International Journal of Innovative Technology & Adaptive Management (IJITAM)
ISSN: 2347-3622, Volume-2, Issue-8, May, 2015

Distributed, Parallel, P2P and Grid-Based Databases

RashmiKumari
Computer Science and Engineering, B. tech 4th Year
C.V. Raman College of Engineering
reshu.rashmikumari@gmail.com

Abstract.
Advances in the computer and communication technologies have led to so many computer systems like Distributed, Parallel, peer-to-peer and grid-based database systems. These new technologies are very important for data shearing over network. The purpose of this paper is to present distributed databases and Parallel Database, Peer-to-Peer databases and Grid-based databases and comparison among them.

Keywords: Distributed database, Parallel database, peer-to-peer and grid-based database.

1. Introduction

Conceptually, a database [1]consists of a collection of logical data items where the granularity of a logical data item may be a file, a record, or an arbitrary collection of data fields. A database management system (DBMS) is designed to:

1. Maintain relationships between logical data items (one to one, one to many or many to many).
2. Isolate programs from data format so that when data changes, the programs do not need to be changed.
3. Enforce security to allow access to authorized users only.
4. Provide integrity of the data in terms of data consistency (the data must correctly reflect the state of a system), currency (the database must contain recent information), and concurrency (the data must be simultaneously accessible by different users).

A typical database management system [1] uses a database dictionary/directory to store the data relationships, data formats and security restrictions; database logs to record the activities of transactions; and lock tables to allow synchronous concurrent access to the databases by several users. Facilities of a DBMS can be categorizedin terms of data definition, data manipulation and operational facilities.

A Distributed [2][3] Database Management System (DDBMS) allows user to access and manipulate data from several databases that are physically distributed to several sites. A DDBMS differs from a DBMS, henceforth referred to as a local DBMS (LDBMS), because it is responsible for providing transparent and simultaneous access to several databases that b are located on, perhaps, dissimilar computer systems where a local DBNS manages a single site database. The DDBMS would allow an end-user to:

1. Create and store a new part anywhere in the network
2. Access a part- no without knowing where the part-no is physically located
3. Delete a part from one database withput having to worry about how and when all the duplicated parts will be updated in other databases.
4. Access a part from an alternative computer when, say, the nearest computer isa not available.

In the future, a common problem for DDBMS will be to provide transparent access to the commercial, engineering and expert system databases that may be stored anywhere in the network.

A parallel database [3] system is one thatseeks to get better performance throughparallel implementation of variousoperations such as loading data, buildingindexes, and evaluating queries.

It is a softwaresystem wheremultipleprocessors ormachines areused to executeand run queriesin parallel.

Peer-to-peer (P2P) computing [5] has been hailed as a promising technology that willreconstruct the architecture of distributed computing (or even that of the Internet). This is because it can harness various resources (including computation, storage and bandwidth) at the edge of the Internet, with lower cost of ownership, and at the same time enjoys many desirable features (e.g., scalability, autonomy, etc.). Since mid-2000, P2P computing technology has spurred increasing interests in both industrialand academic communities. As such, there are increasingly more applications beingdeveloped based on this paradigm. For example, digital content sharing (e.g.,Naspter, Gnutella, and Shareaza), scientific computation (e.g.,BOINC and Folding@home), collaborative groupware (e.g., Groove),instant messages (e.g., ICQ) and so on. Furthermore, many research topics relatedto P2P computing have also been studied extensively—overlay network, routingstrategies, resource location and allocation, query processing, replication, andcaching. However, there has not been much effort to study the architecture of P2Psystems.

The popularity of the Internet as well as the availability of powerful computers and high-speed network technologies as low-cost commodity components is changing the way we use computers today. These technology opportunities have led to the possibility of using distributed computers as a single, unified computing resource, leading to what is popularly known as Grid computing.

The term Grid is chosen as an analogy to a power Grid that provides consistent, pervasive, dependable, transparent access to electricity irrespective of its source. A detailed analysis of this analogy can be found in. This new approach to network computing is known by several names, such asmetacomputing, scalable computing, global computing, and Internetcomputing and more recently peer-to-peer (P2P) computing.

2. Distributed Database System (DDBs)

A distributed database (DDB) [2] is a collection of multiple, logically interrelated databases distributed over a computer network. A distributed database management system (distributed DBMS) is the software system that authorities the management of the distributed database and makes the distribution transparent to the users. The term distributed database system (DDBS) is usually used to refer to the mixture of DDB and the distributed DBMS. Distributed DBMSs are like to distributed file systems in that both assist access to distributed data.

