ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

A New Improved Vertical Partitioning Scheme for Non Relational Databases Using Greedy Method

Jaspreet Kaur, Student of M.Tech (CSE), Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India[1]

Kamaljit Kaur, Assistant Professor (CSE Deptt), Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India[2]

Abstract: Non Relational database is a database used to store large amounts of data. Improving the performance of a database system is one of the key research issues. As publications of context are rising, a new vertical partitioning scheme is proposed to handle better data load and to improve performance for non relational databases. In the proposed work, an algorithm is developed by concatenating the vertical partitioning scheme and greedy algorithm to enhance the performance of data load by altering the vertical partitioning method and to check out the time performance by using the greedy algorithm. In this paper, different performance parameters named data load, time in terms of execution time, communication time are analyzed against data size and nodes using non relational databases so that we get better results from existing algorithm.

Keywords: Non Relational Databases, Vertical Partitioning Scheme, performance, Greedy Algorithm.

1. INTRODUCTION

Non relational databases are a broad class of database management systems identified by non-adherence to the widely used relational database management system model. Non relational databases are not built primarily on tables, and generally do not use SQL as its query language for data manipulation. Other factors which differentiate it are join operations cannot be performed, it doesn't guarantee ACID properties and can be scaled horizontally. In Non relational database, BASE transactions are used instead of ACID transactions (atomicity, consistency, isolation, and durability - four obvious features of traditional relational database systems in relational databases) [1]. BASE transactions are used in non relational databases which mean:-

1. Basically Available (B) -an application works basically all the time. [1]
2. Soft state(S) -Does not have to be consistent all the time. [1]
3. Eventually Consistent (E) -But will be in some known-state eventually. [1]

Non Relational databases emerged as companies, such as Amazon, Google, and LinkedIn and Twitter struggled to deal with unprecedented data and operation volumes under tight latency constraints.
There are various types of non-relational databases which are given below [1]:

- ➢ Key value Stores
- ➢ Document oriented databases
- ➢ Column oriented databases

Other types of non relational databases are-xml databases, graph databases, Object oriented databases etc.

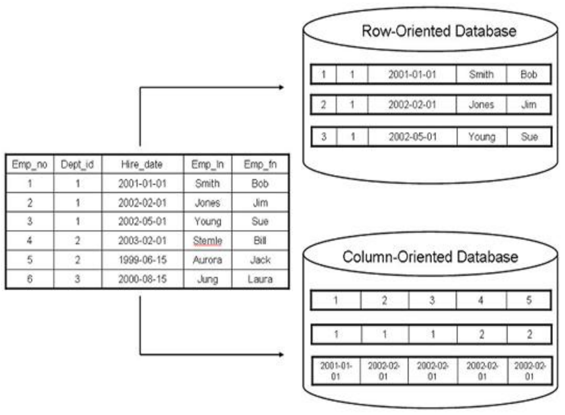
Key-value pairs in the store are organized according to the key. Keys are then assigned to a partition. Once a key is placed in a partition, it cannot be moved to a different partition. Oracle Non-relational Database automatically assigns keys evenly across all the available partitions. E.g. of databases which store key-value pairs are Raik, Redis, Scalaris and Dynamo etc.

In Column-stores each database table store column separately, with attribute values belonging to the same column as compared to traditional database systems that store entire records (rows) one after the other. It is mainly used in OLAP (online Analytical Processing), Data Mining operations. The only main difference between row and column stores

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

is physical storage and query optimization. E.g. of databases to store column oriented are Google big table, Hbase, Cassandra and Pnuts etc. [2]



Emp_no	Dept_id	Hire_date	Emp_ln	Emp_fn
1	1	2001-01-01	Smith	Bob
2	1	2002-02-01	Jones	Jim
3	1	2002-05-01	Young	Sue
4	2	2003-02-01	Stemle	Bill
5	2	1999-06-15	Aurora	Jack
6	3	2000-08-15	Jung	Laura

1	1	2001-01-01	Smith	Bob
2	1	2002-02-01	Jones	Jim
3	1	2002-05-01	Young	Sue

1	2	3	4	5
1	1	1	2	2
2001-01-01	2002-02-01	2002-02-01	2002-02-01	2002-02-01

Fig 1:- Column V/S Row Oriented Database Storage [2]

Document Oriented Databases treat a document as a whole and avoid splitting a document into its constituent name/value pairs. At a collection level, this allows for putting together a diverse set of documents into a single collection. The word document in document databases contains loosely structured sets of key/value pairs in documents, typically JSON (JavaScript Object Notation), and not documents or spreadsheets. Other types of databases are mongo db, Couch db etc. [3]

Advantages of Non Relational Databases
Non relational databases have many advantages which are listed below [3]:

- Non-relational databases process data faster than the relational databases because they do not use ACID properties.
- Non-relational databases have simpler data models than the relational databases.
- Non-relational databases are highly scalable than the relational databases.
- Non-relational databases are very flexible than the relational databases because they are schema less.
- Non-relational databases can handle a very large amount of data.
- Non-relational databases have high performance than relational databases.

