IOSR Journal of Engineering
May. 2012, Vol. 2(5) pp: 1168-1176

A Study on Performance Evaluation of Peer-to-Peer Distributed Databases

Dr. D.I. George Amalarethinam[1], C. Balakrishnan[2]
1(Director-MCA, Associate Professor, Department of Computer Science, Jamal Mohamed College (Autonomous), Tiruchirappalli, India)
2(Assistant Professor, Department of Computer Science, St. Joseph's College (Autonomous),Tiruchirappalli, India)

ABSTRACT
The design phase of the distributed database environment holds a vital part in affecting the performance. The Peer-to-Peer architecture gives a great degree of hope to handle the data in an efficient manner. This work analyses a cluster based Peer-to-Peer architecture named FlexiPeer for the distributed databases to address the fragmentation and allocation phases of database design. This work takes the inspiration of the previous works done based on the predicate based fragmentation and introduces the clustering approach for drafting the database architecture and to allocate the fragmented data across the sites. The performance of the FlexiPeer is studied in a simulated environment.

Keywords: Clustering approach, FlexiPeer, Fragmentation and allocation, Peer-to-Peer databases, priority factor values

1. Introduction
At the heart of the idea of a distributed system, Distributed database is the distribution of data over multiple sites and is a collection of multiple, logically interrelated databases distributed over a computer network [1]. There will be a possibility of improved response times to queries and upgrading system capacity or performance incrementally.
Distributed database design is one of the major research issues in the area of distributed database system. A technique of breaking up the database into logical units, which may be assigned for storage at the various sites called data fragmentation and allocation.
Fragmentation can be horizontal, vertical and mixed or hybrid. Allocation describes the process of assigning each fragment or each copy of a fragment to a particular site in the distributed system [1].
Peer-to-Peer (P2P) based distributed database technology has no strict definition; it is generally described as having a structure that is contrast to the traditional client-server model. Each node in the network acts as both client and server, requesting data from neighboring nodes as well as routing and serving data for others. The nature of P2P technology makes it well suited for storing multiple copies of data between several nodes, in turn offering reliable access to data and distributing the load of requests. All the features inherent in P2P technology promise a network that is dynamic, scalable and reliable.
Of the several issues in P2P based distributed database environment, the basic and first and foremost problem is to know the location of neighbors [2]. Without the knowledge of the neighbors the unsuccessful queries cannot be transformed across the network to find the appropriate data to execute the query.
The above narrated problem is addressed in two ways in general file sharing systems, such as, Chord and Freenet. Since, the Chord and Freenet are widely used in data sharing P2P environments, the characteristics of the two concepts encourage the research directions to include Chord and Freenet in Database environment. Chord was designed to create a network that is reliable, scalable, and decentralized [3]. Chord uses consistent-hashing, a method that evenly distributes hash keys to nodes. Each node contains a finger table of its neighbors and their possible assignments of keys. Nodes are organized in a ring topology, maintaining keys of values that are less than or equal to the assigned node value and greater than the value of the node's predecessor. Unsuccessful queries are forwarded around the ring to successive nodes, which allows each node only track on the order of 'log N' neighbors, where 'N' is the total number of nodes in the network.
The design goals of Freenet were to create a completely decentralized, scalable peer-to-peer application allowing anonymous input, retrieval and storage of data [4]. Unlike Chord, Freenet assigns hash keys to specific items (data). Like Chord, Freenet nodes contain a table of information about their neighbors.
In this paper cluster based architecture of the distributed databases to address the fragmentation and allocation phases of database design has been introduced. This work takes the inspiration of the previous works done based on the predicate based fragmentation and introduces the clustering approach for drafting the database architecture and for allocating the data across the sites.
The paper is organized as follows. The next section of this work presents literature reviews of fragmentation, allocation, clustering, Chord and Freenet. Section III describes the FlexiPeer architecture. In Section IV implementation details are presented. The Section V illustrates the evaluation of simulation of Chord and FlexiPeer architectures. Finally Section VI concludes the paper with future research directions.

2. Literature review
Most of the research related to fragmentation and allocation has been carried out in the context of relational databases.
Navathe [5] has proposed a mixed fragmentation method for distributed database design at the initial level and a mixed fragmentation tool to partition relations using a grid

approach. It is based on a graph theoretic algorithm which clusters a set of attributes and predicates into a set of vertical and horizontal fragments, respectively. Karlapalem and Li [6] made a study on different types of partitioning schemes in object oriented databases. Horizontal fragmentation algorithm for distributed deductive database systems has been proposed by Lim et. Al [7]. This algorithm handled the horizontal fragmentation by clustering all the tuples in a base relation that are used by queries. Lim and Yiu-Kai Ng [8] presented different approaches for vertical fragmentation of relations and allocation of rules and fragments. It helps to maximize locality of query evaluation and minimizes communication cost and execution time during processing the queries.

