage (%)	
		NoSQLayer	LARA
1.	71	1	1
2.	142	1	2
3.	213	2	4
4.	284	3	6
5.	355	6	8

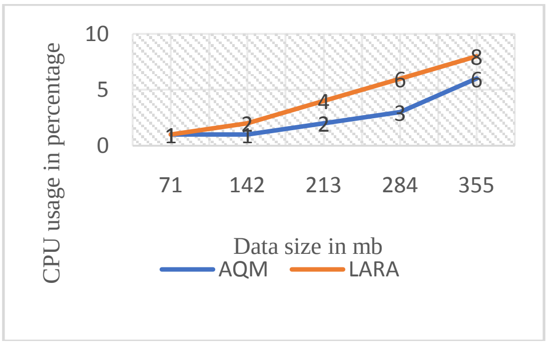

Figure 9. CPU Utilization during Data Migration

Table 11. Comparison of Server Memory Utilization

S. No	Size of the Data (MB)	Memory Usage (%)		
		NoSQLayer	LARA	
1.	71	27	35	
2.	142	35	41	
3.		213	41	47
4.		284	47	54
5.		355	52	60

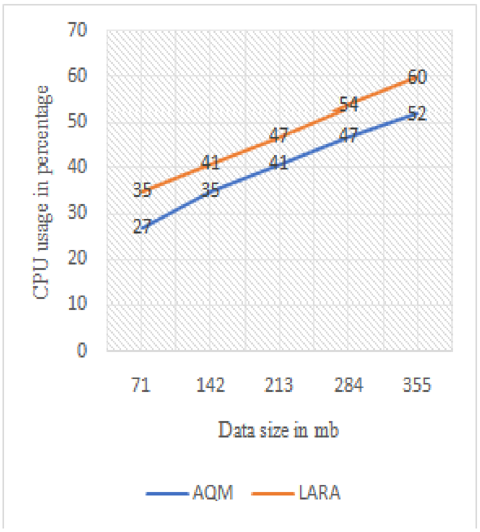

Figure10. Memory Utilization during Data Migration

VI. CONCLUSION

This paper has described a novel framework named LARA which is built to migrate big data in the on-premise (standalone) operating environment.The LARA framework is framed to transfer the big data from the relational database to the non-relational database system. The target database is immature to perform the join operations which is affected the post-migration process. The proposed ABS_DET$_{GB}$ algorithm in the LARA framework outperformed well and produced better results when compared with the existing proposed techniques. The CPU utilization and memory utilization are also evaluated and compared with the proposed techniques. The LARA framework still requires to be tested in other cases or using different data and other operating environments.

REFERENCES

[1]. H. M. L. Dharmasiri and M. D. J. S. Goonetillake, “A federatedapproach on heterogeneousNoSQL data stores,” Int. Conf. Adv.ICTEmerg. Reg. ICTer 2013 - Conf. Proc., no. December,pp.234–239, 2013.
[2]. V. N. Gudivada, D. Rao, and V. V. Raghavan, “NoSQL Systemsfor Big Data Management,”2014 IEEE World Congr. Serv., pp.190–197, 2014.
[3]. M. Scavuzzo, E. Di Nitto, and S. Ceri, “Interoperable datamigration between NoSQLcolumnar databases,” Proc. – IEEEInt. Enterp. Distrib. Object Comput. Work. EDOCW, pp.154–162, 2014.
[4]. A. Bansel, H. Gonzalez-Velez, and A. E. Chis, “Cloud-BasedNoSQL Data Migration,” Proc.- 24th Euromicro Int. Conf.Parallel, Distrib. Network-Based Process. PDP 2016, pp. 224–231, 2016.
[5]. M. N. Shirazi, H. C. Kuan, and H. Dolatabadi, “Design patternsto enable data

portabilitybetweenclouds' databases," Proc. -12th Int. Conf. Comput. Sci. It's Appl. ICCSA 2012, pp. 117–120,2012.

[6]. D. Petcu, H. Gonz´alez-V´elez, B. Nicolae, J. M. Garc´ıa-G´omez,E. Fuster-Garcia, and C. Sheridan, "Next generation HPC clouds: A viewfor large-scale scientific and data-intensive applications," in Euro-Par2014, Revised Selected Papers, Part II, ser. Lecture Notes inComputerScience, vol. 8806. Porto: Springer, pp. 26–37, Aug. 2014.

[7]. B. Thalheim and Q. Wang, "Data migration: A theoretical perspective,"Dataand Knowledge Engineering, vol. 87, pp. 260–278, 2013.

[8]. R. Sellami, S. Bhiri, and B. Defude, "ODBAPI: A unified REST APIfor relational and NoSQL data stores," in Big Data Congress 2014.Anchorage: IEEE, pp. 653–660, Jun. 2014.

[9]. R. Cattell, "Scalable SQL and NoSQL data stores," SIGMOD Rec.,vol. 39, no. 4, pp. 12–27, May 2011.

[10]. Z. Chen, S. Yang, H. Zhao, and H. Yin, "An objective function fordividing class family in NoSQL database," in CSSS 2012. Nanjing:IEEE, pp. 2091–2094, Aug. 2012.

[11]. L. Rocha, F. Vale, E. Cirilo, D. Barbosa, and F. Mourao, "A frameworkfor migrating relational datasets to NoSQL," Procedia ComputerScience, vol. 51, pp. 2593–2602, 2015.

[12]. K. North, "The NoSQL alternative,"

[13]. C. Tauro, N. Ganesan, A. Easo, and S. Mathew, "Convergent replicateddata structures that tolerate eventual consistency in NoSQL databases,"in ICACC 2013. Cochin: IEEE, pp. 70–75, Aug. 2013.

[14]. V. Abramova and J. Bernardino, "NoSQL Databases: MongoDBvsCassandra," in C3S2E '13. Porto: ACM, pp. 14–22, Jul. 2013.

[15]. W. McKnight, "Graph databases: When relationships are the data," inInformation Management: Strategies for gaining a competitive advantagewith data, W. McKnight, Ed. Boston: Morgan Kaufmann, ch. 12, pp. 120–131, 2014.

[16]. M. Qi, "Digital forensics and NoSQL databases," in FSKD 2014.Xiamen: IEEE, pp. 734–739, Aug. 2014.

V. Rathika" A Novel Big Data Migrationframework With Reduced Database Downtime"International Journal of Engineering Research and Applications (IJERA), Vol. 09, No.08, 2019, pp. 43-52