Journal of Computer Science

Original Research Paper

E-GENMR: Enhanced Generalized Query Processing using Double Hashing Technique through MapReduce in Cloud Database Management System

[1]Shweta Malhotra, [1]Mohammad Najmud Doja, [1]Bashir Alam and [2]Mansaf Alam

[1]*Department of Computer Engineering, Jamia Millia Islamia, New Delhi-110025, India*
[2]*Department of Computer Science, Jamia Millia Islamia, New Delhi-110025, India*

Article history
Received: 24-04-2017
Revised: 15-06-2017
Accepted: 12-07-2017

Corresponding Author:
Shweta Malhotra
Department of Computer Engineering, Jamia Millia Islamia, New Delhi-110025, India
Email: shweta.mongia@yahoo.com

Abstract: Big Data, Cloud computing and Data Science is the booming future of IT industries. The common thing among all the new techniques is that they deal with not just Data but Big Data. Users store various kinds of data on cloud repositories. Cloud Database Management System deals with these large sets of data. Cloud Database service provider deals with many obstacles while providing various service. Amongst all the challenges processing of large amount of data, interoperability and security are the major concerns that are explained in this study. Enhanced Generalized Query Processing through MapReduce (E-GENMR) is a prototype model that provides solution for these problems. Firstly, traditional approaches are not suitable for processing such gigantic amount of data as they are not able to handle such amount of data. Various solutions have been developed such as Hadoop, MapReduce Programming codes, HIVE, PIG etc. but these technologies don't provide solution for these problems at the same time and moreover users are not compatible with these latest technologies like MapReduce codes. E-GENMR provides interoperability as it takes queries written in various RDBMS forms like SQL Server, ORACLE, DB2, MYSQL and convert into MapReduce codes as they are considered to be the efficient way for processing large data. Secondly, Client's data is stored in encrypted form and processing is done on this data hence it ensures the security aspect. Indexing plays a very important role in processing queries, in E-GENMR indexing is implemented using closed double hashing technique. We compared various query processing time of E-GENMR for encrypted data and unencrypted data. A comparison of various queries has been done to evaluate the performance of E-GENMR with latest techniques like Hadoopdb, SQLMR, HIVE and PIG and it has been concluded that E-GENMR shows better performance.

Keywords: MapReduce, Cloud Database Management System CDBMS, Generalized Query Processing, Interoperability, Conceptual Middleware Layer, MapReduce Compiler, GENMR, Encrypted Data, Security

Introduction

One of the influential service that a cloud service provider provides is Cloud Database. Many Cloud provider Companies such as Amazon, Yahoo, EMC2, Microsoft, Google, Rackspace etc. provide database services in SQL and NOSQL form. Users on cloud can access Cloud Database service by two ways either by running their databases on virtual machine provided by cloud provider or they can use directly the database services provided by the cloud service provider. MySQL, PostgreSQL, Microsoft SQL Server, NuoDB are some of the SQL services provided by the Cloud service provider. Cassandra, MongoDB, CouchDB are some of the examples of NOSQL types of Database services (Bloor, 2011).

CDBMS is attractive for various reasons as organizations are not bothered about the hardware maintenance, software cost or any administrative cost, they only focus on the efficiency of their business.

In 2016, the latest Beckman Report on database Research (Abadi *et al*., 2016) discussed various research challenges in this field. It has been concluded that among various challenges of Cloud Databases processing, interoperability and security of data present at cloud repositories are the major one's and are the concern of this paper.

Some interesting points of the report are "*Many big data applications will be deployed in the cloud, both public and private, on a massive scale. This requires new techniques to offer predictable performance and flexible interoperation "and "A diverse and data-driven world requires diverse programming abstractions to operate on very large datasets" (Abadi et al., 2016).*

Cloud Database service providers deal with many obstacles while providing the service. Firstly, Processing of the data present on the cloud has become a biggest issue now a days. Such huge amount of data is being generated from various sources like Sensors, social networking sites etc (Manyikaetal, 2011). Traditional database management systems are not able to process such hefty size of data. New technologies such as MapReduce, Hive, PIG, Hadoop etc. are emerging as a solution for processing this data. But till date, users are very much comfortable with traditional DBMS and not with the MapReduce codes.

MapReduce codes available in the market are attractive as they provide benefits like being present in simple Key-value form hence they are easy to use. They are a Cost effective solution for processing large size of data as they provide parallel processing. MapReduce codes provide flexibility as it is not based on any schema, data can either be in structured or unstructured form. MapReduce codes provide scalability as well (Dean and Ghemawat, 2008).

One of the main characteristic of Cloud is that it is based on multitenant environment which means many clients share the same datacenter provided by the cloud provider (Pippal *et al*., 2001). Multiple Clients store their data with the Cloud Service provider so security is the biggest challenge for the Cloud Service Provider. Earlier models deal with the problems of either security or processing but E-GENMR provides solution for both these problems as shown in Fig. 1. E-GENMR provides security features at each step. Client's data is being stored in encrypted form and processing is being done on this encrypted data.

Indexing plays a very important role in processing queries, A survey of various indexing techniques for Big Data in Cloud has been explained in (Adamu *et al*., 2015; Gani *et al*., 2016). In E-GENMR indexing is implemented using closed double hashing technique. Previous techniques like Map reduce uses Inverted indexing technique Table 1 summarized the indexing techniques used in the latest techniques used for processing Big Data.

Inverted indexing technique used in Map Reduce programming paradigm takes $o(|q|*D*|D|)$ times where $|q|$ is the length of query and $|D|$ is the length of document whereas B+ tree indexing technique is used in SQLMR which takes $\log(n)$ complexity for searching any data. In Bit map indexing for low cardinality attributes space complexity is low whereas for higher cardinality attributes the space complexity is very high. Overall advantage with Double hashing is that searching complexity is in the order of 1 i.e., $o(1)$. It takes only time for the computation of hash function.

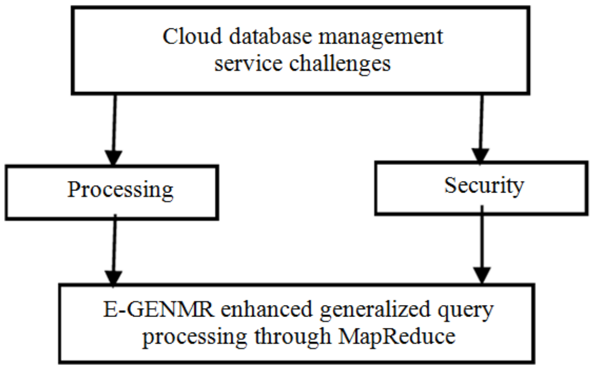
Cloud database management service challenges
Processing
Security
E-GENMR enhanced generalized query processing through MapReduce

Fig. 1. Obstacles of cloud database management system

Table 1. Indexing techniques used in the latest techniques for processing Bigdata on Cloud