1.1 Partitioning Methods
A **partition** is a division of a logical database or its constituting elements into distinct independent parts. Database partitioning is normally done for manageability, performance or availability reasons. A popular and favorable application of partitioning is in a distributed database management system. Each partition may be spread over multiple nodes, and users at the node can perform local transactions on the partition. This increases performance for sites that have regular transactions involving certain views of data, while maintaining availability and security. [4]

Different partitioning schemes are used to handle the work load on distributed databases.
1. Horizontal partitioning is a partitioning method that partitioning the table into no of smaller tables on the basis of rows. It involves putting different rows into different tables. Relational databases introduced the concept of row oriented databases. [4]
2. Vertical partitioning involves creating tables with fewer columns and using additional tables to store the remaining columns. Normalization also involves this splitting of columns across tables, but vertical partitioning goes beyond that and partitions columns even when already normalized.
Different physical storage might be used to realize vertical partitioning. [4] Non relational databases introduced the concept of column oriented databases.

Advantages of Partitioning
Partitioning a database can have the following manageability and performance benefits.

- For transfer or access subsets of data quickly and efficiently, while maintaining the integrity of a data collection.
- To perform maintenance operations on one or more partitions more quickly. The operations are more efficient because they target only these data subsets, instead of the whole table.
- For improving query performance, based on the types of queries you frequently run on your hardware configuration.

2. PROBLEM FORMULATION

In view of limitations of existing partitioning algorithm which are more prone to load crashes, time line delay and are less scalable. The proposed work deals with the cost in terms of time and load sharing of the system by resolving all the existing problems using non-relational databases. A new improved algorithm is proposed by concatenating the vertical partitioning scheme and greedy approach. The proposed work enhances the performance of data load by altering the vertical partitioning method and to check out the time performance by using the greedy method. Different performance parameters named data load, time in terms of execution time, communication time are checked against different data sizes and nodes using non relational databases. The need of proposed work is:-

- To have better scalability results from existing algorithm.
- To have better utilization of resources using load sharing.

3. PROPOSED ALGORITHM (VPartition)

In order to overcome the limitations of existing partitioning algorithm; a new algorithm named VPartition, is proposed that improves the performance of the existing algorithm. The algorithm includes an improved vertical partitioning scheme where cost is calculated in terms of time using greedy method. This results in improved data load as compared to previous approach.

VPartition (I, T, T_I, w, p, r, D, $C_{comm.}$, C_{route}, N)

VPartition (**I**: counter initialized to 0, **T**: time, **T_i**: time at ith second is initialized to 0, **w**: weight of database, **p**: partition data, **r**: path function, **D**: subset of query, **$C_{comm.}$**: cost of comm., **C_{route}**: cost of path finding, **N**: no of servers)

```
Call Greedy Algorithm
$T_i$ = cost (w, |D|)     // Cost Function ………. (1)
For all [Cost (r (I, w)) = $C_{route}$] // Path Function….. (2)
$C_{comm.} = C_{request} + C_{response}$   // Total communication cost for databases
$T\ comm.(w,i,N) = \begin{cases} 0, & \text{if } r(N,w) = i \\ Ccomm., & \text{if}(r(N,w)! = i) \end{cases}$
//Communication Time
$T_{comm.} = (C\ comm.) - (C\text{comm.} |N)$  // For N servers
$T_N = C_{route} + C_{comm.} - C_{comm.} / N + cost(w, |p|)$  // Total time for N servers
$S(N) = T_1 / T_N$          // Speed
```

Pseudo code 3.1: VPartition

```
I =0;
$T_i$ =0;
For (I=0; I< No_Of_Queries; I++)
Cost = Weight_Of_Select_Database + Subset_of_Query;
End For
```

Pseudo code 3.2: Cost Function described in Equation 1of Pseudo Code 1

```
I =0;
$T_i$ =0;
For (I=0; I< No_Of_Queries; I++)
Path = Cost_Of_Weight_Of_Select_Database + Time;
End For
```

Pseudo code 3.3: Path Function described in Equation 2 of Pseudo Code 1

3.4 Greedy Algorithm

Greedy algorithm assigns one reduce task to a rack at a time. When assigning a reduce task to a rack, it chooses the rack which incurs minimum total traffic if the reduce task is assigned to that rack so that to minimize the maximum total traffic for all racks in the data set when given the number of mappers in each rack.