2.1. Architecture of DDBs

2.1.1. Client-Server

A Client-Server system [2][6] has one or more client processes and one or more server processes, and a client process can send a query to any one server process. Clients are responsible for user-interface issues, and servers manage data and execute transactions. Thus, a client process could run on a personal computer and send queries to a server running on a mainframe.

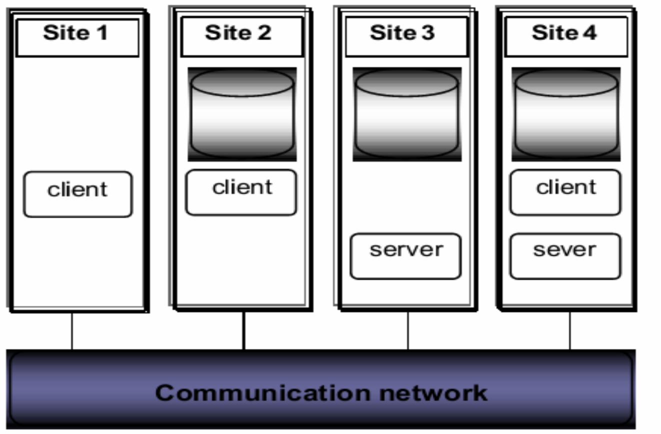

Figure 1: Client-Server architecture

2.1.2. Collaborating Server

In Collaborating Server [2][6] system, we can have collection of database servers, each capable of running transactions beside local data, which cooperatively execute transactions spanning multiple servers. When a server receives a query that requires access to data at other servers, it generates appropriate sub queries to be executed by other servers and puts the results together to compute answers to the original query.

2.1.3. Middleware

Middleware system [6] is as special server, a layer of software that coordinates the execution of queries and transactions across one or more self-governing database servers. The Middleware architecture is designed to allow a single query to span multiple servers, without requiring all database servers to be capable of managing such multi-site execution\

strategies. It is especially attractive when trying to integrate several legacy systems, whose basic capabilities cannot be extended. We need just one database server that is capable of managing queries and transactions spanning multiple servers; the remaining servers only need to handle local queries and transactions.

2.2. Types of Distributed Databases

Homogeneous Distributed Database[2][6] is where the data stored across multiple sites is managed by same DBMS software at all the sites.

Heterogeneous Distributed Database is where multiple sites which may be autonomous are under the control of different DBMS software.

2.3. Properties of Distributed Database

Distributed Data Independence: -
The user should be able to access the database without having the need to know the location of the data.
Distributed Transaction Atomicity: -
The concept of atomicity should be distributed for the operation taking place at the distributed sites.

3. Parallel Database System

A parallel database system [3] is one that seeks to get better performance through parallel implementation of various operations such as loading data, building indexes, and evaluating queries.

3.1. Architecture of Parallel Database

Parallel Database is implemented by using three types of Architecture[3][7] and is:

3.1.1. Shared-memory system

In this Architecture multiple CPUs are attached to an interconnection network and can access a common region of main memory.

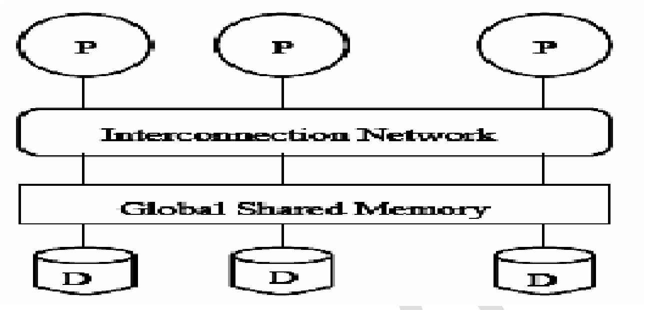

Figure 2: Shear memory system

3.1.2. Shared-Disk system

In this architecture each CPU has a private memory and direct access to all disks through an interconnection network.

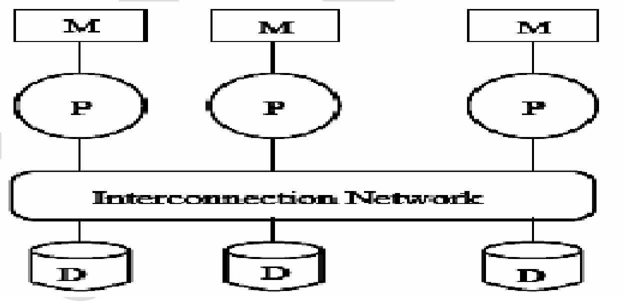

Figure 3: Shear disk system

3.1.3. Shared-nothing system

Each CPU has local main memory and disk space, but no two CPUs can access the same storage area; all communication between CPUs is through a network connection.

