# A Business Model for Hybrid Shared-Nothing and Shared-data Storage and replication Scheme for Large-scale Data Processing

**Anisaara Nadaph, Vikas Maral**

***Abstract— This project is for hybrid storage architecture, to make use of both shared-nothing and shared-disk architectures. The user can either upload file or can just synchronize the original copy from the master computer synchronized before. All the changes made on the original file will be reflected on the file stored on server. All files stored on the cloud server are broken into packets of some definite size and these packets will be distributed on various hard disks this data are replicated using the RAID-1 concept, which are merged as a whole again whenever the user makes an access to the file on the on-line copy or makes changes to the original copy which is synchronized with the application. The packets are encrypted using a block of ECB encrypted cipher text; all the blocks are dependent on all the previous blocks.***

***Index Terms—Cloud Computing, Replication, hybrid architecture, DES, CBC..***

## I. INTRODUCTION

### A. What is Cloud Computing?

Cloud computing[2] is an emerging computing paradigm in which resources of the computing infrastructure are provided as services of the internet. Cloud computing allows consumers and businesses to use applications without installation and access their personal files at any computer with internet access. As promising as it is, this paradigm also brings forth many new challenges for data security and access control when users outsource sensitive data for sharing on cloud servers which is not within the same trusted domain as data owners. To keep sensitive user data confidential against untrusted servers, cryptographic methods are used by disclosing data decryption keys only to authorized users.

### B. Cloud Computing Architecture

Cloud computing architecture is divided into two sections: the front end and the back end. They connect to each other through a network usually called the Internet. The front end includes the client's computer (or computer network) and the application required to access the cloud computing system. On the back end of the system are the various computers, servers and the data storage systems that create the cloud of the computing services. A central server administers the system, monitoring traffic and client demands to ensure everything runs smoothly. It follows a set of rules called protocols and uses a special kind of software called middleware. Middleware allows networked computers to communicate with each other. A.Types of Clouds:

There are four main type of cloud:

**Public cloud:** In Public cloud the computing infrastructure is hosted by the cloud vendor at the vendor's premises. The customer has no visibility and control over where the computing infrastructure is hosted. The computing infrastructure is shared between any organizations.

**Private cloud:** The computing infrastructure is dedicated to a particular organization and not shared with other organizations.

**Hybrid cloud[2]** : Organizations may host critical applications on private clouds and applications with relatively less security concerns on the public cloud. The usage of both private and public clouds together is called hybrid cloud.

**Community cloud[2]** : It involves sharing of computing infrastructure in between organizations of the same community. For example all Government organizations within the state of Maharashtra may share computing infrastructure on the cloud to manage data related to citizens residing in Maharashtra.

*The Contribution Of this Project will gives us 3 results*

i. Combining shared-nothing and shared-data in parallel database systems, would greatly add benefit for many emerging data-intensive applications and will give boost to the development of parallel file systems.

ii. Here we propose hybrid storage architecture for parallel databases to carry the combination of both two architectures(shred-nothing and shared-disk architecture) the design of the proposed storage scheme, will improve data access modes, data organization, and query processing methods..

iii. Replication of data will ensure data reliability of data.

## II. RELATED WORK

### A. Architecture

Shared-nothing [1],[4] and shared-disk[5] are the two widely-used storage architectures in parallel databases. Both two architectures have their own positive and negative features, but neither of them has the edge on the other in all aspects. Figure 1(a) shows a sketch of the shared-nothing storage architectures.