Greedy Algorithm (**N**: No of racks, **M**: No of mappers, **R**: no of total reducers, **{m_1, m_2 …m_n}:** the number of mappers on each rack, **{r_1, r_2…r_n}:** A reducer state tuple, **State_tuples [N]:** {0, 0 ….0})

```
For i=1 to R do
Minimal ← infinite
For j=0 to N do
Traffic = $(M-2m_j)$. (state_tuple [j] +1) + $m_j R$
If traffic <minimal then
Candidate = j
```

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

```
End if
End for
State_tuple [candidate] ++
End for
Return state_tuple
```

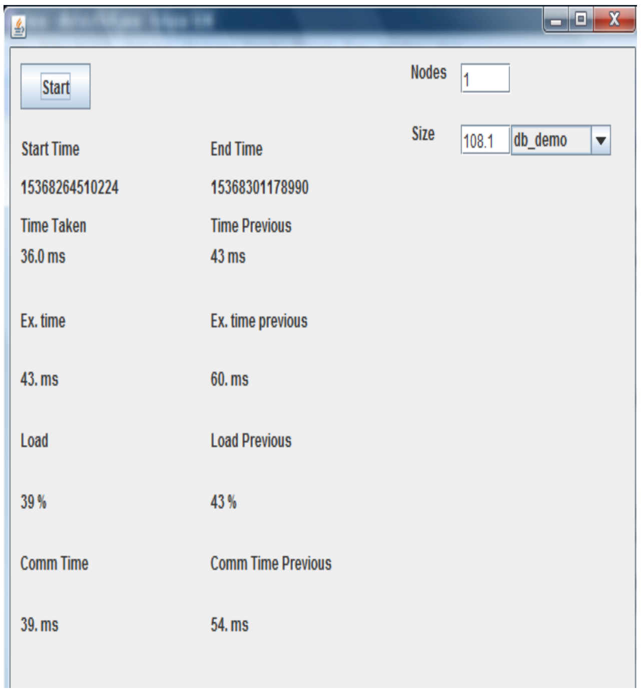
Pseudo code 3.4: Pseudo code for Greedy Algorithm [11]

4. IMPLEMENTATION RESULTS

The result analyses of the proposed work are carried out on java platform and various performance parameters are discussed using Couch db as non relational databases. Apache Couch DB is a free and open-source non relational database. It is a top-level project of the Apache Foundation and it is written in Erlang, java a programming language aimed at concurrent and distributed applications. It complies with the ACID properties, providing serialized updates and with the use of MVCC reads are never locked. It is distributed in nature and supports various types of replication schemes and conflict resolution. In the implementation comparative analysis of different databases of various data sizes and different number of nodes are used in order to analyze the data load, time in terms of execution time, communication time and total time taken with existing work and the time taken to execute the query.

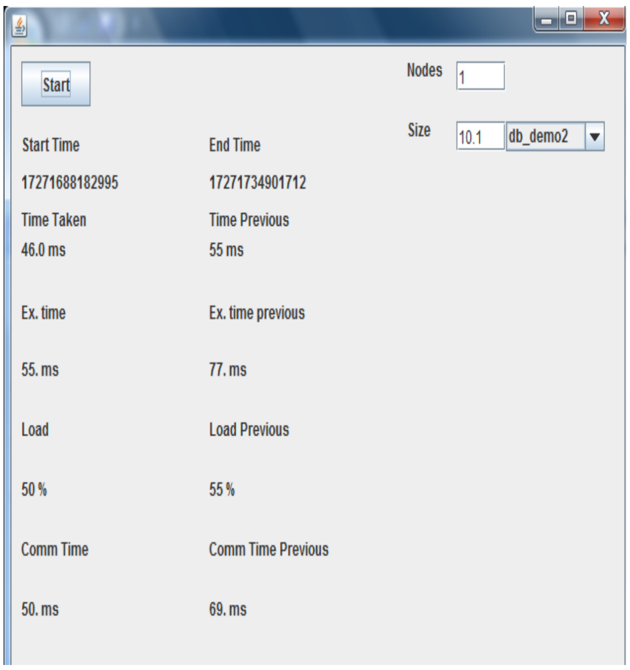
- **Comparative analysis using different data sizes in terms of load, execution time, comm. Time and total time with existing work.**

VPartition algorithm is executed by providing different input parameters (time, execution time, communication time and load) using different data sizes.