System	Data model	Indexing
Map reduce	Key-value pair	Batch Indexing (inverted index)
Pig Latin	Atom, Tuple, Bag, Map	Local, Mapreduce, Batch, Interactive, inverted index
HIVE	tuple	Bit Map Indexing
HadoopDB	tuple	Clustered Multidimensional
SQLMR	tuple	Simple and B+ Partitioning method
E-GENMR	tuple	Double hashing technique

A prototype model E-GENMR has been implemented to give solutions for processing and security of such large sized data for cloud database management system. The key contribution of our work is defined as follows:

- CDBMS Layer wise responsibility and Architecture: Layer wise responsibility related to Cloud Database Management System and Architecture of prototype model has been defined
- Interoperability (Generalization): Users can write their queries in syntax defined by either RDBMS system SQL server, MYSQL, DB2 or Oracle hence this prototype provides interoperability. With the help of E-GENMR compiler queries, data is converted into MapReduce Key-Value form
- Capability: Our approach can takes both simple and complex queries with more than one filter
- Security: For security reasons, Clients data is being stored in encrypted form and queries are being performed on encrypted data
- Efficient Approach: With the help of efficient Double hash indexing technique it provides efficient way of processing large amount of data as compared to the other latest techniques such as HadoopDB, SQLMR, HIVE and PIG

Rest of the paper is organized as follows. Section 2 describes the work that has been done so far related to the field of Cloud Database Management System, Big Data and security issues related to cloud. In section 3, we briefly define our proposed model along with the algorithms which is used for the implementation of E-GENMR. In section 4, Results and analysis have been described. We analyzed E-GENMR with latest techniques. Lastly, we conclude our work with future possibilities in section 5.

Related Work

The literature was reviewed to bring out the salient features and techniques being used in this field. The literature review has been grouped into following categories: Big data and the latest techniques for processing Big data on Cloud, Generalized query interface, Cloud Security and Indexing techniques used for the Big data on cloud.

Big data now a days characterized by seven characteristics named as volume, velocity, variety, veracity, variability, value and complexity (6 Vs and Complexity) are described in (Manyikaetal, 2011). Simple MapReduce (Dean and Ghemawat, 2008) codes in key-value pair are considered to be a suitable solution for large amount of parallel data processing. Dean and Ghemawat (2008), introduced MapReduce Programming paradigm based on Parallel and distributed computing in which Inverted index scheme is used (McCreadie *et al*., 2012).

Another technique used for processing which is Hive defined by acts as data warehouse system built inside the hadoop file system. It provides user with a platform where they can easily use queries similar to SQL but is named differently called HiveQL, which are compiled into mapreduce jobs that are executed using Hadoop. In Hive system Bit Map indexing i.e., a Simple indexes with single attribute is used and creation of indexes is linearly proportional (Liu *et al*., 2013; Fuad *et al*., 2014).

Abouzeid *et al*. (2009) HadoopDB is a data management system that combines the capability of RDBMS and map reduce programming paradigm. It inherits the scalability feature from Hadoop and combines the basic features of DBMS. It achieves better results compared with parallel databases Vertica, DB-X etc. Indexes in HadoopDB are maintained internally by Local DBMS. A lot of time is consumed in pre-partitioning phase.

Hsieh *et al*. (2011) Implemented one system model named “SQLMR”, which is a hybrid approach to fill the gap between SQL-based and MapReduce data processing. With effective part partitioning and B tree indexing, low overhead file construction, optimized rack awareness algorithm, query result cache mechanism the system produced best results as compare to HadoopDB. YSmart (Lee *et al*., 2011a) which is another system similar to SQLMR based on correlation aware SQL-to-MapReduce translator.

An enhancement MapReduce codes is being provided with the help of pipelining concept i.e., Whenever Mapper function produces its results in the intermediate form it goes to Reducer function for generating output (Lee *et al*., 2011b; Dahiphale *et al*., 2014; Condie *et al*., 2009) to provide further parallel processing of data. Jayalath *et al*. (2013) described the efficient way to process Bigdata across geographical distributed data centers.

Li-Yung *et al*. (2011) explained one optimization algorithm for cross Rack Optimization for Reducer program. Here, generalized model takes Mapper function into account as well. A detail related to theoretical proposed model is given in (Malhotra *et al*., 2015) which explains the interoperability in the model that takes queries in SQL, MYSQL, DB2, Oracle form and converts into MapReduce form. Big data analysis (Ramamoorthy and Rajalakshmi, 2013) on Cloud has become an issue; the author provides a solution of MapReduce algorithm and Bigdata analytic techniques. In paper (Li *et al*., 2011) author explained the enhancements that are happening in the Cloud Computing world. MySQL provides a way to process and manipulate data but it is not applicable for large amount of data sets. In (Mongia and Kataria, 2015), Authors discussed about the Layer wise Security issues related to Cloud Database Management System and also discussed about the so far implemented and proposed solutions for each security issue. In this study, implemented model provide solution for data Security by encrypting the data with the help of complex algorithm.

Prototype Model

The problem with the today's world is that users are not comfortable with MapReduce kind of codes to process large size of data present at the cloud repositories. Secondly, Cloud is based on Multitenant environment in which multiple clients uses the services provided by the Cloud. Multiple Clients store their databases with the Cloud Service provider so security is the biggest challenge for the Cloud Service Provider. Earlier models deal with the problems of security and processing but here E-GENMR provides solution for both the problems along with solving the interoperability issue.

The prototype model provides interoperability as it takes up user queries in any of the syntax defined by RDBMS like SQL server, DB2, Oracle, MySQL hence it is called as Generalized and with the help of model's compiler module, queries get converted into MapReduce form. MapReduce is a splendid solution to process large amount of data as these codes process data in parallel. Client's data is stored in the encrypted form and queries runs on encrypted data hence the system ensures security aspects also.

A five layer architecture for Cloud Database Management System has been proposed in (Mongia *et al*., 2013; Alam and Shakil, 2013). In the below sections, a detailed working description in the form of algorithms related to each layer has been provided and briefly explained in Fig. 2. Figure 3 and 4 describes the architecture of Proposed Generalized Model: Data storage phase and data processing phase.

External Layer: User Interface

External Layer is the only layer which is closest to the user and provide interfacing. The main function of this Layer is to provide the transparency and to manage different types of users. User sends their queries in the syntax of SQL server, DB2, Oracle, MySQL. Existed data is pre-partitioned horizontally, indexed with the help of double hashing and stored in encrypted form into the number of Data nodes of the Racks to have parallel and distributed processing as explained by algorithm 1. For efficiently storing data double hashing technique is used as it takes only 0(1) time for searching any data due to the hash indexing. Cloud repository consist of Big Data Centers which consist of many Racks, where Data is stored in inter Racks and Intra Rack. Algorithm 1 also described the way data is stored in Inter-Rack or Intra-Rack to have Inter or Intra Rack Communication. Table 2 and 3 comprised of a symbols and assumptions used throughout the paper.