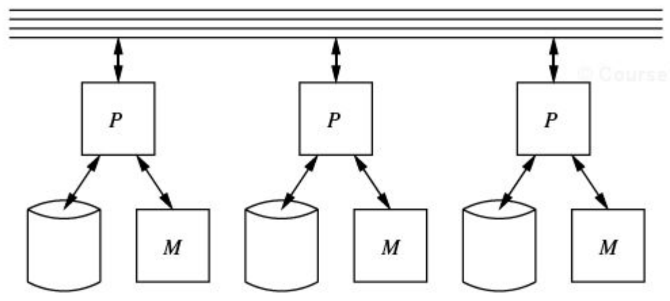

Figure 4: Shear nothing system

3.2. Type of Parallelism

3.2.1. Pipeline Parallelism (Inter-operator parallelism)

By streaming the output of one operator into the input of another operator, the two operators can work in series giving pipelined parallelism [7] . Ordered (or partially ordered) tasks and different machines are performing different tasks.

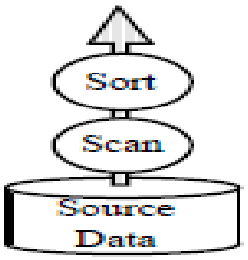

Figure 5: Parallel pipeline

3.2.2. Partitioned Parallelism (Intra-operator parallelism)

By partitioning the input data among multiple processors and memories, an operator can often be split into many independent operators each working on a part of the data. This partitioned data and execution gives partitioned parallelism [7].

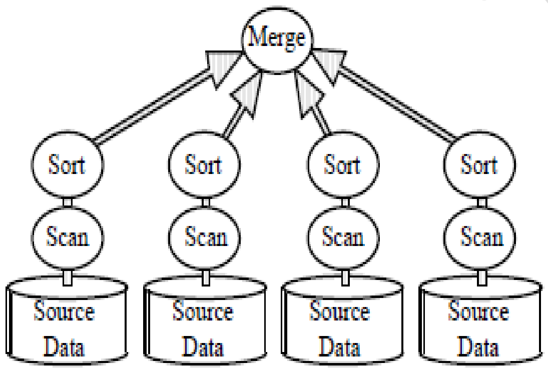

Figure 6: Partition parallelism

A task divided over all machines to run in parallel.

3.3. Parallelism Goals and Metrics: Speedup and Scaleup

The ideal parallel system [3][7] demonstrates two key properties: (1) linear speedup: Twice as much hardware can perform the task in half the elapsed time, and (2) linear scaleup: Twice as much hardware can perform twice as large a task in the same elapsed time

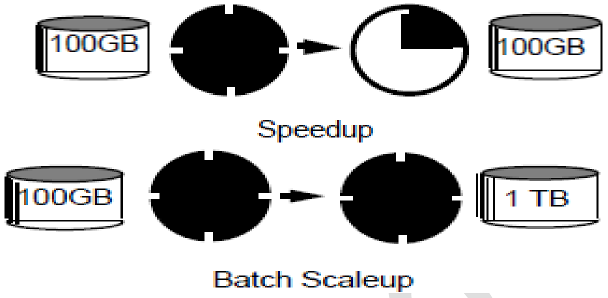

Figure 7: Speedup and Batch speedup

A speedup [7] design performs a one-hour job four times faster when run on a four-time larger system. A scaleup [7] design runs ten-times bigger job is done in the same time by a ten-time bigger system.
More formally, given a fixed job run on a small system, and then run on a larger system, the speedup given by the larger system is measured as:

$$\text{Speedup} = \frac{\text{small\_system\_elapsed\_time}}{\text{big\_system\_elapsed\_time}}$$

Speedup is said to be linear, if an N-times large or more expensive system yields a speedup of N.
Speedup holds the problem size constant, and grows the system. Scaleup measures theability to grow both the system and the problem. Scaleupis defined as the ability of N-timeslarger system to perform an N-times larger job in the same elapsed time as the original system.The scaleup metric is.

$$\text{Scaleup} = \frac{\text{small\_system\_elapsed\_time\_on\_small\_problem}}{\text{big\_system\_elapsed\_time\_on\_big\_problem}}$$

If this scaleup equation evaluates to 1, then the scaleup is said to be linear4. There are twodistinct kinds of scaleup,