Zhou and Sheng [9] tried to solve the vertical fragmentation problem and fragment allocation problem together. Bellatreche et al., [10] made a study on horizontal fragmentation in the object-oriented model. Huang and Chen [11] proposed a simple and comprehensive model for a fragment allocation problem. Also, they have developed Huang and Chen, two heuristics algorithms to find an optimal allocation of the fragments.

Ahmad et al., [12] have addressed the allocation of fragments problem in distributed database system. They have developed a query driven data allocation approach. Various algorithms based on evolutionary computing paradigm have also been proposed by them. Du et al., [13] have proposed new algorithms based on a new measurement to evaluate togetherness among the attributes in a relation. An incremental re-fragmentation method was proposed by Ezeife and Dey [14] and Darabant et al., [15] to define new fragments more quickly. It helps to save system resources and make data to be easily available for network and web access. Grebla et al., [16][17] focused only on allocation problem of fragments. They used mobile intelligent agents to provide a solution in allocation problem for distributed database systems. They have also proposed a new method for horizontal partitioning of classes with complex attributes and methods, using AI clustering techniques.

Darabant et al., [18,19,20,21] proposed some methods for horizontal fragmentation of objects with complex attributes and some methods were based on different similarity measures applied in hierarchical agglomerative clustering algorithms. They rely on AI clustering techniques for grouping objects into fragments. Hababeh et al., [22] proposed a method for allocating fragments to a cluster. Sites in the distributed database systems are grouped based on their communication cost. A method for incrementally maintaining the primary horizontal fragments of an object oriented database has been proposed by Campan et al., [23]. Abdalla and Marir [24] made a comparative study on vertical partitioning algorithms to find the most efficient vertical partitioning schema. Ma et al., [25] addressed vertical fragmentation and allocation simultaneously in the context of the relational model. A heuristic approach to vertical fragmentation, which uses a cost model is followed and is targeted at globally minimizing the costs. Hui Ma and Markus Kirchberg [26] presented a cost-based approach for horizontal and vertical fragmentation. Algorithms were presented for each of the fragmentation techniques used in distribution design to obtain fragmentation schema, which would improve the system performance.

Eltayeb Salih Abuelyaman [27] proposed a vertical partitioning algorithm for improving the performance of database systems without the knowledge of empirical data. The algorithm uses the number of occurrences of an attribute in a set of queries rather than the frequencies of queries accessing these attributes. John and Saravanan [28] proposed a new algorithm for vertical partitioning in object-oriented databases using intelligent agents based on attributes and methods. Arjan Singh and K.S. Kahlon [29] proposed a new dynamic data allocation algorithm for non-replicated distributed database system. The proposed algorithm reallocates data with respect to the changing data access patterns with time constraint. This algorithm will decrease the movement of data over the network and also improve the overall performance of the system.

Shahidul Islam Khan and A. S. M. Latiful Hoque [30] have proposed a new technique of fragmentation to solve the problem of taking fragmentation decision at the initial stage of a distributed database design, according to the attribute locality precedence table. Nilarun Mukherjee [31] proposed a new dynamic fragment allocation algorithm in Non-Replicated allocation scenario incorporating the time constraints of database accesses, the volume threshold and most importantly the volume of data transmitted in successive time intervals to dynamically reallocate fragments to sites at runtime in accordance with the changing access patterns. It was helpful to decrease the movement of fragments over the network and data transfer cost and improve the overall performance of the system by dynamically allocating fragments in a most optimum and intuitive manner. Dimovski et al., [32] presented a novel formal approach for horizontal partitioning of relations based on predicate abstraction.

Several works were carried out about Chord [33] only on Network based file sharing systems. Hence, this paper tries to employ the concept of Chord in P2P based data management. Chord methodology was proposed to assist a node in a P2P network to know about its neighbors using a look up table with a layered approach and worked using three bit identifier space. The Chord concept evaluated using consistent hashing to assign keys to Chord nodes [34]. An effective routing algorithm [35] was developed for P2P overlays using small lookup paths with one-hop and two-hop routing schemes to execute the queries without re-routed. During the operation of P2P systems, the maintenance bandwidth [36] is a very important issue. This issue was analyzed using Chord and an algorithm was analyzed to converge to a correct routing state from an arbitrary initial condition. The operations of Chord were examined in the process of finding neighbors in growth restricted metrics [37], this is very useful in Internet and vector quantization based applications. The characteristics and performance analysis of Freenet [38,39] were explained as an un-structured P2P network architecture.

3. Flexi Peer Architecture

The proposed architecture for FlexiPeer follows the clustering of sites based on the locality priority factor factor. The block diagram of proposed architecture is shown in Fig. 1.