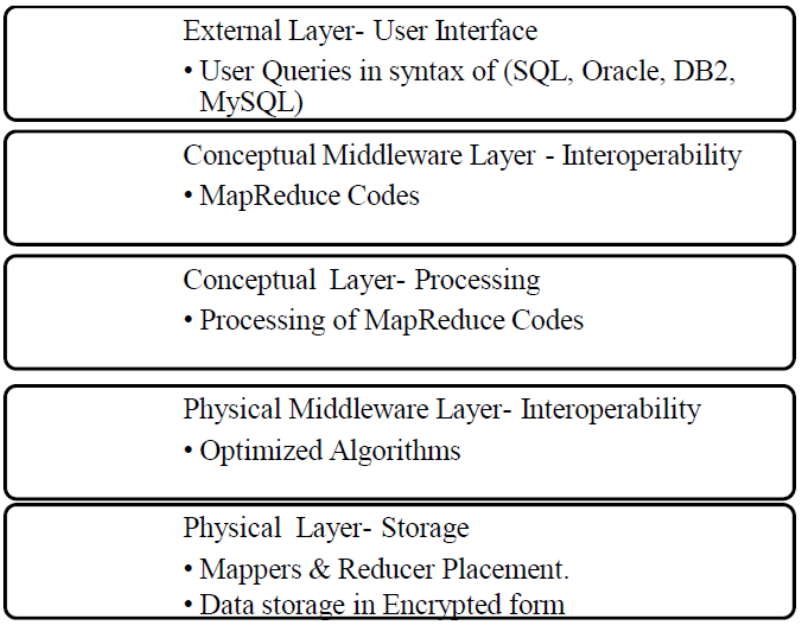
External Layer- User Interface
• User Queries in syntax of (SQL, Oracle, DB2, MySQL)
Conceptual Middleware Layer - Interoperability
• MapReduce Codes
Conceptual Layer- Processing
• Processing of MapReduce Codes
Physical Middleware Layer- Interoperability
• Optimized Algorithms
Physical Layer- Storage
• Mappers & Reducer Placement.
• Data storage in Encrypted form

Fig. 2. Layer wise responsibility

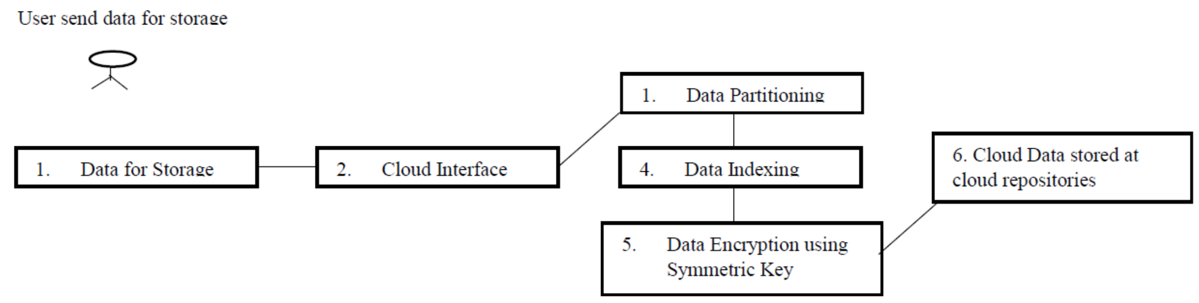
User send data for storage
1. Data for Storage
2. Cloud Interface
1. Data Partitioning
4. Data Indexing
5. Data Encryption using Symmetric Key
6. Cloud Data stored at cloud repositories

Fig. 3. Architecture of data storage phase

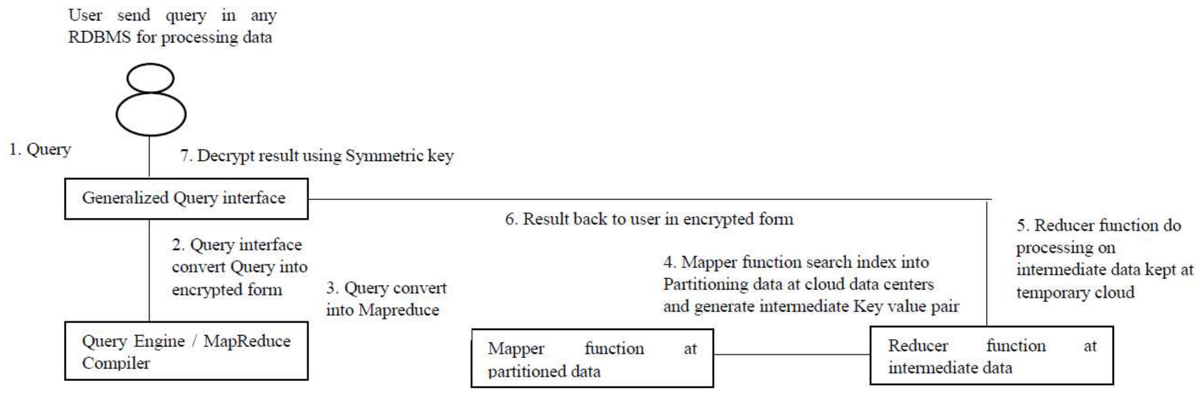
User send query in any RDBMS for processing data
1. Query
7. Decrypt result using Symmetric key
Generalized Query interface
6. Result back to user in encrypted form
5. Reducer function do processing on intermediate data kept at temporary cloud
2. Query interface convert Query into encrypted form
3. Query convert into Mapreduce
4. Mapper function search index into Partitioning data at cloud data centers and generate intermediate Key value pair
Query Engine / MapReduce Compiler
Mapper function at partitioned data
Reducer function at intermediate data

Fig. 4. Architecture of data processing phase

Table 2. Symbol used

Symbol used	Definition/explanation
Rack1,Rack2,Rack3…….. Rackn	n number of Racks.
d11, d12……… d1m	Each Rack consist of m no of Datanodes, example shows these Datanodes are of Rack1
Data1, Data2	Data present on the Datanodes
M	Mapper Function
R	Reducer Function

Table 3. Assumptions used

Sr. No.	Assumptions
1	There are n number of clients and client's data present on the Cloud Database.
2	Client data partitioned on the FCFS basis.
3	One Data row is stored at one Datanode of a Rack
4.	Datanode capacity is *q* …. *q* rows can be kept on that Datanode.