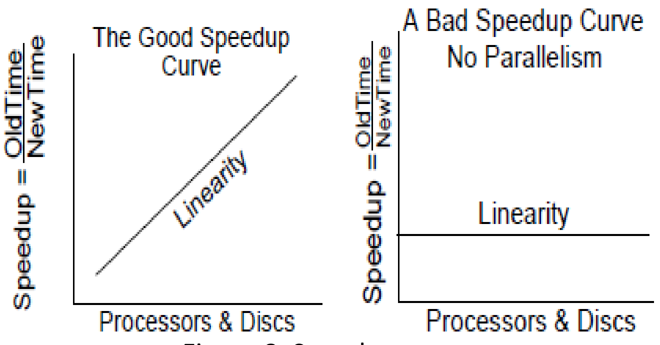

Figure 8: Speedup curve

batch and transactional. If the job consists of performing many smallindependent requests submitted by many clients and operating on a shared database, then scaleupconsists of N-times as many clients, submitting N-times as many requests against an N-timeslarger database. This is thescaleup typically found in transaction processing systems and timesharing systems. This form of scaleup is used by the Transaction Processing PerformanceCouncil to scale up their transaction processing benchmarks [GRAY91]. Consequently, it iscalled transaction-scaleup. Transaction scaleup is ideally suited to parallel systems since eachtransaction is typically a small independent job that can be run on a separate processor.

A second form of scaleup, called batch scaleup, arises when the scaleup task is presentedas a single large job. This is typical of database queries and is also typical of scientificsimulations. In these cases, scaleup consists of using an N-times larger computer to solve an Ntimeslarger problem. For database systems batch scaleup translates to the same query on an Ntimeslarger database; for scientific problems, batch scaleup translates to the same calculation onan N-times finer grid or on an N-times longer simulation.

The generic barriers to linear speedup and linear scaleup are the triple threats of:

Startup: The time needed to start a parallel operation. If thousands of processes must bestarted, this can easily dominate the actual computation time.

Interference: The slowdown each new process imposes on all others when accessing sharedresources.

Skew: As the number of parallel steps increases, the average sized of each step decreases, but the variance can well exceed the mean. The service time of a job is the service time of the slowest step of the job. When the variance dominates the mean, increased parallelism improves elapsed time only slightly.

4. Peer-To-Peer Database (P2P)

Peer-to-peer (P2P) [8][10] computing has been hailed as a promising technology that will reconstruct the architecture of distributed computing (or even that of the Internet).

This is because it can harness various resources (including computation, storage and bandwidth) at the edge of the Internet, with lower cost of ownership, and at the same time enjoys many desirable features (e.g., scalability, autonomy, etc.). Since mid- 2000, P2P computing technology has spurred increasing interests in both industrial and academic communities. As such, there are increasingly more applications being developed based on this paradigm [5]. For example, digital content sharing (e.g., Naspter, Gnutella and Shareaza), scientific computation (e.g., BOINC and Folding@home), collaborative groupware (e.g., Groove), instant messages (e.g., ICQ) and so on. Furthermore, many research topics related to P2P computing have also been studied extensively—overlay network, routing strategies, resource location and allocation, query processing, replication, and caching. However, there has not been much effort to study the architecture of P2P systems. As the architecture of a system is the cornerstone of high-level applications that are implemented upon it, an understanding of P2P architecture is crucial to realizing its full potential. Such a study is important because: (a) It helps researchers, developers, and users to better appreciate the relationships and differences between P2P and other distributed computing paradigms (e.g., client-server and grid computing). (b) It allows us to be conscious of the potential merits of P2P computing for newly emerging application demands, and to determine the most suitable architecture for them. (c) It enables us to determine the architectural factors that are critical to a P2P system's performance, scalability, reliability, and other features.

4.1. Taxonomy of P2P systems

This taxonomy [8] [10] is derived from examining existing P2P systems. Figure shows the taxonomy. In general, we can categorize the systems into two broad categories, centralized vs. decentralized, based on the availability of one or more servers, and to what extent the peers depend on the services provided by those servers. As expected, most of the research focuses on decentralized systems. There are essentially two main design issues to considerin decentralized systems: (a) the structure—flat (single tier) vs. hierarchical (multitier);and (b) the overlay topology—unstructured vs. structured. Besides these twomain categories, there are also hybrid P2P systems that combine both centralizedand decentralized architectures to leverage the advantages of both architectures.