Start Time 15368264510224 End Time 15368301178990
Time Taken 36.0 ms Time Previous 43 ms
Ex. time 43. ms Ex. time previous 60. ms
Load 39 % Load Previous 43 %
Comm Time 39. ms Comm Time Previous 54. ms

Fig 4.1 Using database (db_demo) at size 100KB with one node



Start Time 17271688182995 End Time 17271734901712
Time Taken 46.0 ms Time Previous 55 ms
Ex. time 55. ms Ex. time previous 77. ms
Load 50 % Load Previous 55 %
Comm Time 50. ms Comm Time Previous 69. ms

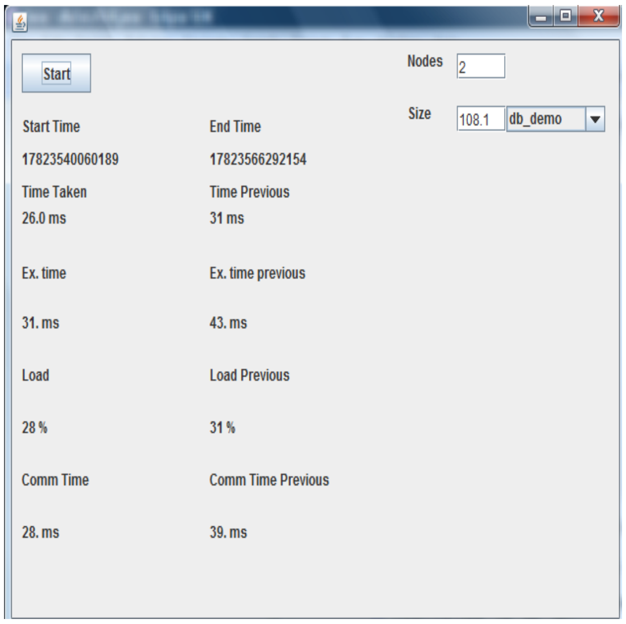
Fig 4.2 Using database (db_demo2) at size 10MB with one node

- **Comparative analysis using increase in number of nodes.**

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

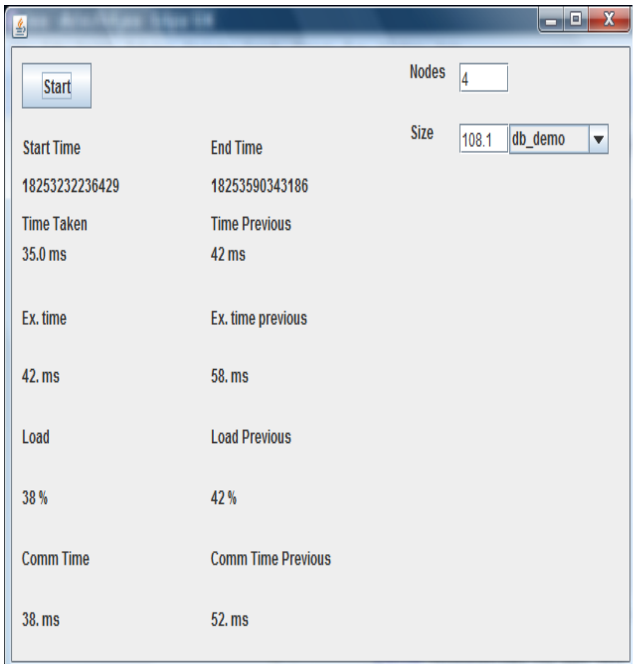
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

VPartition algorithm is executed by providing different input parameters (time, execution time, communication time and load) with increase in no of nodes as data size is constant.



```
Start                                      Nodes  2
                                           Size   108.1  db_demo
Start Time          End Time
17823540060189      17823566292154
Time Taken          Time Previous
26.0 ms             31 ms

Ex. time            Ex. time previous

31. ms              43. ms

Load                Load Previous

28 %                31 %

Comm Time           Comm Time Previous

28. ms              39. ms
```

Fig 4.3 Testing with Two Nodes at Data Size 100KB with Database as db_demo

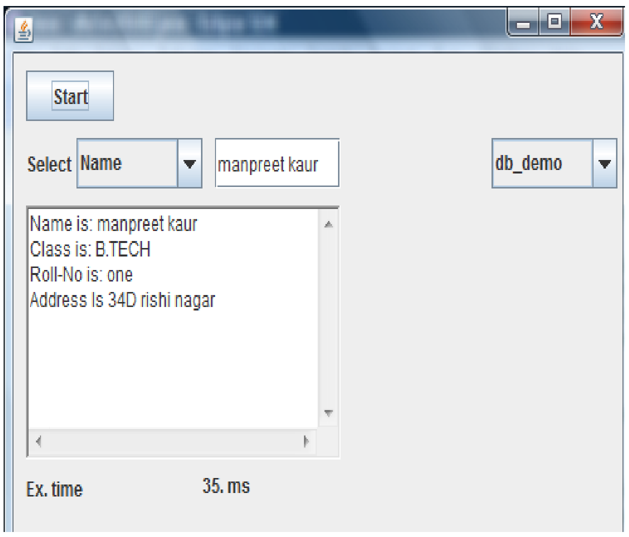


```
Start                                      Nodes  4
                                           Size   108.1  db_demo
Start Time          End Time
18253232236429      18253590343186
Time Taken          Time Previous
35.0 ms             42 ms

Ex. time            Ex. time previous

42. ms              58. ms

Load                Load Previous

38 %                42 %

Comm Time           Comm Time Previous

38. ms              52. ms
```

