Introduction to Client Server Computing

The Vision of the Client/Server Computing, organizations today expect to get more value from their investments in technology. Opportunities are available to organizations and people who are ready and able to compete in the global market. A competitive global economy will ensure obsolescence and obscurity to those who cannot or are unwilling to compete. All organizations must look for ways to demonstrate value. There is more willingness to continue improvement practices and programs to eliminate redundancy and increase effectiveness. Organizations are becoming market-driven while remaining true to their business vision. Client/server computing is the most effective source for the tools that empower employees with authority and responsibility. Workstation power, workgroup empowerment, preservation of existing investments, remote network management, and market-driven business are the forces creating the need for client/server computing. The Client/server computing is an irresistible movement that is reshaping the way computers are being used. Not leaving any area and corner of the computer industry untouched. The Client/Server application development requires hybrid skills that include transaction processing, database design, communication experience, graphical user interface design, and being Internet Savvy. The more advanced applications require knowledge of distributed objects and component infrastructures.

The Client/Server Computing has changed the way in which computers are being used. This has a unique feature of having a strong foothold on the entire spectrum of the computer industry. Let us now know what a client is and what is a server?

• Client - A client is a single-user workstation that provides presentation services and the appropriate computing, connectivity and the database services and the interfaces relevant to the business need.

• Server- A server is one or more multi-user processors with shared memory providing computing, connectivity and the database services and the interfaces relevant to the business need. The Client/Server computing is an environment that satisfies the business need by appropriately allocating the application processing between the client and the server processors. The protocol is the client requests the services from the server; the server processes the request and returns the result to the client. The communication mechanism is a message passing InterProcess communication (IPC) that enables the distributed placement of the client and server processes The Client/Server is the generic model and fits what is known in the industry as the “cooperating processing”. The client/server computing is fundamentally platform independent.

Client/Server Architecture

The user of an application wants the functionality (business) it provides; the computing platform provides access to this business functionality. There are no benefits but a considerable amount of risk of exposing the platform to the users. The changes in the platform and the underlying technology should be transparent to the user, as it is understood that the systems built with transparency to the technology, for the entire user offer the highest probability of solid ongoing return for the technology investment. It is also easily demonstrable that is the developers become aware of the target platform, development will be bound to that platform and they will use special tricks and feature found in that specific platform. Characteristics of the Client and the Server The clients and the servers are the logical entities that work together over a network to accomplish a task. The distinguishing characteristics of the Client/Server systems are:

Service: The client/server is primarily a relationship between processes running on separate machines. The server process is a provider of services.

Client/Server Architecture The client is a consumer of services. In essence, client/server provides a clean separation of function based on the idea of service.

Shared Resources: A server can service many clients at the same time and regulate their access to shared resources.

Asymmetrical protocols: There is a many-to-one relationship between the clients and the server. Clients always initiate the dialog by requesting a service. Servers are passively awaiting request from the clients. In some cases, a client may pass a reference to a callback object when it invokes a service. This lets the server call back the client. So the client becomes a server.

Transparency of location: The server is a process that can reside on the same machine as the client or on a different machine across a network. Client/Server software usually masks the location of the server from the clients by the redirecting the service calls when needed. A program can be a client, a server, or both.

Mix-and-match: The ideal client/server software is independent of hardware or operating system software platforms. You should be able to mix-and-match client and server platforms.

Message-based exchanges: Clients and servers are loosely coupled systems that interact through a message-passing mechanism. The message is the delivery mechanism for the service request and replies.

Encapsulation of services: The server is a specialist. A message tells a server is requested; it is then up to the server to determine how to get the job done. Servers can be upgraded without affecting the clients as long as the published message interface is not changed.

Scalability: Client/Server systems can be scaled horizontally or vertically. Horizontal scaling means adding or removing client workstations with only a slight performance impact. Vertical scaling means either migrating to a larger and faster server machine or distributing the processing load across multiple servers.

Integrity: The server code and server data is centrally managed, which results in cheaper maintenance and the guarding of shared data integrity. At the same time, the clients remain personal and independent.

The Merits of Client/Server Computing Client/server computing provides the capability to use the most cost-effective user interface, data storage, connectivity, and application services. Frequently, client/server products are deployed within the present organization but are not used effectively. The client/server model provides the technological means to use previous investments in concert with current technology options. Organizations see opportunities to use technology to provide business solutions. Service and quality competition in the marketplace further increases the need to take advantage of the benefits available from applications built on the client/server model. Client/server computing in its best implementations moves the data-capture and information-processing functions directly to the knowledgeable worker—that is, the worker with the ability to respond to errors in the data, and the worker with the ability to use the information made available. 1. Enhanced Data Sharing Data that is collected as part of the normal business process and maintained on a server is immediately available to all authorized users. The use of Structured Query Language (SQL) to define and manipulate the data provides support for open access from all client processors and software. SQL grants all authorized users access to the information through a view that is consistent with their business need. Transparent network services ensure that the same data is available with the same currency to all designated users. 2. Integrated Services In the client/server model, all information that the client is entitled to use is available at the desktop. There is no need to change into terminal mode or log into another processor to access information. All authorized information and processes are directly available from the desktop interface.