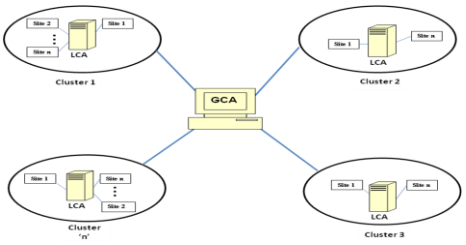

Figure 1. Clustered approach for FlexiPeer architecture

As mentioned in Fig. 1 the sites are clustered with its locality priority factor value and each cluster will be managed by Local Cluster Administrator (LCA) and the whole architecture is administered by Global Cluster Administrator (GCA). The clustering process at the top level (architecture level) is done by the GCA based on the unique number of regions of the sites using Site Information Table (SIT). SIT will give details about each site. It contains information such as site ID, Locality and Region using which clustering of sites is done. The attributes of sites such as, Local Cluster Identification (LCA_id), Site identification (site_id), region of the cluster (cluster_region), Location of the site (site_location) and type of the data stored in that site (site_type) will be handled by LCA of the respective cluster. The global attributes of all clusters like, Global cluster identification (GCA_id), Local Cluster identification (LCA_id) and the region of the cluster (cluster_region) are maintained by GCA of the architecture.

The LCA and GCA are equipped with the functions like, Validator, which validates the relevance of the query. The queries that dissatisfy the criteria expected by Validator will be rejected. Hence, wastage of processing capacity with irrelevant queries is reduced. The LCA Resource Checker finds the appropriate site and data within the cluster and the GCA Resource Checker finds the respective Cluster which owns the required data. The LCA Forwarder, will forward the un-successful queries to GCA and GCA Forwarder will re-direct the query to the appropriate cluster (LCA of the cluster).

Based upon the region of the sites, the sites are categorized by the GCA and send the values to the LCA. When the database is submitted for fragmentation and allocation, the GCA will fragment the records horizontally based on the region value mentioned as one of the field of the record. This group of records is framed as a sub-relation and is given to the respective LCA. The particular LCA takes the next highest priority factor value of the sub-relation (type of data) and re-fragment the records horizontally and creates multiple numbers of sub-relations (equivalent to the number of unique values in the type of data). After this re-fragmentation, the LCA compares the number of sites within the cluster and number of sub-relations derived from re-fragmentation, if the number of sites is more, then the LCA will once again horizontally fragment a sub-relation which has large number of records based on the next highest priority factor valued attribute.

The relational algebraic notations for the fragmentation and re-fragmentation of relation is as follows in the Equations 1,2 and 3.

Let

R be the Relation
n be the number of sites
t_i are the tuples of the relation
t_h is set of ordered tuples based on highest priority factor value
t_{hl} is a tuple having highest priority factor value
t_{hn} is a tuple having next highest priority factor value
t_{hm} is a tuple having next highest priority factor value
SR_i be the Sub-relations of original relation for Clusters
RSR_i be the re-fragmented relations of sub-relation for the sites within the Cluster
NSR_i be the next level re-fragmented relations of re-fragmented relations for the newly added sites of the Cluster

Equation 1 is the predicate for Fragmentation of relation into sub-relations for Clusters

$$SR_i \rightarrow {}^{\sigma} t_{hx}{}^{(R)} \text{ where } i = 1\ldots n \text{ and } x = 1 \quad (1)$$

Equation 2 will be used for Fragmentation of sub-relation into re-fragmented relations for sites

$$RSR_i \rightarrow {}^{\sigma} t_{hy}{}^{(SR)} \text{ where } i = 1\ldots n \text{ and } y = 2 \quad (2)$$

Equation 3. Fragmentation of re-fragmented relations into sub-relations for newly added sites or the number of sites more than the number of re-fragmented relations

$$NSR_i \rightarrow {}^{\sigma} t_{hz}{}^{(RSR)} \text{ where } i = 1\ldots n \text{ and } z = 3\ldots n \quad (3)$$

After fragmentation and allocation processes, query processing can be handled by LCA and GCA. The working methodology of LCA, GCA and Fragments and Sites Cluster Algorithm is given in the following sections.

3.1 LCA

The LCA in FlexiPeer architecture works in the following mechanism. The query that requires data will be submitted to any node of the architecture. The node is equipped with resource analyzer, this will take responsibility of checking the query syntax and required data is found in the particular node or not. If a query satisfies the syntax check and data are not found, then, that node will redirect the query to the LCA of that cluster. The respective LCA of that node will receive the query with its identifier and checks whether the query comes from a node or from GCA. If the query comes from a node, the Validator of LCA checks the query for redundancy and relevance; this will work as a semantic checker. If a query satisfies the requirements during validation, it will be sent to the Resource Checker, which owns a relation contains the information about the sites of that particular cluster. If the Resource Checker finds the required data within any of the sites of that particular

cluster, it directs the query to the Executor for execution. Then the Executor fetches the required data from the specified site and query will be processed to publish the result. If the query requirement fails during Resource Checking, the Resource Checker will pass the query to the Forwarder to forward the query to GCA for further execution. The workflow of LCA is shown in Fig. 2.