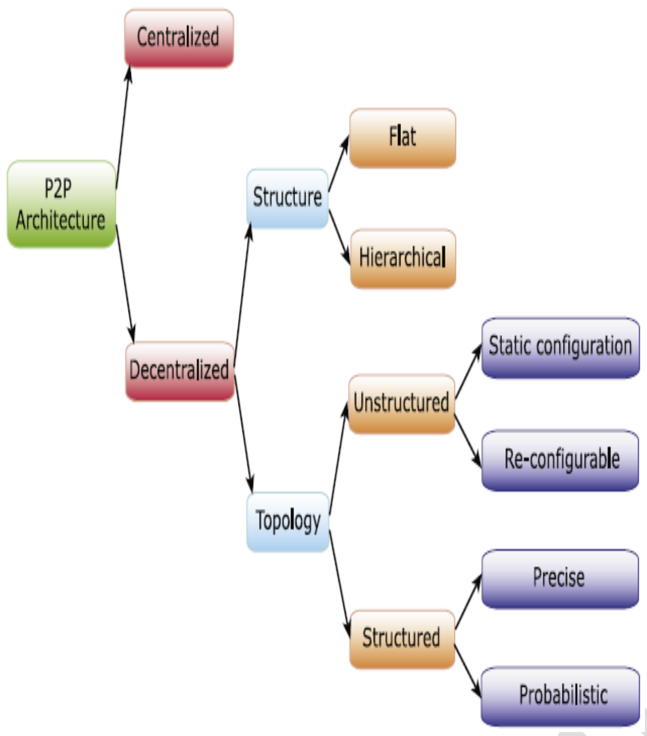

Figure 9: taxonomy of p2p systems

4.1.1. Centralized P2P Systems

Centralized P2P [10][11] systems beautifully mix the features of both centralized (e.g., client-server) and decentralized architectures. Like a client-server system, there are one or more central servers, which help peers to locate their desired resources or act as task scheduler to coordinate actions among them. To locate resources, a peer sends messages to the central server to determine the addresses of peers that contain the desired resources (e.g., Napster), or to fetch work units from the central server directly (e.g., BOINC).

4.1.2. Decentralized P2P Systems

In a decentralized P2P [5][10] system, peers have equal rights and responsibilities. Each peer has only a partial view of the P2P network and offers data/services that may be relevant to only some queries/peers. As such, locating peers offering services/data quickly is a critical and challenging issue. The advantages of these systems are obvious: (a) they are immune to single point of failure, and (b) possibly enjoy high performance, scalability, robustness, and other desirable features.

There are two dimensions in the design of decentralized P2P systems. First, the network structure can be flat (single-tier) or hierarchical (multi-tier). In a flat structure, the functionality and load are uniformly distributed among the participating nodes. It turns out that most of the existing decentralized systems are nonhierarchical. On the other hand, hierarchical design naturally offers certain advantages including fault isolation and security, effective caching and bandwidth utilization, hierarchical storage and so on. In a hierarchical structure, there are essentially multiple layers of routing structures. For example, at a national level, there may be a routing structure to interconnect states; within each state, there may be another routing structure for universities within the state; and within each university, there may be yet another level that connects departments, and so on. Representatives of this category are the super-peer architecture and the Crescendo system.
The second dimension concerns the logical network topology (the overlay network), whether it is structured or unstructured. The difference between these two designs lies in how queries are being forwarded to other nodes. In an unstructured P2P system, each peer is responsible for its own data, and keeps track of a set of neighbors that it may forward queries to. There is no strict mapping between the identifiers of objects and those of peers. This means (a) locating data in such a systemis challenging since it is difficult to precisely predict which peers maintain thequeried data; (b) there is no guarantee on the completeness of answers (unless theentire network is searched), and (c) there is no guarantee on response time (exceptfor the worst case where the entire network is searched). The famous forerunners ofunstructured P2P systems [5] are FreeNetand the original Guntella The formerapplies unicast-based lookup mechanisms to locate expected resources, whichis inefficient in terms of response time, but efficient with respect to the bandwidthconsumption and the number of messages used; the latter adopts flooding-basedrouting strategy, which is efficient in terms of response time but inefficient in bandwidthconsumption and the number of messages used (since the network is floodedwith exponential number of messages). A key issue in unstructured P2P systems isthe determination of the neighbors. These neighbors can be (pre-)determined staticallyand fixed.

However, more often, neighbors are determined based on a peer's(or rather the user's) interests. Thus, as the user interests (reflected by the queries)change, the set of neighbors may change. This is based on the inherent assumptionthat a peer is likely to be issuing similar queries during a period of time, and nodesthat have previously provided answers are likely to be contributing answers as well.

Thus, keeping these nodes as neighbors can reduce the querying time (in the immediatefuture). We refer to the latter approach as reconfigurable systems, and onesuch system is the Best Peer system.

On the contrary, in a structured P2P system, data placement is under the controlof certain predefined strategies (generally a distributed hash table, or simply DHT).