Client/Server Architecture processing—are available and can be used to deal with information provided by application and database servers resident on the network. Desktop users can use their desktop tools in conjunction with information made available from the corporate systems to produce new and useful information. Sharing Resources among Diverse Platforms The client/server computing model provides opportunities to achieve true open system computing. Applications may be created and implemented without regard to the hardware platforms or the technical characteristics of the software. Thus, users may obtain client services and transparent access to the services provided by database, communications, and applications servers. Operating systems software and platform hardware are independent of the application and masked by the development tools used to build the application. In this approach, business applications are developed to deal with business processes invoked by the existence of a user-created "event." Data Interchangeability and Interoperability SQL is an industry-standard data definition and access language. This standard definition has enabled many vendors to develop production-class database engines to manage data as SQL tables. Almost all the development tools used for client/server development expect to reference a back-end database server accessed through SQL. Network services provide transparent connectivity between the client and local or remote servers. With some database products, such as Ingres Star, a user or application can define a consolidated view of data that is actually distributed between heterogeneous, multiple platforms. Systems developers are finally reaching the point at which this heterogeneity will be a feature of all production-class database engine products. Most systems that have been implemented to date use a single target platform for data maintenance. The ability to do high-volume updates at multiple locations and maintain database integrity across all types of errors is just becoming available with production-level quality performance and recovery. Systems developed today that use SQL are inherently transparent to data storage location and the technology of the data storage platform. The SQL syntax does not specify a location or platform.

Client/Server Architecture This transparency enables tables to be moved to other platforms and locations without affecting the application code. This feature is especially valuable when adopting proven, new technology or if it makes business sense to move data closer to its owner. Location Independence of Data and Processing We are moving from the machine-centered computing era of the 1970s and 1980s to a new era in which PC-familiar users demand systems that are user-centered. Previously, a user logged into a mainframe, mini-, or micro application. The syntax of access was unique in each platform. Function keys, error messages, navigation methods, security, performance, and editing were all very visible. Today's users expect a standard "look and feel." Users log into an application from the desktop with no concern for the location or technology of the processors involved. In most cases, client-server architecture enables the roles and responsibilities of a computing system to be distributed among several independent computers that are known to each other only through a network. This creates an additional advantage to this architecture: greater ease of maintenance. For example, it is possible to replace, repair, upgrade, or even relocate a server while its clients remain both unaware and unaffected by that change. This independence from change is also referred to as encapsulation. All the data is stored on the servers, which generally have far greater security controls than most clients. Servers can better control access and resources, to guarantee that only those clients with the appropriate permissions may access and change data. Since data storage is centralized, updates to that data are far easier to administer than what would be possible. Many mature client-server technologies are already available which were designed to ensure security, 'friendliness' of the user interface, and ease of use. It functions with multiple different clients of different capabilities.

The Demerits of Client/Server Computing (Comparions with the P2P). Traffic congestion on the network has been an issue. As the number of simultaneous client requests to a given server increases, the server can become severely overloaded.

Client/Server Architecture Contrast that to a P2P network, where its bandwidth actually increases as more nodes are added, since the P2P network's overall bandwidth can be roughly computed as the sum of the bandwidths of every node in that network. The client-server paradigm lacks the robustness of a good P2P network. Under client-server, should a critical server fail, clients’ requests cannot be fulfilled. In P2P networks, resources are usually distributed among many nodes. Even if one or more nodes depart and abandon a downloading file, for example, the remaining nodes should still have the data needed to complete the download.

Types of Servers ORB Client Server Architectures about the Fat Servers and the Fat Clients. The Architectures Stored Procedure, Remote Procedure Call (RPC) InterProcess Communication Types of Servers:

I. **File Server** • File Servers are useful for sharing information across the network • The client passes a request for file records over a network to the file server. • This is the most primitive type of data service used for exchanging messages over the network to find the requested data. • The file servers provide access to the remote server processors. In the typical implementation the software, shared data, databases and backups are stored on disk, tape and optical storage devices that are managed by the file server.

II. **Database Server** • The client passes the SQL requests as messages to the database server; the result of each SQL command is returned over the network. • The code, which processes the SQL request and the data, reside in the same machine, the server uses its own processing power to find the requested data back to the client, instead of passing all the records back to the client. This results in a much more efficient use of the distributed processing power. • Note that the application code resides on the client; thus you must either write code for the client or you can buy a shrink-wrap query tool. • The database servers provide the foundation for decision-support systems and also provide key role in data warehousing.

III. **Transaction Servers** The client can invoke remote procedure or services that reside on the server with an SQL database engine using the transaction server. • The network exchange consists of a single request/ reply. The SQL statements either all succeeded or fail as a unit.

Client/Server Architecture • These grouped SQL statements are called transactions. • With a transaction server you create the client/server application by writing the code for both the client and the server components. • The client component usually includes a Graphical User Interface (GUI). The server component consists of SQL transaction against a database. These applications are called Online Transaction Processing or OLTP. • The OLTP are mission-critical applications that require a very less response time (1-3 sec). • The OLTP applications also require tight controls over the security and integrity of the database. • It has two forms o TP-Lite: based on the stored procedures provided by database vendors. TP-Heavy: bases on the TP Monitors which is provided by OLTPvendors.

IV. **Groupware Servers** • It involves the management of semi-structured information such as text, image, mail, bulletin boards and the flow of work. • These client/server system places people in direct contact with other people. Best examples are Lotus Notes and Microsoft Exchange. • Specialized groupware software can be built on top of a vendor's canned set of client/server API's. In many cases, applications are created