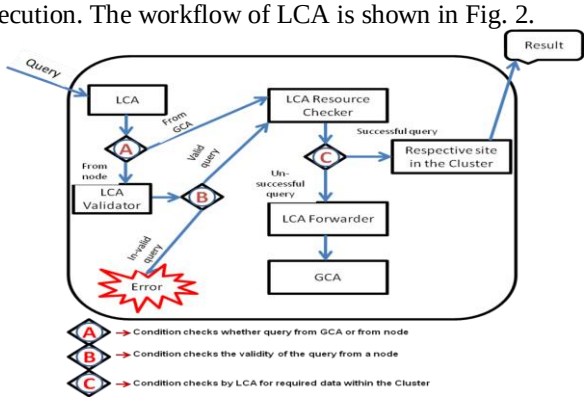

Figure 2. Flow diagram of LCA

PSEUDO CODE FOR LCA:

```
LCA (query_id qi, site_id ni, flag f)
{
   li = LCA id;
   FLAG_CHECK (f)
   {
        if f = 0              // query from a node
        VALIDATOR(qi, ni);
        else
        {
          LCA_RESOURCE_CHECK (qi, ni)
                //query from GCA & LCA receives query
            {
                if (data available)
                EXECUTOR (qi, ni);
                        // data found within cluster
                else
                   FORWARDER (qi);
                        // data not found in cluster
                                }
                    }
                }
            VALIDATOR(qi, ni)
             {
                let irrelevant = return (if (qi is the
                        copy or irrelevant query))
                irrelevant→0      // no redundancy
   LCA_RESOURCE_CHECK (qi, ni);
        else
        irrelevant→1; display ERROR; // irrelevant query
                        }
                 FORWARDER (qi, li)
                {
                Forward query to GCA along with
                LCA-id; // data not found in cluster
                  }
        }
```

3.2 GCA

The GCA in FlexiPeer architecture works in the following mechanism. The GCA will receive the query with its identifier which is given to that node for execution. The GCA Resource Checker checks the query for redundancy and relevance. If a query satisfies the requirements during validation, it will be sent to the Forwarder to forward the query to the LCA of the corresponding cluster for execution. If it fails during validation, an error will be created and the process will be terminated.

The workflow of GCA is shown in Fig. 3.

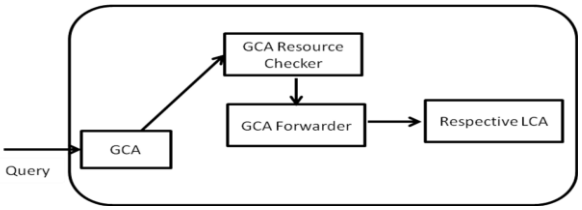

Figure 3. Flow diagram of GCA

PSEUDO CODE FOR GCA:

```
        GCA (query_id qi, LCA_id li)
        {
            GCA_RESOURCE_CHECK (qi)
                        //Assumes that GCA receives
query
            {
              let li = return (respective LCA-id owns data)

GCA_FORWARDER (qi, li)
                    // data found in cluster
                    }
        GCA_FORWARDER (qi, li)
        {
           Forward query qi to LCA li;
                        // data found in li
                }       }
```

4. Cluster algorithm for fragmentation

Shahidul Islam Khan and A. S. M. Latiful Hoque [30] have proposed a new technique of fragmentation to solve the problem of taking fragmentation decision at the initial stage of a distributed database. By taking their work as an inspiration, a new architecture has been proposed by incorporating the clustering approach for architecture drafting. Initially horizontally fragment the relation based on the highest priority factor based on the attribute locality, and re-fragment the relations within the cluster using the next highest priority factor to allocate the sub-relations in the sites of the cluster.

A relation is horizontally fragmented according to priority factor based on attribute locality, i.e., the value of importance of an attribute with respect to sites of distributed database. At the time of designing the database, database designer will construct Attribute Locality Priority factor Table (ALPT) using the Enhanced CRUD (Create, Read, Update and Delete) matrix and cost functions. ECRUD matrix is a table constructed by placing predicates of attributes of a relation as the rows and applications of the

sites of a distributed database management system as columns.

Cost is treated as the effort of access and modification of a particular attribute of a relation by an application from a particular site. Using ECRUD matrix and cost functions [30], priority factor of an attribute of a relation is calculated. Since fragmentation is done at the initial stage, the actual frequencies of read, write, delete and update of a particular attribute from different applications of a site is not known. Hence it is assumed that f_C, f_R, f_D and f_U =1 and C=2, R=1, D=2 and U=3, where

f_C = frequency of create operation
f_R = frequency of read operation
f_D = frequency of delete operation
f_U = frequency of update operation
C = weight of create operation
R = weight of read operation
D = weight of delete operation
U = weight of update operation

Set of predicates is generated for the attribute with highest priority factor value in the Attribute Locality Priority factor Table. Then each relation is fragmented horizontally using the predicates. Fragmented segments are clustered. After this clustering process, re-fragmentation is done based on the next highest priority factor value in the ALPT within the cluster. Finally allocate the fragmented sub-relations to the sites within the cluster. The algorithm for this procedure is given in Fig. 4.

Input: Total number of sites
Site Information Table: LCA and GCA
Relation to be fragmented: R
Enhanced CRUD matrix: ECRUD
Output: Fragmented segments
Step 1: Construct ALPT[R] from ECRUD[R] based on cost functions
Step 2: Generate predicate set P for the attribute with highest priority factor value
Step 3: Fragment the relation using the predicate
Step 4: Cluster fragmented sub-relations.
Step 5: Re-fragment the fragments based on next highest priority factor valued attribute within the cluster.