Algorithm 1. Pre-partitioning and storage of data in encrypted form

Input: Data is stored horizontally in rows. Rows are stored at Datanode's of the Racks.
Data is placed as per the Datanode capacity
Datanode capacity is q rows.
Output: Partition and store the data in encrypted form on the Intra racks i.e., Users data is placed at the Datanodes of same Racks
$d_{1\_rack1\_row1}$, $d_{1\_rack1\_row2}$, …………………. $d_{1\_rack1\_rowq}$, $d_{2\_rack1\_rowq+1}$, $d_{2\_rack1\_rowq+2}$, ………….. $d_{2\_rack1\_rowq+q}$, $d_{z/q\_rack1\_row(z-i-2)}$, $d_{z/q\_rack1\_row(z-i-1)}$, ………… d_{z/q_rack1_rowz},
or Inter Rack i.e., Users data is placed at the Datanodes of different Racks to have parallel processing.
$d_{1\_rack1\_row1}$, …$d_{i\_rack1\_rowq}$,..$d_{i+1\_rack2\_rowq+1}$, …$d_{i+i\_rack2\_rowq+q}$, … d_{z/q_rackn_rowz},

```
1. Procedure: Pre-Partitioning and hash indexing
2. For user's data
3. Case 1: Intra rack
4. If total Data size is z.
5. Total number of Datanodes required on that particular
   rack will be
    Total Datanodes = z/q………………………..(i)
      For indexing double indexing is used
      $h_i(data) = (h(data)+f(i)) \mod (datanode\_size)$
      where $f(i) = i+hash_2(x)$
      Store data in Encrypted form with the help of AES
      algorithm.
```

```
6. Until all the data is placed at the Datanodes of the
   Rack.
7. Case 2: Inter Rack.
8. Total number of Datanodes required on all the Racks
   will be same i.e., Total Datanodes = z/q
9. for i = 1 to n … for n number of Racks
10.   for j = 1 to m ….. for m datanodes
11.   Data is partitioned as to have total datanodes = z/q
        For indexing double indexing is used
        hi(data) = (h(data)+f(i)) mod (datanode_size)
        where f(i) = i+$hash_2$(x)
        Store data in Encrypted form with the help of AES
        algorithm.
12.   Until all the data is placed at the Datanodes of the
      Racks.
13.   End of For loop
14. End of For Loop.
```

In algorithm 1, intra rack communication is explained in lines (2-6) and inter rack communication is explained in lines (7-12). In Intra rack communication data is stored at the datanode's of same rack while in Inter rack communication data can be stored at the datanodes of any rack (line 9). Double hashing technique is used for indexing data on datanode's as indexing is used for efficient searching of data after indexing data is stored in encrypted form using symmetric key algorithm (AES) for security reasons. At external Layer when data is being partitioned other clients will not be able to predict the data as they can see only the encrypted form of data.

Conceptual Middleware Layer: Any Database to MapReduce Compiler

This layer provides interoperability which means it hides the availability of different databases to the users and operates irrespective of the underlying available databases. User's process their queries in the Databases languages in which they are comfortable. Users till date are comfortable with RDBMS tools but RDBMS is not a probable solution for processing large amount of Data. Users are not compatible with new technologies like MapReduce Programming Paradigm, Hive, Pig, HBase which can process large amount of data. This layer provides the facility to the users such that their queries are converted into NOSQL Map-Reduce key-value form. Compiler takes input queries from the user interface which is at the external layer. It converts these queries into MapReduce codes. Query takes pre-partitioned data from the text file stored at the DataNodes of the Racks. On the basis of queries again partitioning is done. The data obtained after this partitioning is called intermediate data. Table 4 has the detail of queries considered in this study and the corresponding Key-value pairs defined by the Model's compiler.

This prototype model can handle queries with more than one filter and takes up the complex queries like Order-by, Group-by, Join with more than one Data Table. At conceptual middleware and conceptual layer data is being processed. MapReduce key-value pairs are being generated by this process are in encrypted form so none of the other clients can see the processing of other clients.

Conceptual Layer: Data Processing

This layer deals with actual processing of data. At this layer actual processing of key value pair is being done. Reducer will be applied to the partitioned intermediate data. Table 4 comprised of the queries for data processing and with the help of conceptual Middleware Layer's Compiler these queries are converted into key value pair. Now, at conceptual Layer reducer program takes the key-value pair and give results accordingly as described by the algorithm 2.

Algorithm 2 has the detail of the reducer function which gives result back to the user.

Table 4. Database queries

Query	MapReduce- Key-Value Pair	
Select * from Table Name where Column name= value	Key = Column name	Value = all other fields name except key column name
Select Count Column name from Table Name	Key = Column name	Value = 1
Select Distinct Column name from Table Name	Key = Column name	Value = 1
Select Upper Column name from Table Name	Key = Column name	Value = 1
Select substring Column name from Table Name	Key = Column name	Value = 1
Select Count Column Name from Table Name where Column Name = value	Key = Column name	Value = 1
Select Distinct Column Name from Table Name + where Column Name = value	Key= Column name	Value = 1
Select Upper Column Name from Table Name + where Column Name = value	Key = Column name	Value = 1
Select substring Column Name from Table Name + where Column Name = value	Key = Column name	Value = 1
Select +(Count/ Distinct/ Upper/ substring)+Column Name from Table Name+ where Column Name = value + (and) + Column Name = Value	Key = Column name	Value = 1
Select + (Count/ Distinct/ Upper/substring)+Column Name from Table Name + where Column Name = value + (or) + Column Name = Value	Key = Column name	Value = 1
Select * from Table Name Orderby Column Name Asc/Desc	Key = Column name	Value = all other fields name except key column name
Groupby	Key = Column name	Value = 1
Join Query	Key = Column name	Value = 1
Subqueries with (NOT IN, NOT EXIST)	Key = Column name	Value = 1

Algorithm 2 – Generation of Key Value Pair: Any Database to MapReduce compiler

Queries are applied on the pre-partitioned encrypted data. Proposed model can take queries in the syntax of SQL Server, MYSQL, ORACLE and DB2.

Input: Queries in SQL Server, DB2, MYSQL, Oracle
Output: Key-Value Pair in NOSQL form and Processed result back to user.
For Queries in SQL Server, DB2, MYSQL, ORACLE, following algorithm explained the generation of the encrypted Key-Value Pair. Mapper program is used to generate key-value pair and Reducer program gives results back to the user on the basis of key-value pair.

```
1. Procedure: Any Database to MapReduce compiler
2. For each query, Mappers will generate the Key-value pair
3. If Query == "SELECT * FROM Table_Name WHERE Column_Name == Value"
     Key = Column Name
     Value == all the fields of Table except key Column name
     Result = Key + value
4. ElseIf Query == "SELECT COUNT Column_Name FROM Table_Name + WHERE Column_ name1==Value + AND/OR + Column_Name2==Value
     Key= Column_Name
     Value= 1
     Result= Sum (values)
5. ElseIf Query == "SELECT DISTINCT Column_Name FROM Table_Name + WHERE Column_ name1==Value + AND/ OR + Column_Name2==Value
     Key = Column_Name
     Value = 1
     Result = Sum (Distinct value of key Column Name)
6. ElseIf Query == "SELECT UPPER Column_Name FROM Table_Name + WHERE Column_ name1==Value + AND/ OR + Column_Name2==Value
     Key= Column_Name
     Value= 1
     Result= Upper (Column_Name)
7. ElseIf Query == "SELECT SUBSTRING Column_Name FROM Table_Name + WHERE Column_ name1== Value + AND/ OR + Column_Name2==Value
     Key= Column_Name
     Value = 1
     Result = Substring (Column_Name)
8. Elseif Query == "SELECT Column_Name == Value ORDERBY Asc/Desc"
       Key = column name
       Value = all the fields of Table except key Column name
       Result = Asc/Desc(Column_Name)
9. Elseif Query == Group by
       Key = Column name
       Value = 1
       Result = Column name
10. Elseif Query == Join
       Key = column name
       Value = all the fields of Table except key Column name
       Result = Key + Value
11. Elseif Query == Subquery with NOT IN and NOT EXIST
       Key = column name of subquery's table
       Value= all the fields of main select Query except key Column name
       Result = Key+Value
12. End of nested if-else
13. End of for loop for queries.
```

Physical Middleware Layer and Physical Layer: Mapper Reducer Placement and Storage Issues

Two important aspects related to storage like Inter Rack or Intra Rack communication shown in Fig. 5 and Mapper and Reducer function Placement problems with encrypted data are considered at these layers. Physical Middleware layer provides interoperability but main storage related issues are dealt on Physical Layer.