Fig 4.4 Testing with four Nodes at Data Size 100KB with Database name db_demo

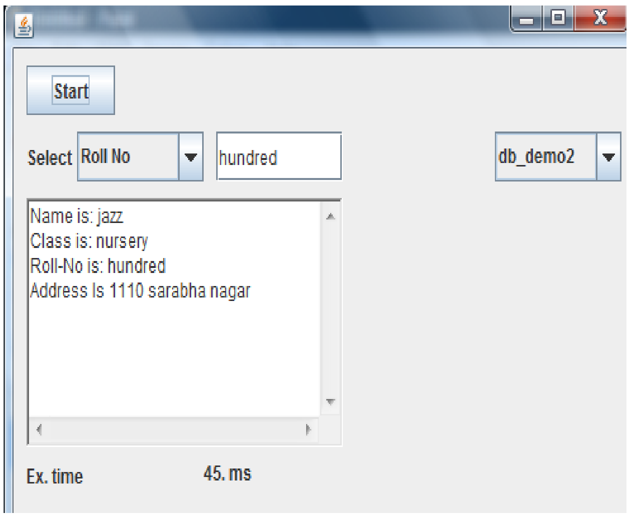
To execute a query and execution time is calculated according to particular query.

In the implementation, a query is executed according to various databases and execution time is calculated according to particular query. On selecting any value of particular field, all records of selected database are display on the screen and execution time is calculated according to query.



```
Start

Select  Name  manpreet kaur                db_demo

Name is: manpreet kaur
Class is: B.TECH
Roll-No is: one
Address Is 34D rishi nagar

Ex. time            35. ms
```

Fig 4.5 Image for Name Query using database as (db_demo)



```
Start

Select  Roll No  hundred                   db_demo2

Name is: jazz
Class is: nursery
Roll-No is: hundred
Address Is 1110 sarabha nagar

Ex. time            45. ms
```

Fig 4.6 Image for Roll no. query using database as (db_demo2)

5. GRAPHICAL ANALYSIS

- **Graphical analysis using different data sizes in terms of load, execution time, communication time and total time with existing work.**

In the graphs, all input parameters are checked 5 times for different data sizes of different databases named as db_demo, db_demo1, db_demo2, db_demo, db_test are discussed. E.g. first check the execution time parameter 5 times at 100KB

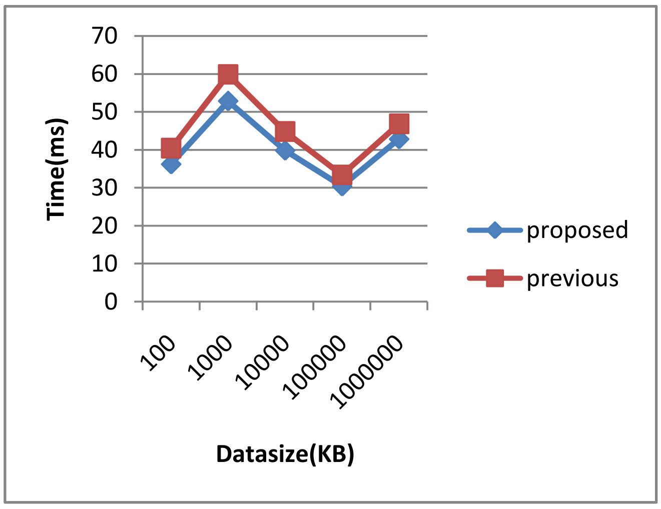
ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

for db_demo database, then at 1MB, 10MB and so on. Then calculate the mean of depicted values of various input parameters for different data sizes. The formula for mean:-

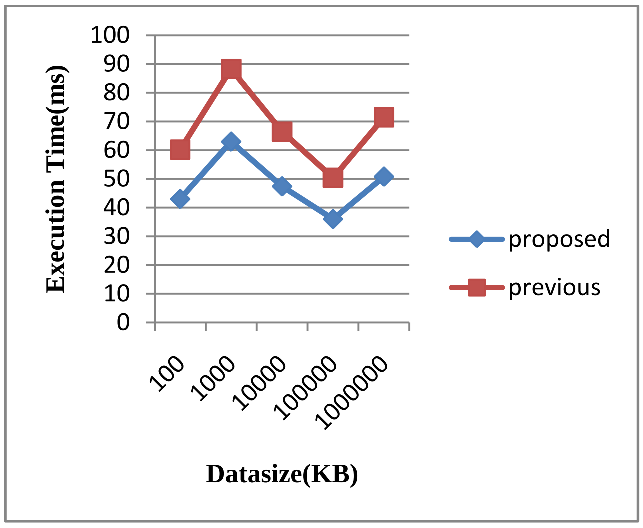
Mean= Sum of the depicted values at data size (100KB) /total no of cases
Total no of cases =5
Different databases with different data sizes are named as db_demo =100KB, db_demo1=1MB, db_demo2=10MB,db_temp=100MB, db_test=1000MB.