In a shared-nothing system, the data set is usually partitioned[1] into

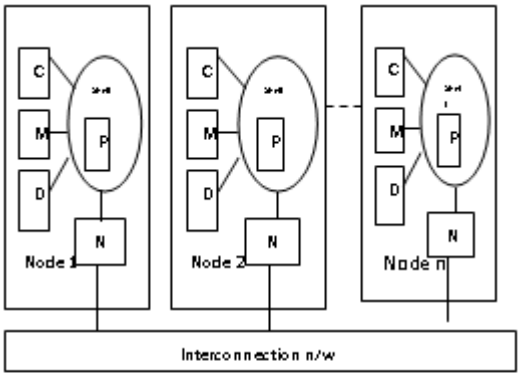
Manuscript Received on October 2012

**Ms. Anisaara Nadaph,** Computer Engg., K.J College Of Engg. And Management Research, Pune, Pune, India

**Prof. Vikas Maral,** Computer Engg., K.J College Of Engg. And Management Research, Pune, Pune, India

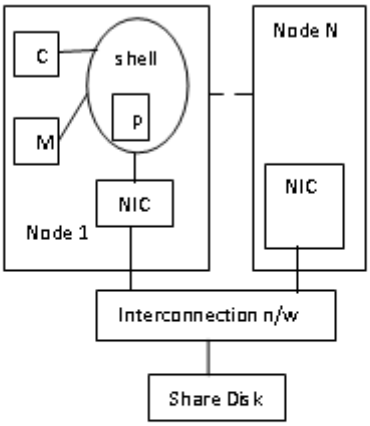
several subsets and each node keeps one subset in its native disks. Generally, the shared-nothing systems provide a high degree of parallelism for both I/O and computing. Nevertheless, the shared-nothing systems also have multiple nodes transaction, data shipping, and data skew issues. In a shared-disk system, data is stored in a large centralized storage, which is accessible by all database nodes.

**Advantage** of shared-nothing architecture is that since every node has direct access to all disks, there is no need of data partitioning according to the number of nodes, which eliminates the data skew problem. Figure 1(b) shows a sketch of the shared-disk storage architectures. **Disadvantage** of shared-disk systems, however, are low I/O bandwidth and poor scalability.




C –Cache
D-Disk
M - Memory
NIC - Network Interface circuitary
P – Processor

Fig.1 (a) Shared-Nothing Architecture




C - Cache
M - Memory
NIC - Network Interface circuitary
P – Processor

Fig1(b) Shared-Disk Architecture

### B. Parallel Database

Parallel database [1],[11] machine architectures have evolved from the use of exotic hardware to a software parallel data flow architecture based on conventional shared-nothing hardware. These new designs provide impressive speedup and scale up when processing relational database queries

*Early stage of parallel database:* The bandwidth and latency of network is worse than accessing data to local disks, hence it is natural to avoid accessing data from a remote disk through network. For that reason, the shared-nothing and the shared-disk architectures were presented separately. However, network transmission bandwidth grows rapidly during the past years.

*Today's Scenario :* reading data from remote disks is no longer limited by network in a cluster environment. The tradeoffs between shared-disk and shared-nothing storage architectures need to be reevaluated. On the other hand, in the domain of data-intensive scientific computing, parallel file systems, such as Lustre, PVFS2, and GPFS, are widely used for high I/O performance. Compared with parallel databases, parallel file systems are shared-nothing structured (as shown in Figure 1), but they also unite all disks on multiple nodes and provide a single namespace.

Based on the design of parallel file systems, we propose hybrid storage architecture for large-scale parallel databases, by integrating parallel database with parallel file system techniques together. It adopts shared-nothing for the upper layer database instances, but also provides data sharing through a lower-layer parallel file system. Therefore, the proposed hybrid system can work both as a shared-nothing system if each database node only accesses data on its own disks, and as a shared-disk system if the database nodes access data from the global namespace of the parallel file system.

### C. Partitioned Methods

In shared-nothing systems, data is partitioned and distributed across all the database nodes. DeWitt and Gray introduced three well known partitioning methods:

**a. Round-robin partitioning:**

In round-robin partitioning, There is no partitioning criteria. Round-robin-partitioned tables have no partition key. To each partition rows are assigned in a round-robin manner so that each partition contains a more or less equal number of rows and load balancing is achieved. Because there is no partition key, rows are distributed randomly across all partitions.