Initially, at the physical layer, data is being pre-partitioned and stored in encrypted form as per the Algorithm 1.

Equation 1 defined below, explains the Mappers and reducer function placed on the Racks. To place Mappers and reducer function, three possibilities can be considered:

$$f_i\left(m_i, r_i\right) = m_i d_i r_i \qquad (1)$$

- Mapper and reducer is placed at the same DataNode of the same Rack where Users Data is present
- Mapper is placed at the same DataNode where data is present but reducer is placed at another DataNode of the same rack
- Mapper is placed at the same DataNode of the same rack where data is present and reducer is placed at the different DataNode of the different rack

where, i = 1, 2, 3………………, n are the number of racks, m represents Mapper function, r represents Reducer Function and d represents the Datanode.

Cross rack optimization (Abouzeid *et al.*, 2009) considered only Reducer Placement Problems. In this study both Mapper and Reducer Placement Problems on the DataNodes of the Racks have been considered.

Earlier Reducer function problem on the Racks DataNodes were not considered.

Algorithm 3: Mapper Function Placement

```
Input: Data Size and Datanode capacity.
Output: Mappers are placed on the Datanodes for data Processing.
1. Procedure: Mapper Function Placement Problem
2. If Datanode capacity is q …. q Rows can be kept on
   that data node.
3. If total data size is z.
4. Then total number of datanodes required on that
   particular rack will be:
        Total Datanodes = z/q
5. No. of mappers = total Datanodes for that data
6. End if
```

Algorithm 4: Reducer Function Placement

```
Input: Mapper function gives output in the form of Key value pair i.e., intermediate data.
Output: Reducer Functions are placed at the intermediate data.
1. Procedure: Reducer Function Placement Problem
2. For each Query as per the Algorithm 2 MapReduce
   compiler will produce intermediate data in the form
   of Key-value pair.
3. As per the query aggregation function i.e., Reducer
   function is applied on the intermediate data.
4. End of for loop
```

Reducer function gives back result of the aggregation function to the user.

At physical layer data is stored in encrypted form so that none of the clients can see each other's data.

Results and Performance Analysis

In our research we did experiment analysis on the data upto 512 GB, additionally we did comparison of E-GENMR with two SQL based database system MYSQL, DB2 and four NOSQL MapReduce based systems including HadoopDB, SQLMR, PIG, HIVE. Since all these MapReduce systems only handle read only operations so in our experiments we compare performance for read only operations including range, Join and OrderBy queries.

Two traditional databases have been used i.e., MYSQL and DB2 for the purpose of showing that these databases do not provide scalability feature. HIVE and PIG are considered to be a suitable choice for testing such situations. HIVE is SQL-Like language in which users send their queries in SQL form and with the help of Hadoop framework their queries internally get converted into MapReduce code and users get result. Similarly in PIG, PIG is a scripting Language where users write their queries in the scripts and these queries gets converted into MapReduce form with the help of Hadoop framework internally and users gets their results back. HadoopDB and SQLMR are the hybrid systems equipped of MapReduce and DBMS technologies for systematic workloads. Here, the prototype model E-GENMR is implemented in C#, with the help of .NET 2012 framework.

The experiment contain two parts: First part is to show the scalability with respect to data size. In the second part a comparison of E-GENMR is done with respect to encrypted and unencrypted data. Table 5 shows the system requirements that is being considered in this study.

Performance Comparison with MYSQL, DB2, HadoopDB, SQLMR, HIVE and PIG on Different Data Sizes

This set of experiments compare the scalability with respect to increase in data size for two queries i.e., SELECT and JOIN with ORDERBY. The data size varies from 512 GB Table 6 and 7 shows the execution time of MYSQL, DB2, HadoopDB, PIG HIVE, SQLMR and E-GENMR with different data sizes for SELECT and JOIN with ORDERBY queries. Figure 5 and 6 gives the graphical representation of the performance comparison for SELECT Query and Fig. 7 gives the graphical representation of the performance comparison for JOIN with ORDERBY Query. The SQL SELECT Query and JOIN with ORDERBY is as follows:

Query 1: SELECT COUNT (Column Name) FROM Table 1 WHERE Column Name = 'Value1 ' OR 'Value2'

Query 2: SELECT Table 1.ColumnName1, Table 1.ColumnName 2…, Table 2.ColumnName1, Table 2.Column Name2…….. FROM Table 1 INNER JOIN Table 2 ON Table 1.ColumnName 1 = Table 2.ColumnName1 OrderBy Table 1.ColumnName1

Table 5. System Requirement

System	Minimum Hardware Requirement	Software Requirement
E-GENMR	RAM-512 MB, Harddisk-20GB for processing and storage space =280 GB	Windows 8, Microsoft Visual Studio 2012,
HIVE	RAM-4 GB, Harddisk-20GB for processing and storage space =280 GB	Pseudo-Hadoop cluster with HIVE-0.13.1 version
PIG	RAM-4 GB, Harddisk-20GB for processing and storage space =280 GB	Pseudo-Hadoop cluster with PIG-0.12.0 version
HadoopDB	RAM-4 GB, Harddisk-20GB for processing and storage space =280 GB	Hadoopdb-all- 0.1.1.0