In other words, there is a mapping between data and peers. (Very often, for security/privacy reasons, the owners have full control over their own data. Instead, it isthe metadata that is being "inserted" into the P2P network, e.g., (id, ptr) pairs that indicatethat object with identifier id is located at peer pointed to by ptr. However, weshall use the term data in our discussion to refer to both.) More importantly, thesesystems provide a guarantee (precise or probabilistic) on search cost. This, however,is typically at the expense of maintaining certain additional information. Employingthe principle of the mapping, most of the structured P2P systems, including CAN, Chord, and Pastry, adopt the key-based routing (KBR) strategyto locate the desired resource. As a result, a request can be routed to the peer whomaintains the desired data quickly and accurately. However, since the placement ofdata is tightly controlled, the cost of maintaining the structured topology is high,especially in a dynamic network environment, where peers may join and leave thenetwork at will.

If the search item is a popular item, it should be found in the first few steps, and hence the search cost is not expensive. On the other hand, if the search item is a rare item, i.e., there are not enough results returned within a predefined time or a predefined number of search steps, the system performs structured search, which should locate the item if it exists in the system. As a result, the system can alleviate the problems of search in conventional unstructured P2P systems. The problem of this system, however, is that without global knowledge, it is not easy to identify if a data item is a popular item or a rare item for indexing.

4.1.3. Hybrid P2P Systems

The main advantage of centralized P2P systems is that they are able to provide a quick and reliable resource locating. Their limitation, however, is that the scalability of the systems is affected by the use of servers. While decentralized P2P systems are better than centralized P2P systems in this aspect, they require a longer time in resource locating. As a result, hybrid P2P[5][10] systems have been introduced to take advantages of both centralized and decentralized architectures. Basically, to maintain the scalability, similar to decentralized P2P systems, there are no servers in hybrid P2P systems. However, peer nodes that are more powerful than others can be selectedto act as servers to serve others. These nodes are often called super peers. In this way, resource locating can be done by both decentralized search techniques and centralized search techniques (asking super peers), and hence the systems benefit from the search techniques of centralized P2P systems.

While it is clearly that different P2P systems belonging to different categories have different advantages and disadvantages, P2P systems in the same category also have different strengths and weaknesses depending on the specific design of the systems.

This leads to the fact that different P2P systems are different in the system performance, resource location, scalability, load-balancing, autonomy, and anonymity.

In the following sections, we will examine the architectural features of outstanding P2P systems from different categories in detail. Throughout most of our discussion, we shall deal with data sharing applications. However, it should be clear that the systems can also be used for other applications, e.g., sharing of storage, processing cycles and so on.

5. Grid Database

The popularity of the Internet as well as the availability of powerful computers and high-speed network technologies as low-cost commodity components is changing the way we use computers today. These technology opportunities have led to the possibility of using distributed computers as a single, unified computing resource, leading to what is popularly known as Grid computing [9].

Grids are sharing environments implemented by the employment of a **persistent**, **standards-based**service infrastructure that supports the creation of, and resources sharing within, distributedcomities.

The term Grid is chosen as an analogy to a power Grid that provides consistent, pervasive, dependable, transparent access to electricity irrespective of its source. This new approach to network computing is known by several names, such as metacomputing, scalable computing, global computing, Internet computing, and more recently peer to peer (P2P) computing.

Grids enable the sharing, selection, and aggregation of a wide variety of resources including supercomputers, storage systems, data sources, and specialized devices that are geographically distributed and owned by different organizations for solving large-scale computational and data intensive problems in science, engineering, and commerce. Thus creating virtual organizations and enterprises as a temporary alliance of enterprises or organizations that come together to share resources and skills, core competencies, or resources in order to better respond to business opportunities or large-scale application processing requirements, and whose cooperation is supported by computer networks.

The concept of Grid computing [9] started as a project to link geographically dispersed supercomputers, but now it has grown far beyond its original intent. The Grid infrastructure can benefit many applications, including collaborative engineering, data exploration, high-throughput computing, and distributed supercomputing.

The users interact with the Grid resource broker to solve problems, which in turn performs resource discovery, scheduling, and the processing of application jobs on the distributed Grid resources. From the end-user point of view,Grids can be used to provide the following types of services.

• **Computational services.** These are concerned with providing secure services for executing application jobs on distributed computational resources individually or collectively. Resources brokers provide the services for collective use of distributed resources. A Grid providing computational services is often called a computational Grid. Some examples of computational Grids are: NASA IPG, the World Wide Grid, and the NSF TeraGrid.