Different performance parameters are analyzed using different data sizes. Results are shown below in the graph with previous approach.



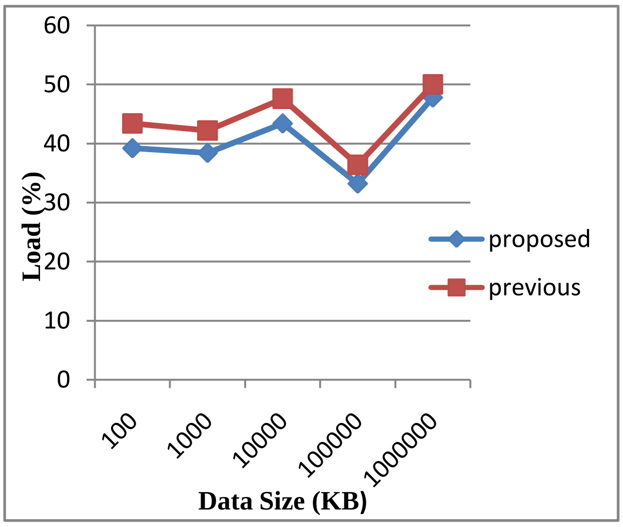
Graph 5.1 Comparison graph of Data size v/s Time

From the above graph, it has been concluded that data size increases, time also increases. Time of proposed algorithm is quite less than previous time. There is a little degradation in proposed work as compared to existing work.



Graph 5.2 Comparison graph of Execution Time v/s Data size

From the above graph, the execution time of proposed algorithm is quite less than previous execution time as shown in graph. All the time in terms of execution time is reduced as compared to previous results by proposed algorithm using greedy method. As the data size increase there is a little increase in execution time.

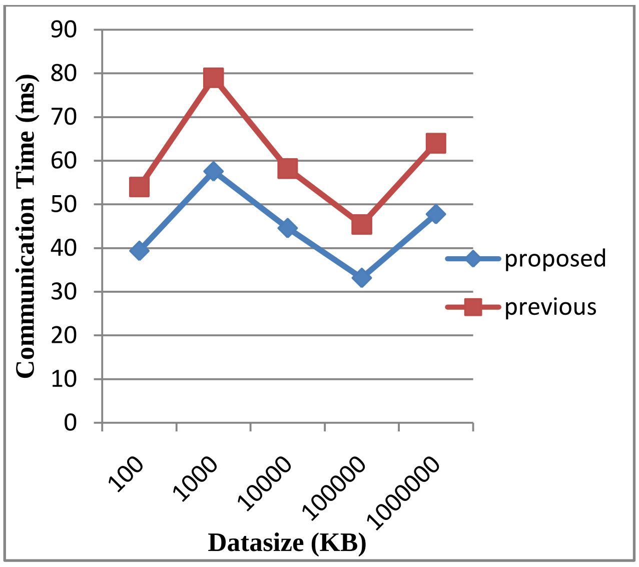


Graph 5.3 Comparison graph of Load v/s Data size

From the above graph, the load and resource utilization of proposed algorithm is quite less than previous load and resource utilization as shown in graph. All the cost in term of time is reduced in algorithm due to the vertical partitioning approach clubbing in proposed scenario.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013



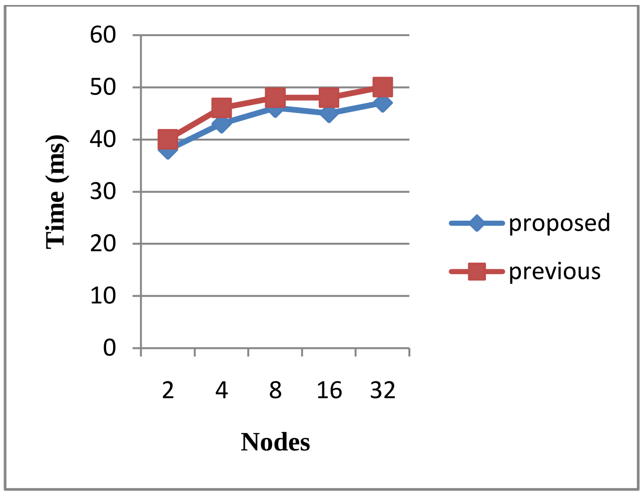
Graph 5.4 Comparison graph of Comm. time v/s Data size

From the above graph, the communication time of proposed algorithm is quite less than previous communication time as shown in this graph. Time in terms of communication time is reduced in our algorithm due to the greedy method clubbing in our proposed work. Communication time of proposed at 100KB takes an average of 39ms as compared to previous one ,it takes an average of 52ms so there is a little degradation in performance of communication time as compared to previous approach with increase in data size.