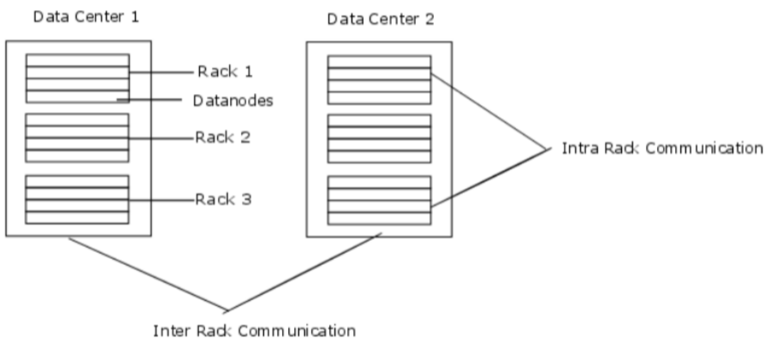

Fig. 5. Inter rack and intra rack communication

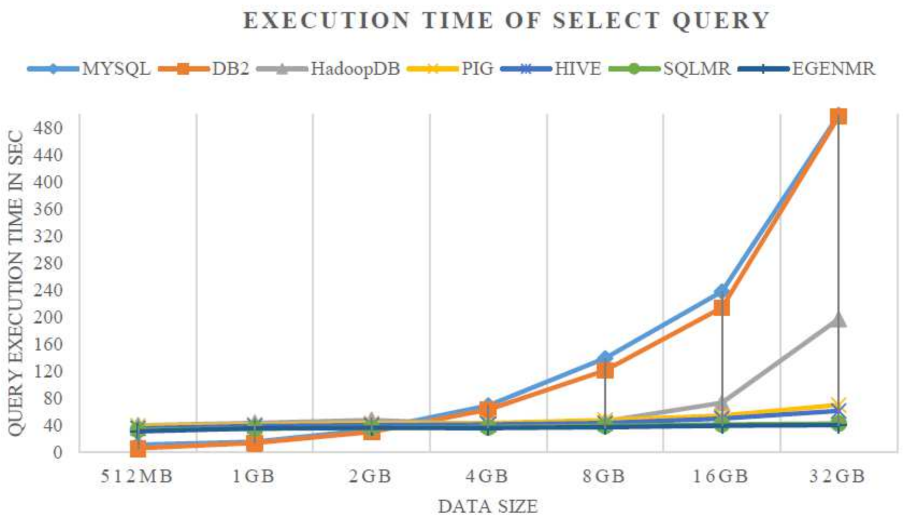

Fig. 6. Comparison of execution time for different SQL and NOSQL databases for SELECT query with small Data sizes

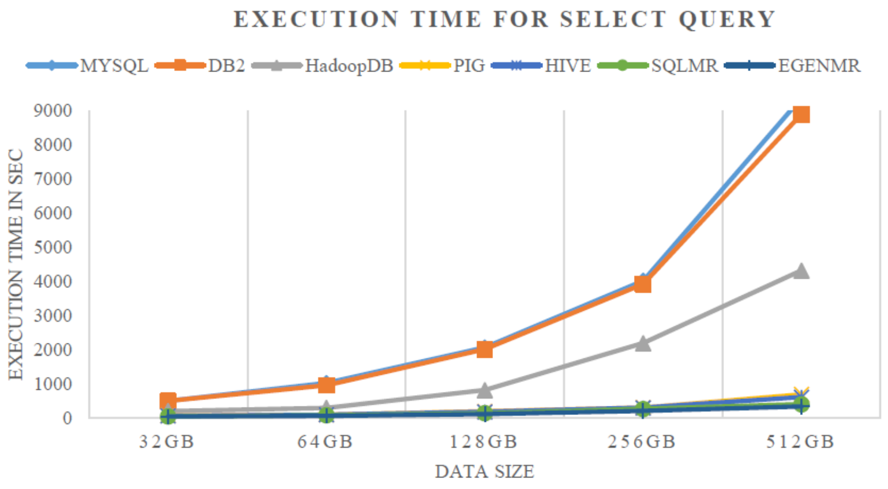

Fig. 7. Comparison of execution time for different SQL and NOSQL databases for SELECT query with large Data sizes

Figure 6 shows the graphical representation for the small data sizes while Fig. 7 shows the graphical representation for the large amount of data size. For small data size as shown in Fig. 6, DB2 performed better than MYSQL because it consumes primary memory for the processing. SQL based systems outperformed MapReduce Based systems for up to 2 GB of data size because SQL based system does not provide parallelism but in case of MapReduce based systems initial time is consumed for providing parallelism. When the data size increased further than 4GB Mapreduce based systems outperformed.

In Fig. 7, when the data size reaches to 512GB, MySQL, DB2, RDBMS performed very poorly but all the other systems performed better because of the parallel processing. Execution time of MapReduce based system HadoopDB increases intensely with increase of data set this is because of the higher input workload due to pre-partitioning done in HadoopDB System. SQLMR outperforms because of the effective part partitioning with B tree indexing, low overhead file construction, optimized rack awareness algorithm produced best results as compared to HadoopDB, HIVE and PIG. But E-GENMR consistently performed very well than all the other MapReduce based systems because of the various optimizations in terms of hash based pre-partitioning, double hash indexing and flexibility in terms of Mapper reducer placement described in section 3, as shown in Fig. 8, Join with Orderby Queries takes 2.70 more times than select queries as more time is required for processing of sort operation along with joining of two tables.

In general for SELECT queries E-GENMR model is 4.17 times faster than HadoopDB, 1.43 times faster than PIG, 1.19 times faster than HIVE and 1.11 times faster than SQLMR system. For JOIN with ORDERBY queries E-GENMR model is 1.28 times faster than SQLMR, 1.59 times faster than HIVE and 6.38 times faster than HadoopDB system.

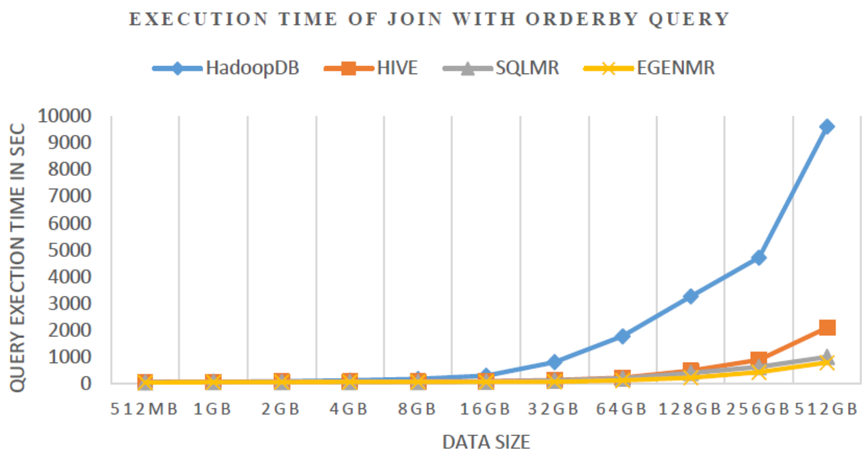

Fig. 8. Comparison of execution time for different SQL and NOSQL databases for JOIN with ORDERBY query with different Data sizes