• **Data services.** These are concerned with proving secure access to distributed datasets and their management. To provide a scalable storage and access to the data sets, they may be replicated, catalogued, and even different datasets stored in different locations to create an illusion of mass storage. The processing of datasets is carried out using computational Grid services and such a combination is commonly called data Grids. Sample applications that need such services for management, sharing, and processing of large datasets are high-energy physics and accessing distributed chemical databases for drug design.

• **Application services.** These are concerned with application management and providing access to remote software and libraries transparently. The emerging technologies such as Web services are expected to play a leading role in defining application services. They build on computational and data services provided by the Grid. An example system that can be used to develop such services is NetSolve.

• **Information services.** These are concerned with the extraction and presentation of data with meaning by using the services of computational, data, and/or application services. The low-level details handled by this are the way that information is represented, stored, accessed, shared, and maintained. Given its key role in many scientific endeavors, the Web is the obvious point of departure for this level.

• **Knowledge services.**These are concerned with the way that knowledge is acquired, used, retrieved, published, and maintained to assist users in achieving their particular goals and objectives. Knowledge is understood as information applied to achieve a goal, solve a problem, or execute a decision. An example of this is data mining for automatically building a new knowledge.

5.1. Layered Architecture of Grid-Based

The components of a Data Grid [9][11] can be organized in a layered architecture as shown in Figure. This architecture follows from similar definitions given by Foster et al. (2001) and Baker et al. (2002). Each layer builds on the services offered by the lower layer in addition to interacting and co-operating with components and the same level (eg. Resource broker invoking VO tools). We can describe the layers from bottom to top as below:

_ **Grid Fabric:** Consists of the distributed computational resources (clusters, supercomputers), storage resources (RAID arrays, tape archives) and instruments (telescope, accelerators) connected by high-bandwidth networks. Each of the resources runs system software such as operating systems, job submission and management systems and relational database management systems (RDBMS).

_ **Communication:** Consists of protocols used to query resources in the Grid Fabric layer and to conduct data transfers between them. These protocols are built on core communication protocols such as TCP/IP and authentication

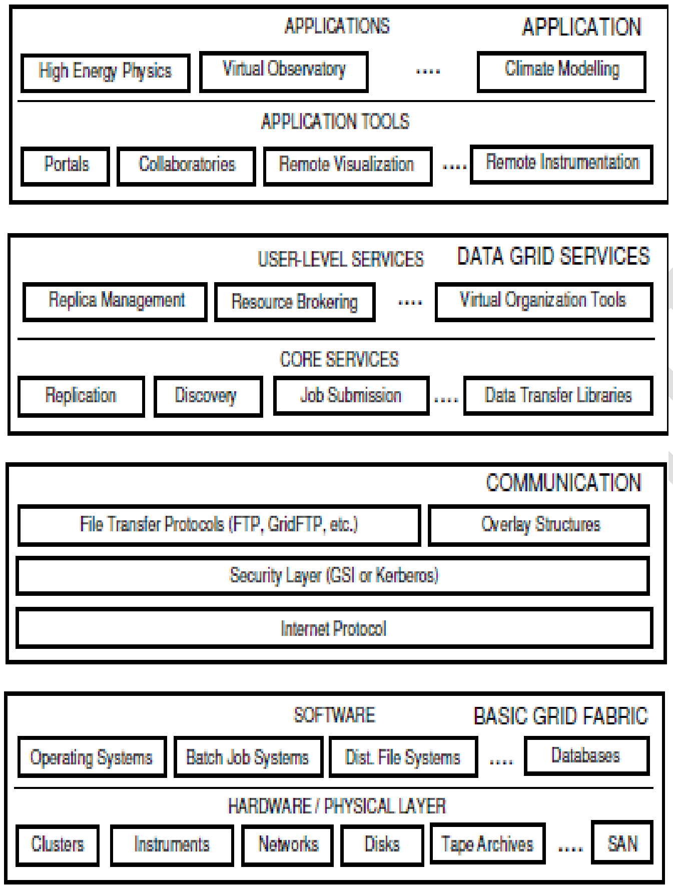

Figure 2: Layered Architecture of Grid-Based

protocols such as PKI (Public Key Infrastructure), passwords or SSL (Secure Sockets Layer). The cryptographic protocols allow verification ofusers' identities and ensure security and integrity of transferred data. These security mechanismsform part of the Grid Security Infrastructure (GSI) (Foster et al., 1998) [11]. File transferprotocols such as GridFTP (Grid File Transfer Protocol), among others, provide services for efficient transfer of data between two resources on the Data Grid. Application-specific overlay structures provide efficient search and retrieval capabilities for distributed data by maintainingdistributed indexes.