In addition, round-robin partitioning offers:

i. Multiple insertion points for future inserts

ii. A way to enhance performance using parallelism

iii. A way to perform administrative tasks, such as updating statistics and truncating data on individual partitions

**b. Range partitioning:**

Rows in a range-partitioned table or index are distributed among

partitions according to values in the partitioning key columns. The partitioning column values of each row are compared with a set of upper and lower bounds to determine the partition to which the row belongs.

i. Every partition has an inclusive upper bound, which is specified by the **values** <= clause when the partition is created.

ii. Every partition except the first has a non-inclusive lower bound, which is specified implicitly by the **values** <= clause on the next-lower partition.

Range partitioning is particularly useful for high-performance applications in both OLTP and decision-support environments. Select ranges carefully so that rows are assigned equally to all partitions—knowledge of the data distribution of the partition key columns is crucial to balancing the load evenly among the partitions.

Range partitions are ordered; that is, each succeeding partition must have a higher bound than the previous partition.

**c. Hash partitioning:**

With hash partitioning, use of hash function to specify the partition assignment for each row. You select the partitioning key columns, but Adaptive Server chooses the hash function that controls the partition assignment.

Hash partitioning is a good choice for:

i. Large tables with many partitions—particularly in decision-support environments

ii. Efficient equality searches on hash key columns

iii. Data with no particular order, for example, alphanumeric product code keys

If you choose an appropriate partition key, hash partitioning distributes data evenly across all partitions. However, if you choose an inappropriate key—for example, a key that has the same value for many rows—the result may be skewed data, with an unbalanced distribution of rows among the partitions.

Besides, there are some combined strategies on multiple columns. When a query comes, it will be routed to the appropriate database nodes according to the data partitioning scheme. Complex queries, e.g. 'multi-join' or 'nested', are often executed as recursive computing and re-distributing phases on multiple database nodes. It is unlikely to find a universal partitioning strategy suitable for all query types. As dataset and query pattern vary with the time, a good partitioning scheme in the past may become sub-optimal, resulting in load imbalance and performance degradation. Dynamic or adaptive algorithms were introduced to deal with the load balance problems. Nowadays, there are numerous database clusters using the shared-nothing architecture, such as IBM DB2 UDB, Mysql Cluster, and Teradata products, etc.

In shared-disk systems, data is stored in shared disks, and every node has full access to the entire data. There is no need to partition data across multiple database nodes, which eliminates the data skew problems. During query processing, the collaboration of different nodes relies on inter-nodal message and shared data access. Data sharing between memories on different nodes can deliver a superior performance in shared-disk system. Cache fusion is a memory-sharing technique applied by Oracle RAC version, which can largely improve the query performance, because the high transmission speed and low latency of network make it much faster than data write back to shared disk and re-read courses. Currently, the shared-disk systems usually adopt storage area network (SAN) to provide high I/O performance, and it is costly due to dedicated hardware.

### *D. Parallel and Distributed File Systems*

IParallel file systems [5], such as Lustre, PVFS2, and GPFS, are widely used in large-scale and data-intensive applications, to provide high I/O capabilities to high performance computing (HPC) clusters. Normally, parallel file systems provide high I/O performance by striping data files over multiple storage nodes, and accessing these data strips in parallel. I/O clients can access files by logic address as in a single namespace, without the knowledge of physical layout.

The transparency of data block placement is a convenient feature for users. Google file system (GFS)[8] and Hadoop distributed file system (HDFS)[9] are scalable distributed file systems for large distributed web search engine applications.

Both GFS and HDFS are designed with big file chunks (chunk size is 64MB) and MapReduce programming model. The write-once-read-many data access manner means no data modifications, in addition, MapReduce programming model are not designed for low latency. For those reasons, GFS and

HDFS are not suitable for general purpose parallel databases.