Figure 4. Fragments and Sites Cluster Algorithm

To analyze the performance of the FlexiPeer architecture and implementation a distributed banking database system has been taken.

To demonstrate the performance of FlexiPeer, initially the number of sites is considered to be ten as shown in Fig. 5.

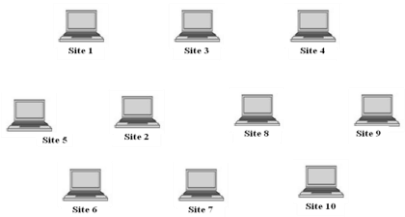

Figure 5. Initial number of sites

Information about each and every site will be given in Site Information Table (SIT) as shown in Table 1

.

TABLE 1. SITE INFORMATION TABLE (SIT)

Site Id	Locality	Region
1	L1	R1
2	L2	R2
3	L3	R1
4	L4	R3
5	L5	R2
6	L6	R1
7	L7	R3
8	L8	R4
9	L9	R4
10	L10	R1

Based on the initial requisites of FlexiPeer architecture, the sites are clustered as follows, there are four unique regions given in the SIT, hence, the FlexiPeer is framed with four clusters and the sites are categorized as four groups based on the respective regions of the site. The clustered formation is as shown in Fig. 6.

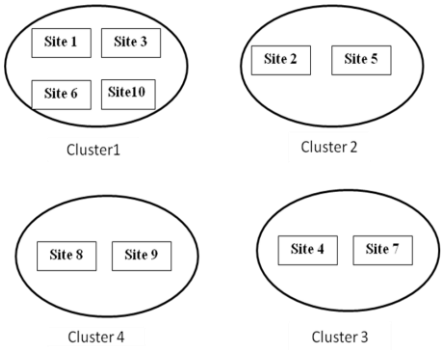

Figure 6. Clustered framework of sites

By taking the derived clusters, the FlexiPeer architecture is as shown in Fig. 7.

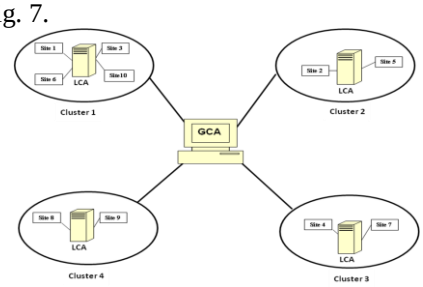

Figure 7. Flexi Peer architecture with four Clusters

The Accounts relation is taken for analyzing the Fragmentation and Allocation in FlexiPeer. The attributes and values in Accounts relation is shown in Table 2.

TABLE 2. ACCOUNTS RELATION

Ano	Category	Cid	Date	Balance	Region
1	A	C1	11/1/11	21000	R1
2	B	C2	21/1/11	13500	R2
3	B	C3	2/2/11	18000	R1
4	C	C4	8/2/11	22000	R3
5	D	C5	24/2/11	3200	R4
6	C	C6	15/3/11	52000	R1
7	E	C7	18/3/11	38000	R2
8	D	C8	28/3/11	11500	R1
9	A	C9	4/4/11	16800	R3
10	A	C10	9/4/11	78000	R1
11	B	C11	11/4/11	23000	R4
12	B	C12	18/4/11	11800	R2

ECRUD matrix should be constructed for the Accounts relation during the requirement analysis phase. From this matrix ALP values will be calculated using the cost functions [30]. For example a sample ALPT for Accounts relation in shown in Table 3.

TABLE.3. PRIORITY FACTOR VALUES OF ACCOUNTS RELATION

Name of Attributes	Priority factor Value
ANO	10
CATEGORY	25
CID	11
DATE	14
BALANCE	18
REGION	58

The highest priority factor valued attribute will be considered as an important attribute for fragmentation. According to that predicate set will be generated. For instance, our ALPT shows that Region has the highest priority factor value. So the predicate set will be as follows:
P={Region=R1; Region=R2; Region=R3; Region=R4}
Based on these predicate sets, relation will be fragmented. So we will get the fragments as shown in Table 4.

TABLE 4. SUB-RELATION BASED ON PREDICATE 'REGIONS'

ANO	CATEGORY	CID	DATE	BALANCE	REGION
1	A	C1	11/1/11	21000	R1
3	B	C3	2/2/11	18000	R1
6	C	C6	15/3/11	52000	R1
8	D	C8	28/3/11	11500	R1
10	A	C10	9/4/11	78000	R1
2	B	C2	21/1/11	13500	R2
7	E	C7	18/3/11	38000	R2
12	B	C12	18/4/11	11800	R2
4	C	C4	8/2/11	22000	R3
9	A	C9	4/4/11	16800	R3
5	D	C5	24/2/11	3200	R4
11	B	C11	11/4/11	23000	R4

After clustering, re-fragmentation is done on the fragments based on the next highest priority factor value in the ALPT within the cluster. The re-fragmented sub-relations are then allocated to the sites within the cluster as shown in Table 5.