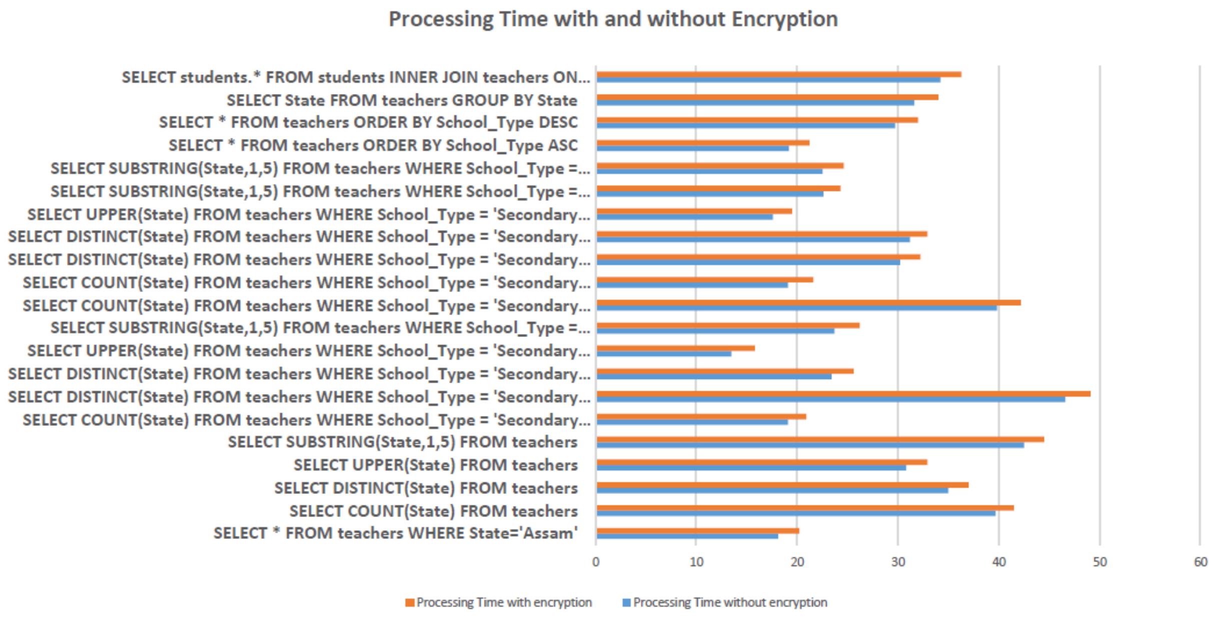

Fig. 9. Comparison of data processing time of E-GENMR for encrypted and unencrypted data for 21 queries

Table 6. Comparison of execution time for different SQL and NOSQL databases for SELECT query with different Data sizes

Data size	MYSQL	DB2	HadoopDB	PIG	HIVE	SQLMR	E-GENMR
256 MB	10.27	5.6	37.89	33.57	32.18	31.37	30.14
512 MB	11.34	5.87	40.14	39.17	35.63	32.92	31.22
1 GB	15.72	13.88	43.66	41.12	39.17	35.27	35.34
2 GB	33.58	30.12	47.91	43.22	40.48	36.48	36.34
4 GB	69.63	63.12	41.34	43.26	41.43	37.19	36.16
8 GB	139.34	121.22	46.29	47.85	43.29	39.21	37.17
16 GB	237.99	213.98	73.18	54.33	50.11	41.34	39.42
32 GB	499.17	497.12	197.34	70.2	61.37	42.79	40.26
64 GB	1021.21	953.88	294.32	97.22	93.39	76.32	60.42
128 GB	2059.23	2012.52	811.65	194.53	178.54	136.43	107.92
256 GB	4019.77	3919.76	2187.98	301.32	297.11	263.56	205.99
512 GB	9365.76	8897.43	4315.69	687.33	606.45	402.54	337.45

Table 7. Comparison of execution time for different SQL and NOSQL databases for JOIN by ORDERBY query with different Data sizes

Data size	HadoopDB	HIVE	SQLMR	E-GENMR
512 MB	80.99	74.12	69.55	54.79
1 GB	82.65	80.33	71.26	69.28
2 GB	98.79	81.45	75.38	72.37
4 GB	132.76	85.38	77.93	75.96
8 GB	189.88	89.65	81.28	77.37
16 GB	306.65	102.34	91.23	80.18
32 GB	819.75	151.21	134.19	83.89
64 GB	1785.43	239.65	207.63	132.64
128 GB	3269.82	492.14	412.87	234.77
256 GB	4718.37	901.23	635.29	441.73
512 GB	9613.74	2107.23	1014.87	801.56

Table 8. Queries considered for data processing of encrypted data and non- encrypted data

Queries	Time without encryption	Time with encryption
Select * from teachers where State='Assam'	39.7	41.5
Select count(State) from teachers	31.1	33.7
Select distinct(State) from teachers	35.0	37.0
Select upper(State) from teachers	30.8	32.9
Select substring(State,1,5) FROM teachers	42.5	44.5
Select count(State) from teachers where School_Type = 'Secondary School'	19.1	20.9
Select distinct(State) from teachers where School_Type = 'Secondary School'	46.6	49.1
Select lower(State) from teachers where School_Type = 'Secondary School'	23.4	25.6
Select upper(State) from teachers where School_Type = 'Secondary School'	13.5	15.8
Select substring(State,1,5) FROM teachers where School_Type = 'Secondary School'	23.7	26.2
Select count(State) from teachers where School_Type = 'Secondary School' and School_Type = 'Senior Secondary School'	39.8	42.2
Select count(State) from teachers where School_Type = 'Secondary School' and State = 'Assam'	19.1	21.6
Select distinct(State) from teachers where School_Type = 'Secondary School' and State = 'Assam'	30.2	32.2
Select distinct(State) from teachers where School_Type = 'Secondary School' and State = 'Assam'	31.2	32.9
Select upper(State) from teachers where School_Type = 'Secondary School' and State = 'Assam'	17.6	19.5
Select substring(State,1,5) from teachers WHERE School_Type = 'Secondary School' and State = 'Assam'	22.6	24.3
Select substring(State,1,5) from teachers WHERE School_Type = 'Secondary School' and State = 'Assam'	22.5	24.6
Select * from teachers order by School_Type ASC	19.2	21.2
Select * from teachers order by School_Type DESC	29.7	32.0
Select State from teachers group by State	31.6	34.0
Select students.* from students inner join teachers on students. state = 'Assam'	34.2	36.3
Select * from student where teacherid not in (Select teacherid from teacher)	41.2	45.8

E-GENMR Data Processing with Encrypted and Unencrypted Data

This set of experiment analyzed E-GENMR by processing 21 queries on the Data file of 16 GB as shown below in Table 8 and Fig. 9. These queries are applied on both encrypted and unencrypted data. It has been observed that Data processing time for encrypted data is 1.08 more than the data processing time for unencrypted data, which can be bearable because the advantage of encrypted data are more as it provides security to the system.

Conclusion

As users are comfortable with Relational Databases, in this study a model has been implemented which takes users queries and through the model's compiler these queries gets converted into Map-Reduce key-value form. It is easier to process large amount of data with the help of MapReduce codes as compare to Traditional databases. The model has also been implemented with pre-partitioning and indexing using double hash functions. Model is also flexible in terms of choosing number of mappers and reducers for parallel processing. E-GENMR is evaluated and compared with the latest technologies in the field of Cloud and Big data i.e., HadoopDB, SQLMR, Pig and Hive with respect to the increase in data size. It has been observed from the conducted experiment that the prototype model E-GENMR achieves improvement in query processing time with improvement ratio of 4.17 against HadoopDB, 1.43 against PIG, 1.19 against HIVE and 1.11 against SQLMR for SELECT Query and for JOIN with ORDERBY queries E-GENMR model is 1.28 times faster than SQLMR, 1.59 times faster than HIVE and 6.38 times faster than HadoopDB system. It has also been observed that Data processing time for encrypted data is 1.08 times more as compare to the Data processing time of unencrypted Data, which can be bearable because the advantage of encrypted data are more as it provides security to the system.