- **Graphical analysis using different no. of nodes in terms of load, execution time, communication time and total time with existing methods.**

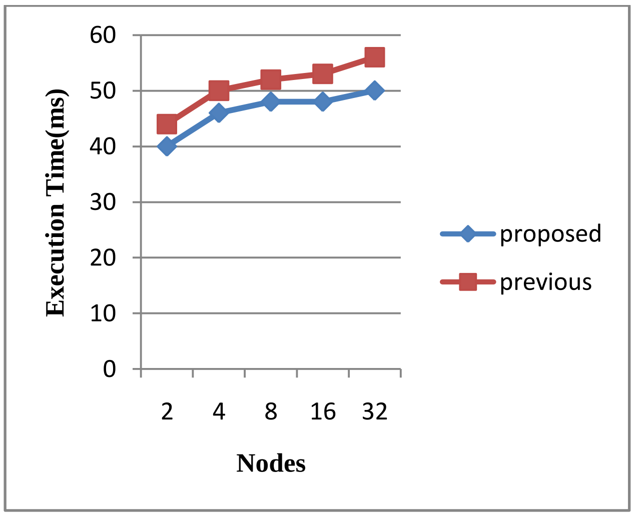
In the graphs, all input parameters are checked 5 times according to increase in no. of nodes with data size to be constant of different databases named as db_demo, db_demo1, db_demo2, db_demo, db_test. E.g. first test the execution time parameter 5 times with two nodes, then with 4 nodes and so on for one database (db_demo) to be constant always. Then calculate the mean of depicted values of various input parameters for different no. of nodes. The formula for mean:-

Mean= Sum of the depicted values with two nodes, four nodes……. /Total no of cases
Total no of cases =5
Different performance parameters are analyzed using increase in no of nodes. Results are shown below in the graph.



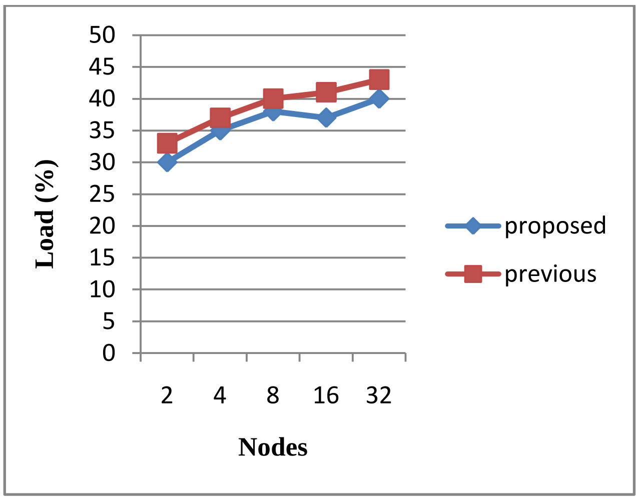
Graph 5.5 Comparison graph of Nodes v/s Time

From the above graph, it has been concluded that within increase in no of nodes, time also increases. Time of proposed algorithm is quite less than previous time using greedy algorithm. There is a little degradation in proposed work as compared to previous method.



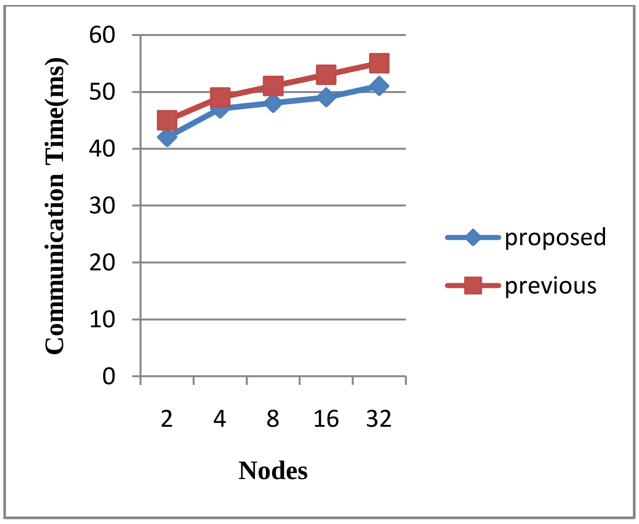
Graph 5.6 Comparison graph of Execution Time v/s Nodes

From the above graph, the execution time of proposed algorithm is quite less than previous execution time in case of n number of nodes as shown in graph. Execution time analyzed with four nodes takes an average of 45ms as compared to previous approach; it takes an average of 50ms execution time. There is a little degradation in performance of execution time acc to increase in no of nodes as compared to previous approach.