_ **Data Grid Services:** Provides services for managing and processing data in a Data Grid. Thecore level services such as replication, data discovery and job submission provide transparentaccess to distributed data and computation. User-level services such as resource brokering(selection of resources for a user based on his requirements) and replica management provide mechanisms that allow for efficient resource management hidden behind initiative commandsand APIs (Application Programming Interfaces). VO tools provide easy way to perform functionssuch as adding new resources to a VO, querying the existing resources and managingusers' access rights.

_ **Applications:** Specific services cater to users by invoking services provided by the layers below and customizing them to suit the target domains such as high energy physics, biology and climate modeling. Each domain provides a familiar interface and access to services such as visualization. Portals are web interfaces that provide single-point access to available VO services and domain-specific applications and tools. Collaboratories (Kouzes et al., 1996) have similar intent and also provide applications that allow users to conduct joint operations with their colleagues.

The security layer and Data Grid services provide applications uniform access to resources in theFabric layer while abstracting out much of the inherent complexity and heterogeneity. Formation ofVOs requires interoperability between the resources and components that are provided by differentparticipants. This motivates the use of standard protocols and service interfaces for information exchangeamong VO entities. Service interfaces themselves have to be separated from implementationdetails and have to be described in language- and platform-independent format. Realization of theserequirements have led the Grid computing research community, through forums such as Global GridForum (GGF), to adopt a new Open Grid Services Architecture (OGSA) (Foster et al., 2002) that isbased on the emerging *Web services* paradigm.

6. Comparison among Distributed , P2P and Grid Databases

Property	DDB	P2P	Grids
Purpose	Integrating existing databases, Replicating database for reliability & throughput	File sharing	Analysis, collaboration
Aggregation	Specific, Stable	Ad hoc, Dynamic	Specific, Dynamic
Organization	Centralized, federation	Centralized, two level hierarchy, flat	Hierarchical, federation, Bottom up or hybrid
Data Access Type	Equally read and write	Mostly read with frequent writes	Mostly read with rare writes
Data Discovery	Relational Schemas	Central directory, Flooded requests or document routing	Catalogues
Latency Management & Performance	Replication, Caching	Replication, Caching, Streaming	Replication, Caching, Streaming, Pre-staging, High-speed data movement
Consistency Requirements	Strong	Weak	Weak
Transaction Support	Yes	None	None currently
Computational Requirements	Transaction Processing	None currently	Data Production And Analysis
Autonomy	Operational (federated)	Operational, Participation	Access, Operational, Participation

7. Conclusion

Through this paper, I want to draw readers towards the helpful side of Distributed database, Parallel, p2p and Grid Databases. I also described Architecture of Distributed, Parallel, p2p and Grid Databases, Comparison of Distributed database and p2p and grid database also in order to make readers totally aware about the topic being described here.

References

1. Korth, Silberchatz, Sudarshan, "Database System Concepts" McGraw Hill.
2. MuhammadShahidJamal,MohsinNazir, "Fundamental Research on Distributed Database "April 2012.
3. Hiren H Darji, BinalS Shah, Manisha K Jaiswal, "Concepts Of Distributed And Parallel Database", *(IJCSITS), ISSN: 2249-9555,Vol. 2, No.6, December 2012.*
4. SrikumarVenugopal, RajkumarBuyya and RamamohanaraoKotagiri, "A Taxonomy of Data Grids for Distributed Data Sharing, Management and Processing".
5. Steven Gribble Alon Halevy Zachary Ives Maya Rodrig Dan Suciu, "What Can Databases Do for Peer-to-Peer?"
6. Mohamed E Ltabakh," Distributed Databases", Cs561-Spring 2012Wpi.
7. David J. DeWitt, Jim Gray, "Parallel Database Systems: The Future of High Performance Database Processing1", January 1992.
8. Angela Bonifati, Panos K. Chrysanthis, Aris M. Ouksel, Kai-Uwe Sattler, "Distributed Databases and Peer-to-Peer Databases: Past and Present".
9. Mark Baker1, Rajkumar Buyya2 and Domenico Laforenza3, "Grids and Grid technologies for wide-area distributed computing", *Pract. Exper.* 2002;.
10. Maurizio Lenzerini," Principles of peer-to-peer data integration".
11. N.A. Azeez1; A.P. Abidoye1; A.O. Adesina1; K.K. Agbele1; Iyamu Tiko2, and I.M. Venter1, " Peer-to-Peer Computing and Grid Computing: Towards a Better Understanding."