TABLE 5. RE-FRAGMENTED SUB-RELATIONS ALLOCATED TO SITES IN CLUSTERS

A	B	C	D	E	F	G	H
1	A	C1	11/1/11	21000	R1	1	1
10	A	C10	9/4/11	78000	R1	1	1
3	B	C3	2/2/11	18000	R1	3	1
6	C	C6	15/3/11	52000	R1	6	1
8	D	C8	28/3/11	11500	R1	10	1
2	B	C2	21/1/11	13500	R2	2	2
12	B	C12	18/4/11	11800	R2	2	2
7	E	C7	18/3/11	38000	R2	5	2
4	C	C4	8/2/11	22000	R3	4	3
9	A	C9	4/4/11	16800	R3	7	3
5	D	C5	24/2/11	3200	R4	8	4
11	B	C11	11/4/11	23000	R4	9	4

Description of Attributes of Table 5 is as follows:

A – ACCOUNT NO	B - CATEGORY
C- CUSTOMER ID	D - DATE
E - BALANCE	F - REGION
G - SITE	H – CLUSTER

If another site is added to any of the clusters, next highest priority factor valued attribute will be taken for further fragmentation.

V. PERFORMANCE EVALUATION OF FlexiPeer

The performance of FlexiPeer Distributed Database Architecture is studied in a simulated environment. The simulation encompassed a tool written in Java® and having the data in Oracle® 10g. This simulation is intended to evaluate the performance of FlexiPeer in execution of transactions to reach out the data in an appropriate site and the time taken to notice an error when the data not found or an invalid query. The execution time and error indication time is measured in milliseconds. This evaluation includes 15 cases for analyzing the transaction execution and 10 cases for measuring time taken to indicate error.
The evaluation results as shown in below:
Table 6 describes the evaluation results of time taken to execute a transaction in both Chord and FlexiPeer architectures.

TABLE 6. TIME TAKEN TO EXECUTE A TRANSACTION IN Chord AND FlexiPeer

A1	A2	A3	A4	A5
1	2	6	432	276
2	1	7	528	277
3	4	4	162	180
4	5	6	281	275
5	6	3	452	274
6	3	3	160	178
7	4	10	540	278
8	2	9	560	277
9	8	8	159	179
10	7	9	362	276
11	10	1	285	277
12	8	7	575	275
13	2	5	417	276
14	3	7	431	275
15	1	10	561	277

Description of Attributes of Table 6 is as follows:
A1 - Transaction ID A2 - Queried in (Site ID)
A3 - Data found in (Site ID)
A4 - Execution time in Chord (ms)
A5 - Execution time in FlexiPeer (ms)

Fig. 8 illustrates the evaluation results for reach out an appropriate site for each transaction in both Chord and FlexiPeer architectures

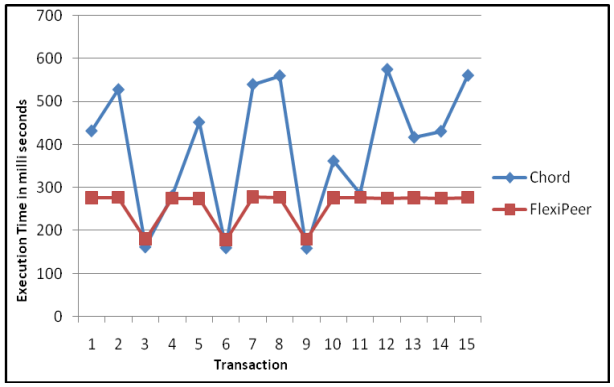

Figure 8. Graph illustrates the time taken for executing Transaction in Chord and FlexiPeer

Table 7 depicts the time taken for indicating the errors against 'no data found' / 'invalid' queries in Chord and FlexiPeer architectures.

TABLE 7. TIME TAKEN TO NOTICE ERROR FOR NO DATA FOUND / INVALID QUERY IN Chord AND FlexiPeer

B1	B2	B3	B4	B5	B6
1	2	10	2	522	172
2	4	10	4	472	171
3	5	10	5	436	174
4	6	10	6	417	173
5	10	10	10	157	171
6	2	10	2	487	172
7	8	10	8	434	172
8	7	10	7	461	171
9	9	10	9	288	174
10	1	10	1	561	173

Description of Attributes of Table 7 is as follows
B1 - Transaction ID B2 - Queried in (Site ID)
B3 - Error displayed in Chord (Site ID)
B4 - Error displayed in FlexiPeer (Site ID)
B5 - Execution time in Chord (millisecond)
B6 - Execution time in FlexiPeer (millisecond)

Fig. 9 correlates the time taken to indicate the errors for 'No data found' / 'invalid' queries in Chord and FlexiPeer architectures