Graph 5.7 Comparison graph of Load v/s Nodes

In the above graph, the load and resource utilization of proposed algorithm is quite less than previous load as shown in graph. Data load is reduced in our algorithm due to the vertical partitioning approach clubbing in proposed scenario. Data load analyzed with two nodes takes an average of 30% as compared to previous algorithm; it takes an average of 33% load. There is a little degradation in performance of load acc to increase in no of nodes as compared to previous approach.



Graph 5.8 Comparison graph of Comm. Time v/s Nodes

In the above graph, the communication time of proposed algorithm is less than previous communication time as taking n number of nodes as shown in graph. Time in terms of communication time is reduced in our algorithm due to the greedy algorithm clubbing in proposed work.

6. CONCLUSION AND FUTURE WORK

In context management platform in which number of context publications always keep rising, so to handle better data load, a new vertical partitioning scheme is proposed for non-relational databases using greedy algorithm. Our proposed work is heading toward the improvement in Database execution time and also in load sharing. The proposed work enhances the performance of data load by altering the vertical partitioning method and to check out the time performance by using the greedy method. In the existing algorithm, the load and time parameter is quite higher which are more prone to load crashes, time line delay and less scalable. Different performance parameters are graphically analyzed in terms of execution time, communication time and load with different data sizes and increase in no of nodes with previous results. From the results, it has been concluded that the time in terms of execution time, communication time and load sharing is approximately half of previous method using new improved vertical partitioning scheme which leads to better scalability, robustness and better utilization of resources. There is a little degradation in performance of proposed scheme with increase in data size and with increase in no of nodes as compared to previous method.

In future work, a hybrid vertical partitioning scheme can be implemented for distributed database using all non-relational databases to check the issues of scalability and robustness.

REFERENCES

[1] C. Strauch , "*NoSQL databases*", February 2011. [Online].Available:http://www.christof- strauch.de/nosqldbs.pdf.

[2] Column oriented database technologies [online] Available: http://dbbest.com/blog/ Column-oriented-database-technologies/

[3] Sharma Vatika, Dave Meenu, "*Comparison of SQL and NoSQL Databases*", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 8, August 2012.

[4] Partitioning [online.] Available: http://en. Wikipedia.org/wiki/Partition (database)

[5] Muthura. J, Chakravarthy.S, Varadarajan.R , Navathe S.B, "*A Formal Approach to the Vertical Partitioning Problem in Distributed Database Design*", In Proccedings of the second international conference on Parallel and distributed information systems, August 1992, pp. 26-35.

[6] Khan Shahidul Islam, Latiful Hoque Dr. A. S. M., "*A New Technique for Database Fragmentation in Distributed Systems*", In International Journal of Computer Applications, Volume 5- No.9, August 2010.

[7] Abuelyaman Eltayeb Salih, "An Optimized Scheme for Vertical Partitioning of a Distributed Database" International Journal of Computer Science and Network Security, Volume 8 -No.1, January 2008.

[8] Lakshman Avinash, DeCandia Giuseppe, Hastorun Deniz, Madan Jampani, Gunavardhan Kakulapati, Lakshman Avinash, "*Dynamo: Amazon's Highly Available Key-value Store*", In proceeding of symposium on operating systems principles, October 2010.

[9] Gomes Diogo, Jogao Gonc, Ricardo Santos and Rui L Aguiar, "*XMPP based Context Management Architecture*", In IEEE Globecom, Dec 2010.

[10] Jindal Alekh and Dittrich Jens, "*Relax and Let the Database Do the Partitioning Online*", In Proc. BIRTE, June 2011, pp.65-80.

[11] Li-Yung Ho, Jan-Jan Wu and Pangfeng Liu, "*Optimal Algorithms for Cross-Rack Communication Optimization in Map Reduce Framework*" IEEE International Conference on Cloud Computing , June 2011, pp.420-427.

[12] Cattell Rick, "*Scalable SQL and NoSQL Data Stores*", Communications of the ACM, December 2011.

[13] Santos Nuno, Pereira Oscar M. and Gomes Diogo, "*Context Storage Using NoSQL*", Conferência sobre Redes de Computadores , Coimbra, Portugal, Volume 2, Nov 2011.