There are some research efforts in integrating database and parallel file system technologies. Some scientific computing systems employ databases for metadata management and parallel file systems for data storage respectively. Hive and HBase are built on HDFS and based on MapReduce model. They are designed for special purpose, and the query types and query processing manners are different from traditional databases. In addition, they are not suitable for low latency applications.

The proposed hybrid architecture is different from others work. It is designed for general parallel databases. It has a shared-nothing design in the hardware layer, and provides data sharing through the underlying parallel file system. It has the merits of both shared-nothing and shared-disk parallel database

## III. PROPOSED WORK

### *A. System Architecture*

In the scheme that we are adopting we are using

shared-nothing at the hardware layer, but parallel file system to unite all scattered disks to provide data sharing capability. Figure 2 illustrates the system architecture of the proposed hybrid storage scheme, which can leverage the advantages of both the shared-nothing hardware structure and the shared-data facility of parallel file systems. In the proposed hybrid database, each node runs a database instance, and serves as both a parallel file system I/O server and an I/O client. Therefore, all database instances can access data in the parallel file system, namely data can be shared among different nodes conveniently. In a word, it is shared-nothing in hardware layer, but it is also a shared-data system. Usually, in parallel file systems, data are striped across multiple I/O servers in a round-robin way, thus one file is

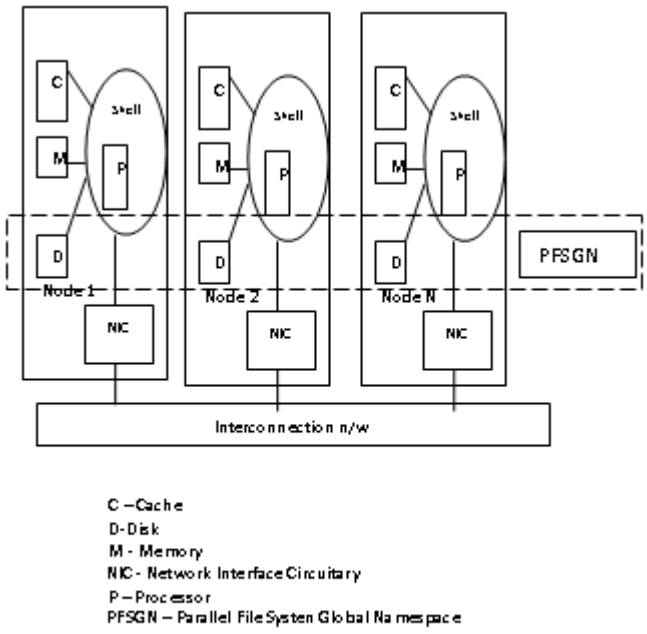



Fig.2 Hybrid Parallel database Storage Scheme

divided into several sub-files on these servers. In the proposed hybrid system, one file in parallel file system is called global file, and a sub-file on one node is called local file. Thus, a global file is mapped into a set of local files, as shown in Figure 3. We introduce two data access modes in the proposed hybrid systems: local mode and global mode.

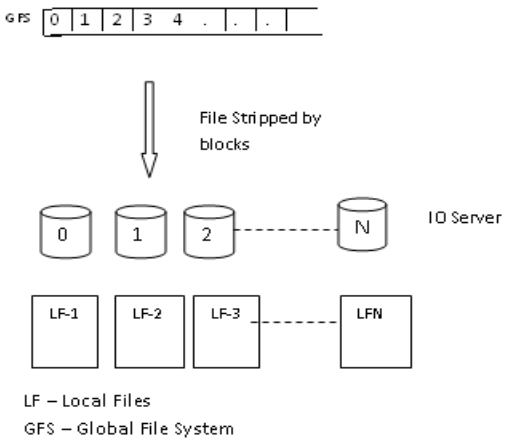



Fig. 3 Data Layout of parallel File System

• *Local Mode:* each database node accesses data via a local file descriptor on its native disks.