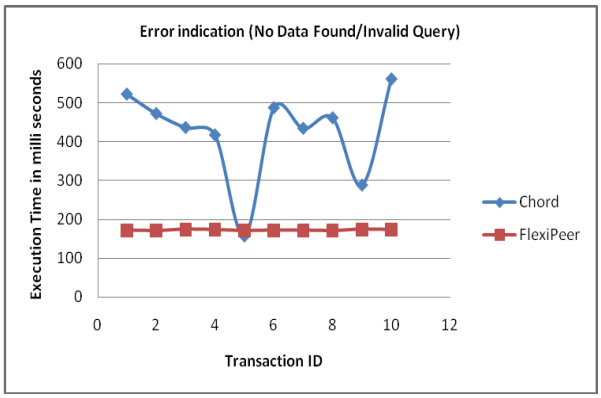

Figure.9. Graph illustrates the time taken for executing Transaction in Chord and FlexiPeer

VI. CONCLUSION

This paper addressed the design requirements for a Peer-to-Peer distributed database. This paper also framed an architecture named FlexiPeer. The clustering of sites is done based on the geographical regions of the sites. The clusters are managed by LCA and the overall architecture managed by GCA. Both LCA and GCA are equipped with functions to facilitate the data processing. With this FlexiPeer, the data are stored only in sites and the information about sites in a cluster is stored in respective LCA and the information about all clusters is stored in GCA. The relation will be fragmented based on the highest priority factor and those fragments are clustered along with the sites based on common predicate. Within a particular cluster, once again the sub-relation is re-fragmented to allocate data to the sites within the particular cluster based on the next highest priority factor. Finding appropriate data and respective site will be taken care by LCA and GCA. Hence the sites can effectively store and produce results for queries instead of wasting its processing capacity by listening to all queries though the required data are not available in that particular site. The query processing operations are simulated and studied with the results produced with Chord architecture. In future, the query

processing and concurrency control mechanisms can be studied in FlexiPeer environment.

REFERENCES

[1] M. T. Ozsu and P. Valduriez, *Principles of Distributed Database Systems*, (2nd ed., New Jersey: Prentice-Hall, 1999).
[2] David R. Karger and Matthias Ruhl, Finding Nearest Neighbors in Growth-restricted Metrics. *ACM Symposium on Theory of Computing (STOC '02)*, Montréal, May 2002
[3] Chord website: http://pdos.lcs.mit.edu/chord
[4] Freenet website: http://freenetproject.org
[5] Shamkant Navathe, A Mixed Fragmentation Methodology for Initial Distributed Database Design, 1995.
[6] Kamalakar Karlapalem and Qing Li, A Framework for Class Partitioning in Object-Oriented Databases, *Distributed and Parallel Databases, Vol. 8, No. 3*, 2000.
[7] Seung-Jin Lim and Yiu-Kai Ng, A Formal Approach for Horizontal Fragmentation in Distributed Deductive Database Design, *Proceedings of the 7th International Conference on Database and Expert Systems Applications*, 1996.
[8] Seung-Jin Lim and Yiu-Kai Ng, Vertical Fragmentation and Allocation in Distributed Deductive Database Systems, *Information Systems, Vol. 22, No. 1, pp. 1-24*, 1997.
[9] Zehai Zhou and Olivia R.Liu Sheng, Vertical Data Fragmentation and Fragment Allocation in Distributed Database Systems, *Proceedings of PACIS*, 1997.
[10] Ladjel Bellatreche, Kamalakar Karlapalem and Ana Simonet, Algorithms and Support for Horizontal Class Partitioning in Object-Oriented Databases, *Distributed and Parallel Databases, Vol. 8, No. 2, pp. 155-179*, 2000.
[11] Yin-Fu Huang and Jyh-Her Chen, Fragment Allocation in Distributed Database Design, *Journal Of Information Science And Engineering 17, pp. 491-506*, 2001.
[12] Ishfaq Ahmad, Yu-Kwong Kwok and Siu-Kai So, Evolutionary Algorithms for Allocating Data in Distributed Database Systems, *Distributed and Parallel Databases, Vol. 11*, 2002.
[13] Jun Du, Ken Barker and Reda Alhajj, Attraction- A Global Affinity Measure for Database Vertical Partitioning, *Proceedings of ICWI*, 2003.
[14] C.I.Ezeife and Pinakpani Dey, Incremental Horizontal Fragmentation of Database Class Objects, *Proceedings of ICEI*, 2003.
[15] A.S.Darabant, A.Campan and H.Todoran, Incremental Horizontal Fragmentation: A New Approach in the Design of Distributed Object Oriented Databases, *International Conference on Computers, Communications and Control, ICCCC 2006*, 2006, pp 170-174,.
[16] Horea Grebla, Grigor Moldovan, Sergiu Adrian Darabant and Alina Campan, Data Allocation in Distributed Database Systems Performed by Mobile Intelligent Agents, *Proceedings of the International Conference on Theory and Applications of Mathematics and Informatics*, 2004.
[17] Sergiu Adrian Darabant, Alina Campan, Grigor Moldovan and Horea Grebla, AI Techniques: A New Approach in Horizontal Fragmentation of Classes with Complex Attributes and Methods in Object Oriented Databases, *Proceedings of the International Conference on Theory and Applications of Mathematics and Informatics*, 2004.
[18] Sergiu Adrian Darabant, Alina Campan and Octavian Cret, Hierarchical Clustering in Object Oriented Data Models with Complex Class Relationships, *Proceedings of the Eighth IEEE International Conference on Intelligent Engineering Systems INES2004*, 2004, pp. 307-312,.
[19] Sergiu Adrian Darabant and Alina Campan, AI Clustering Techniques: A New Approach to Object Oriented Database Fragmentation, *Proceedings of the 8th IEEE International Conference on Intelligent Engineering Systems*, 2004, pp. 73-78.
[20] Sergiu Adrian Darabant and Alina Campan, Semi-supervised Learning Techniques: k-means Clustering in OODB Fragmentation, *IEEE International Conference on Computational Cybernetics ICCC 2004*, 2004, pp 333-338.
[21] Sergiu Adrian Darabant, Horea Todoran, Octavian Cret and George Chis, The Similarity Measures and their Impact on OODB Fragmentation Using Hierarchical Clustering Algorithms, *WSEAS Transactions on Computers, Vol. 5*, 2006.
[22] Ismail O. Hababeh, A Method for Fragment Allocation Design in the Distributed Database Systems, *The Sixth Annual U.A.E. University Research Conference*, 2005.
[23] Alina Campan, Sergiu Adrian Darabant and Gabriela Serban, Clustering Techniques for Adaptive Horizontal Fragmentation in Object Oriented Databases, *Proceedings of the International Conference on Theory and Applications of Mathematics and Informatics ICTAMI*, 2005, pp 263-274.
[24] Hassan I. Abdalla and F. Marir, Vertical Partitioning Impact on Performance and Manageability of Distributed Database Systems (A Comparative study of some vertical partitioning algorithms), *18th National Computer Conference, Saudi Computer Society*, 2006.
[25] Hui Ma, Klaus-Dieter Schewe and Markus Kirchberg, A Heuristic Approach to Vertical Fragmentation Incorporating Query Information, *Proceedings of the 2007 conference on Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference DB&IS'2006 IOS Press Amsterdam, The Netherlands, The Netherlands* , 2007.
[26] Hui Ma and Markus Kirchberg, Cost-Based Fragmentation for Distributed Complex Value Databases, *Proceedings of the 26th International Conference on Conceptual modeling*, 2007.
[27] Eltayeb Salih Abuelyaman, An Optimized Scheme for Vertical Partitioning of a Distributed Database, *IJCSNS*