In Future, instead of non artificial indexing techniques, artificial techniques can also be applied.

Acknowledgment

This work was supported by Big Data and Cloud Computing Laboratory, Department of computer Science, Jamia Millia Islamia, New Delhi and also supported by "Young Faculty Research Fellowship" under Visvesvaraya PhD Scheme for Electronics and IT, Department of electronics and Information Technology (DeitY), Ministry of Communications and IT, Government of India.

Author's Contributions

Shweta Malhotra: Paricipated in all the experment, coordinated data analysis and contributed to the writing and formatting of manuscript.

Mohammad Najmud Doja: Designed research plan and organized the study also participated in all the experments, coordinated data analysis and contributed to the writing and formatting of manuscript.

Bashir Alam: Paricipated in all the experment, coordinated data analysis, contributed in conceptualizing the idea, drafting the article and helped in developing the Pre-partitioning and MapReduce compiler algorithm.

Mansaf Alam: Paricipated in all the experment, coordinated data analysis, proof reading and helped in developing the mapper and reducer placement algorithm.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and there are no ethical issues involved.

References

Abadi, D., R. Agrawal, A. Ailamaki, M. Balazinska and P.A. Bernstein *et al*., 2016. The backman report on database research. Commun. ACM, 59: 92-99. DOI: 10.1145/2845915

Abouzeid, A., K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz and A. Rasin, 2009. HadoopDB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads. Proc. VLDB Endowment, 2: 922-933. DOI: 10.14778/1687627.1687731

Adamu, F.B., A. Habbal, S. Hassan, R.L. Cottrell and B. White *et al*., 2015. A Survey on big data indexing strategies. SLAC National Accelerator Lab.

Alam, M. and K. Shakil, 2013. Cloud database management system architecture. UACEE Int. J. Comput. Sci. Applic., 3: 27-31.

Bloor, R., 2011. What is cloud database. The Bloor Group.

Condie, T., N. Convway, P. Alvaro, J.M. Hellerstein and R. Sears, 2009. MapReduce Online. Technical Report No. UCB/EECS-2009-136, Electrical Engineering and Computer Sciences University of California at Berkeley.

Dahiphale, D., R. Karve, A.V. Vasilakos, H. Liu and Z. Yu *et al*., 2014. An advanced MapReduce: Cloud MapReduce, enhancements and applications. IEEE Trans. Netw. Service Manage., 11: 101-115. DOI: 10.1109/TNSM.2014.031714.130407

Dean, J. and S. Ghemawat, 2008. MapReduce: Simplified data processing on large clusters. Commun. ACM, 51: 107-113. DOI: 10.1145/1327452.1327492

Fuad, A., A. Erwin and H.P. Ipung, 2014. Processing performance on apache pig, apache hive and MySQL cluster. Proceedings of the International Conference on Information, Communication Technology and System, IEEE Xplore Press, Sept. 24-24, pp: 297-302. DOI: 10.1109/ICTS.2014.7010600

Gani, A., A. Siddiqa, S. Shamshirband and F. Hanum, 2016. A survey on indexing techniques for big data: Taxonomy and performance evaluation. Knowl. Inform. Syst., 46: 241-284. DOI: 10.1007/s10115-015-0830-y

Hsieh, M., C. Chang, L. Ho, J. Wu and P. Lui, 2011. SQLMR: A scalable database management system for cloud computing. Proceedings of the International Conference on Parallel Processing, Sept. 13-16, IEEE Xplore Press, Taipei City, Taiwan, pp: 315-324. DOI: 10.1109/ICPP.2011.54

Jayalath, C., J. Stephen and P. Eugster, 2013. From the cloud to the atmosphere: Running MapReduce across data centers. IEEE Trans. Comput., 63: 74-87. DOI: 10.1109/TC.2013.121

Lee, K., Y. Lee, H. Choi, Y. Chung and B. Moon, 2011a. Parallel data processing with MapReduce: A survey. ACM SIGMOD Record, 40: 11-20. DOI: 10.1145/2094114.2094118

Lee, R., T. Luo, Y. Huai, F. Wang and Y. He *et al*., 2011b. YSmart: Yet another SQL-to-MapReduce translator. Proceedings of the 31th International Conference on Distributed Computing Systems, Jun. 20-24, IEEE Xplore Press, USA, pp: 25-36. DOI: 10.1109/ICDCS.2011.26

Li, K., L.T. Yang and X. Lin, 2011. Advance topics in cloud computing. J. Netw. Comput. Applic., 34: 1033-1034. DOI: 10.1016/j.jnca.2010.07.012

Liu, T., J. Liu, H. Liu and W. Li, 2013. A performance evaluation of Hive for scientific data management. Proceedings of the IEEE International Conference on Big Data, Oct. 6-9, IEEE Xplore Press, USA, pp: 39-46. DOI: 10.1109/BigData.2013.6691696

Li-Yung, H., W. Jan-Jan and L. Pangfeng, 2011. Optimal algorithms for cross-rack communication optimization in MapReduce framework. Proceedings of the IEEE International Conference on Cloud Computing, Jul. 4-9, IEEE Xplore Press, USA, pp: 420-427. DOI: 10.1109/CLOUD.2011.17

Malhotra, S., M.N. Doja, B. Alam and M. Alam, 2015. Generalized query processing mechanism in cloud database management system.

Manyikaetal, J., 2011. Big Data: The Next Frontier for Innovation, Competition and Productivity. 1st Edn., McKinsey Global Institute, San Francisco, CA, USA., ISBN-10: 0983179697, pp; 143.

McCreadie, R., C. Macdonald and I. Ounis, 2012. MapReduce indexing strategies: Studying scalability and efficiency. Inform. Process. Manage., 48: 873-888. DOI: 10.1016/j.ipm.2010.12.003

Mongia, S. and S. Kataria, 2015. Analysis of layer-wise security issues and solutions in cloud database management system. Proceedings of the 2nd International Conference, (IC' 15), Oct. 30-31.

Mongia, S., M.N. Doja, B. Alam and M. Alam, 2013. 5 layered architecture of cloud database management system. AASRI Proc., 5: 194-199.

Pippal, S., V. Sharma, S. Mishra and D.S. Kushwaha, 2011. An efficient schema shared approach for cloud based multitenant database with authentication and authorization framework. Proceedings of the International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Oct. 26-28, IEEE Xplore Press, Barcelona, Spain, pp: 213-218. DOI: 10.1109/3PGCIC.2011.39

Ramamoorthy, S. and S. Rajalakshmi, 2013. Optimized data analysis in cloud using BigData analytics techniques. Proceedings of the 4th International Conference on Computing, Communications and Networking Technologies, Jul. 4-6, IEEE Xplore Press, Tiruchengode, India, pp: 1-5. DOI: 10.1109/ICCCNT.2013.6726631