• *Global Mode:* each database node accesses data via a global file descriptor in the global namespace.

Data access pattern can be either local mode or global mode, by calling read/write functions on a local file descriptor or a global file descriptor respectively. Which data access mode to use for query execution is determined by system query optimizer, based on the costs of query execution of the two modes. With high-speed interconnection techniques, the proposed hybrid scheme can obtain the advantages of both shared-nothing and shared-disk systems.

*A. Data Organization*

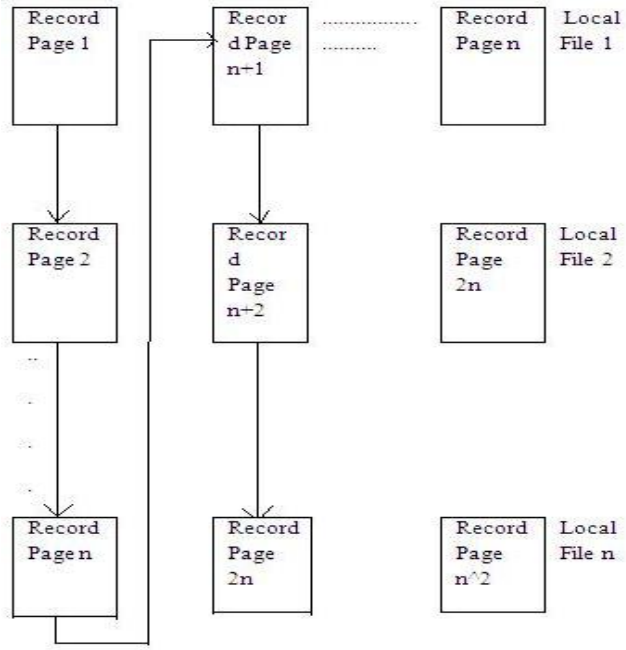



Fig. 4 Striped File by Record Pages in Hybrid System

In relational databases, data files are composed of record pages. We design the stripe size in parallel file systems the same as the record page size in the hybrid system, thus database tables are naturally striped across all database nodes.

Figure 4 shows the organization of the relational tables in the proposed hybrid system [6]. Each table can be stored as a global file, which is striped by record pages. Hence each local table is a subset of original table, which could be regarded as a sub-table for each database node. Especially, record assigned across pages is not recommended, because it would lead to complex data access behaviors like data shipping and distributed locks.

For that reason, we reserve a small area in each page, which is also useful for updating records with variable lengths. The percentage of reserved area in a record page is configurable, like that in current commercial databases. With the design of record page striping and reserved area, each node can be regarded as a standalone sub-database in local data access mode. The data dictionary is modified to support two data access modes in the proposed

scheme. For local data access, each node is a standalone sub-database, and it should keep all the metadata information in the local data dictionary. This design can eliminate the data dependency among different nodes. Furthermore, some additional information should be included on each node for global access. For example, file information in data dictionary on each node must contain a local file name and a global file name, as shown in Figure 4, to support for both two data access modes. This data organization is quite different from existing GFS and HDFS. In GFS and HDFS, storage nodes do not contain the global data set information; they have to turn to the dedicated metadata server for remote data access.

*B.Replication strategy*

Using the RAID level 1 (Mirroring) data are stored twice on two disk i.e. data disk and mirror disk. If a disk fails, the controller uses controller uses the data drive or the mirror drive for data recovery and continues operation.

Advantages of RAID1 are

a. Offers excellent read speed and write-speed that is comparable to that of a single disk.
b. In case a disk fails, data do not have to be rebuild, they just have to copy the data to the replacement disk
c. It is very simple technology

As we can see of this project it is only essential to maintain the backup copy and there is no need of parity check to be done hence RAID-1 is the feasible storage technique.