International Journal of Computer Science and Network Security, Vol.8, No.1, 2008.

[28] Rajan John and Dr. V. Saravanan, Vertical Partitioning in Object Oriented Databases Using Intelligent Agents, *IJCSNS International Journal of Computer Science and Network Security, Vol.8, No.10*, 2008.

[29] Arjan Singh and K.S. Kahlon, Non-replicated Dynamic Data Allocation in Distributed Database Systems, *IJCSNS International Journal of Computer Science and Network Security, Vol.9, No.9*, 2009.

[30] Shahidul Islam Khan and Dr. A. S. M. Latiful Hoque, A New Technique for Database Fragmentation in Distributed Systems, *International Journal of Computer Application, Vol. 5, No.9*, 2010.

[31] Nilarun Mukherjee, Synthesis of Non-Replicated Dynamic Fragment Allocation Algorithm in Distributed Database Systems, *Proceedings of International Conference on Advances in Computer Science*, 2010.

[32] Aleksandar Dimovski, Gorgan Velinov and Dragan Sahpaski, Horizontal Partitioning by Predicate Abstraction and its Application to Data Warehouse Design, *Proceedings of the 14th east European conference on Advances in databases and information systems*, 2010.

[33] Frank Dabek, Emma Brunskill, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica, and Hari Balakrishnan, Building Peer-to-Peer Systems With Chord, a Distributed Lookup Service, *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII),* May 2001.

[34] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, *ACM SIGCOMM 2001, San Deigo, CA,* August 2001, pp. 149-160.

[35] Anjali Gupta, Barbara Liskov, and Rodrigo Rodrigues, Efficient routing for peer-to-peer overlays, *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1 San Francisco, California*, 2004

[36] David Liben-Nowell, Hari Balakrishnan, David Karger, Observations on the Dynamic Evolution of Peer-to-Peer Networks, *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA*, March 2002.

[37] David R. Karger and Matthias Ruhl, Finding Nearest Neighbors in Growth-restricted Metrics . *ACM Symposium on Theory of Computing (STOC '02), Montréal*, May 2002

[38] Ian Clarke, Scott G.Miller, Theodore W.Hong, Oskar Sandberg and Brandon Wiley, Protecting Free Expression Online with Freenet, *IEEE Internet Computing, Jan.2002*

[39] *Xiuguo Bao, Binxing Fang, Mingzhen Hu and Binbin Xu, Heterogeneous Search in Unstructured Peer-to-Peer Networks, IEEE DISTRIBUTED SYSTEMS ONLINE Published by the IEEE Computer Society Vol. 6, No. 2*; 2005