## IV. DATA ENCRYPTION STANDARD (DES)

The Data Encryption Standard (DES) [3] is the name of the Federal Information Processing Standard (FIPS) 46-3, which describes the data encryption algorithm (DEA). The DES has been extensively studied since its publication and is the most widely used symmetric algorithm in the world. The DES has a 64-bit block size key during execution. DES is a symmetric cryptosystem [7], specifically a 16-round Feistel Cipher. When used for communication, both sender and receiver must know the same secret key, which can be used to encrypt and decrypt the message, or to generate and verify a Message Authentication Code (MAC). The DES can also be used for Single – user encryption, such as to store files on a hard disk in encrypted form .The DES has a 64-bit block size and uses a 56 bit key during execution.

## V.IMPLEMETATION

i. User uploads a doc file and gives a password for the file to be uploaded.

ii. File is split into 16parts at server-side using zip4j library

Class ZipFile

Base class to handle zip files. Some of the operations supported in this class are:

Create Zip File

Add files to zip file

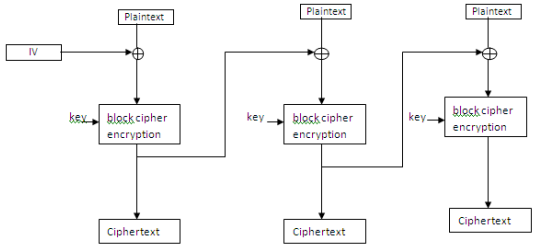
Add folder to zip file

Extract files from zip files

Remove files from zip file

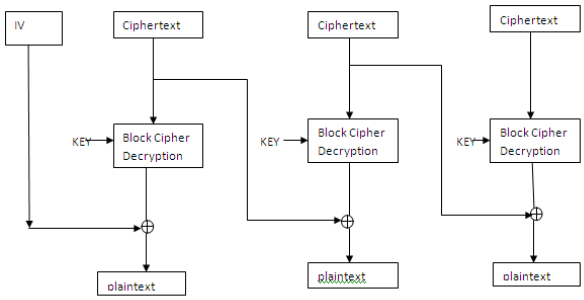
iii. Each part is encrypted using CBC (DES) encryption algorithm. The encryption of the 16 parts is as follows:

In Cipher Block Chaining mode [2] of operation of DES, each block of ECB encrypted ciphertext is XORed with the next plain text block to be encrypted, thus making all the blocks dependent on all the previous blocks .this means that in order to find the plaintext of a particular block, you need to know the ciphertext, the key and the cipher text for the previous block. The first block to be encrypted has no previous cipher text, so the plaintext is XORed with a 64-bit number called the initialization vector (referred as IV).So if data is transmitted over network or phone line and there is a transmission error, the error will be carried forward to all the subsequent blocks since each block is dependent upon the last .this mode of operation is more secure than ECB (electronic code book) because the extra XOR step adds one more layer to the encryption process.




IV- Initialization Vector

Fig. 5 Cipher Block Chaining (CBC) mode encryption




IV- Initialization Vector

Fig. 6 Cipher Block Chaining (CBC) mode decryption

Compositions of Encryption and Decryption :

Encryption E = eH1 o eH2 …… o eH16

Decryption D = dH16 o dH15 o ……o dH1

Header H is derived from the Password. Here we have 16 rotations. So we need 16 Leaders (H1 to H16) from Password.

H1 = First two bits of Password.

H2 = Second two bits of Password

H3 = Third two bits of Password and so on

**Steps:**

1. Get the Plaintext.
2. Get the Password.
3. Convert the Characters into binary form.

4. Derive the headers (H1 to H16) from the Password.

5. Apply the Formula to get the encrypted and decrypted message.

**Encryption:**

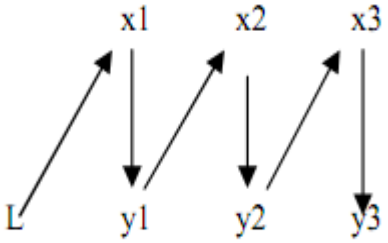



**Fig.7. Encryption using CBC**

**Decryption:**

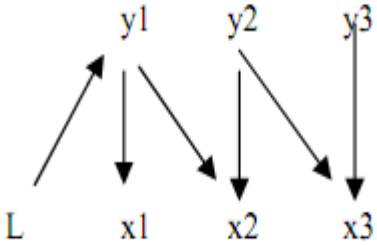



**Fig.8.Decryption using CBC**

iv. Each part is stored separately on different servers.

v.Hardware is shared-nothing architecture.

vi. A software layer makes the hardware as shared-everything architecture.

vii. While retrieving a file a query is fired on software layer which gathers all parts from the hardware layer.

viii. Each part has two bytes of password saved which while retrieving is checked again.

ix. A backup of each part is stored on different server using RAID-1 Level.

x.Incase a part of file is not retrieved or lost during the process then the backup server is used to retrieve the packet.

xi. The file is merged after checking the password, if the password is not matched then the file is not shown to the user.

## VI. CONCLUSION

As cloud is an emerging trend of the decade a more secure and easily accessible storage for enterprises and Individuals is in need. As off now data security has become the most important issue of Cloud Computing

The main contribution of this is the new view of data security solution with encryption, which is important and can be used as reference for designing the complete security solution.

This Study proposes a novel hybrid shared-nothing/shared-data scheme for large-scale and data-intensive applications, is to advantages of both shared-nothing and shared-disk architectures. We adopt a shared-nothing architecture as the hardware layer and leverage a parallel file system as the storage layer to combine the scattered disks on all database nodes. With the idea of the new hybrid architecture, we first introduce two data access modes: global and local data access modes, which enable the new system can work as both a shared- nothing system and a shared-disk system. Second, we present the methodology of organizing data files in the underlying parallel file system so that each node can run as a standalone sub-database.

## ACKNOWLEDGMENT

The Author is thankful to Prof. Vika Maral who has guided Author for the survey. Also to Prof. Das and Prof. Mehtre from "K.J. College of Engg. And Management Research, Pune" for their help and suggestions.

## REFERENCES

1. Huaiming Song, Xian-He Sun, Yong Chen "A Hybrid Shared-nothing/Shared-data Storage Scheme for Large-scale Data Processing" Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on 23-26 May 2011, 616 – 617
2. Neha Jain and Gurpreet Kaur "Implementing DES Algorithm in Cloud for Data Security" VSRD-IJCSIT, Vol. 2 (4), 2012, 316-321
3. [3]. Eman M. Mohamed "Modern Encryption Techniques for Cloud Computing Randomness and Performance" e Informatics and Systems (INFOS), 2012 8th International Conference on 14-16 May 2012, CC-1 - CC-6
4. Michale G .Norman "Much Ado Shared-Nothing" ACM SIGMOD Record Homepage archive Volume 25 Issue 3, Sept. 1996 , 16 – 21
5. Matthieu Exbrayat " A Parallel Extension for Existing Relational Database Management Systems" Information Technology, 1997. BIWIT '97., Proceedings of the Third Basque International Workshop on Digital Object Identifier: 10.1109/BIWIT.1997.614053 , 1997 , 75-81
6. David J. DeWitt ""Data placement in shared-nothing parallel database systems". The VLDB Journal —aG. O. Young, "Synthetic structure of industrial plastics (Book style with paper title and editor)," in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.
7. W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.
8. H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.
9. B. Smith, "An approach to graphs of linear forms (Unpublished work style)," unpublished.

## AUTHOR PROFILE

**Anisaara Nadaph :** Student of K.J College of Engg. And Management Research, Pune.

a. M.E (Computer Engg. Pursuing) from Pune University
b. B.E(Computer) from Pune University
c. Teaching Experience – 7 years
d. Seminar Guide Prof. Vikas Maral

**Prof. Vikas Maral :** Lecturer in K.J College of Engg. And Management Research, Pune.