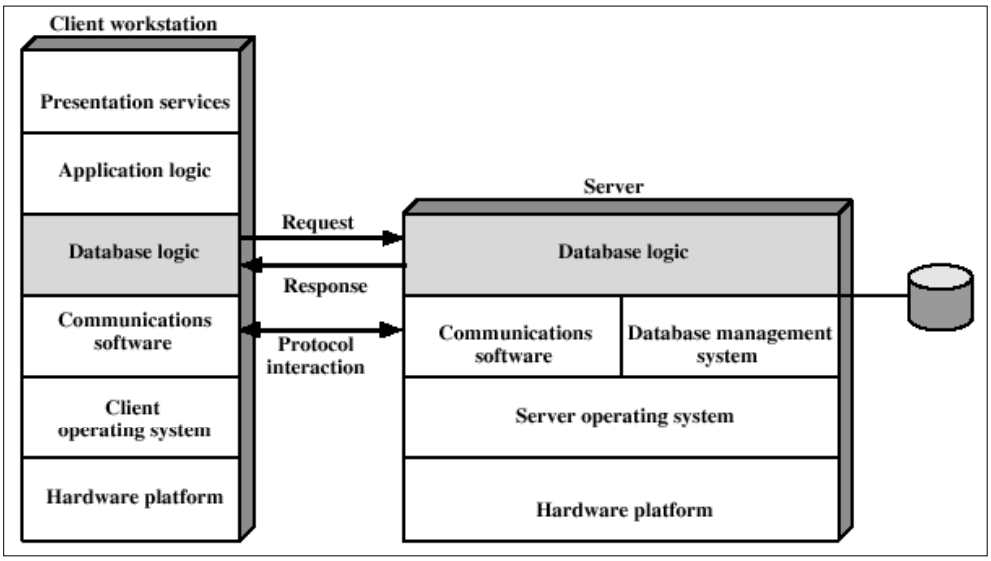
Multi-user Database Application

Hi! Today we will discuss one of the finest concepts in the modern world of computers and its relevance in terms of database management systems. Here we are giving emphasis to client-server technologies and distributed databases which facilitates the multi user environment. The security issues in a multi-user environment are dealt in lectures involving database security.

Introduction

Client/server computing is the logical extension of modular programming. Modular programming has as its fundamental assumption that separation of a large piece of software into its constituent parts creates the possibility for easier development and better maintainability. Client/server computing takes this a step farther by recognizing that those modules need not all be executed within the same memory space. With this architecture, the calling module becomes the "client" (that which requests a service), and the called module becomes the "server" (that which provides the service).The logical extension of this is to have clients and servers running on the appropriate hardware and software platforms for their functions. A “server” subsystem provides services to multiple instances of “client” subsystem Client and server are connected by a network .Control is typically a client requests services from the server provides data access and maintains data integrity To handle load, can have more than one server

Database Client-Server Architecture



What is the function Client?

The client is a process that sends a message to a server process, requesting that the server perform a ser vice. Client programs usually manage the user-interface portion of the application, validate data entered by the user, dispatch requests to server programs, and sometimes execute business logic. The client-based process is the front-end of the application that the user sees and interacts with. The client process often manages the local resources that the user interacts with such as the monitor, keyboard, CPU and peripherals. One of the key elements of a client workstation is the graphical user interface (GUI). Normally a par t of operating system i.e. the window manager detects user actions, manages the windows on the display
and displays the data in the windows.

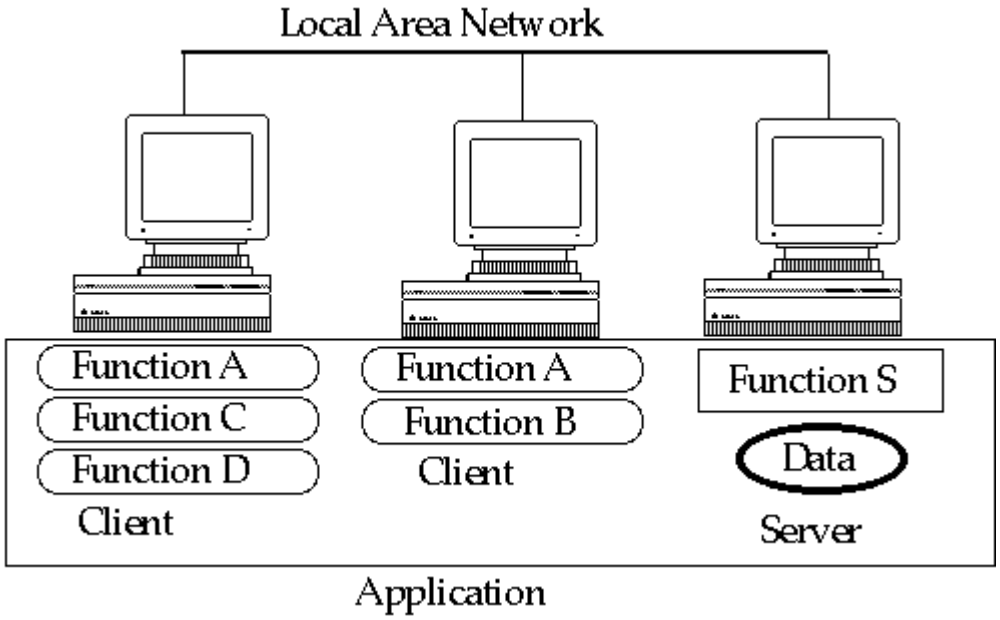
What is the function Server?

Server programs generally receive requests from client programs, execute database retrieval and updates, manage data integrity and dispatch responses to client requests. The server-based process may run on another machine on the network. This server could be the host operating system or network file server, providing file system services and

application services. The server process often manages shared resources such as databases, printers, communication links, or high powered processors. The server process performs the back-end tasks that are common to similar applications.

What do mean by Middleware?

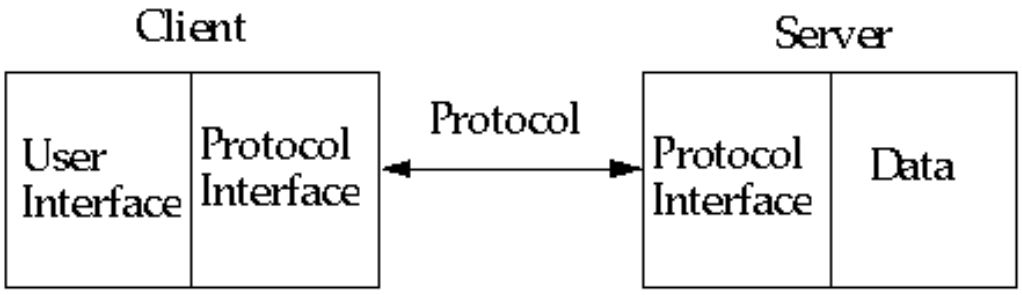
- It is a standardized interfaces and protocols between clients and back-end databases.
- It hides complexity of data sources from the end-user
- Compatible with a range of client and server options
- All applications operate over a uniform applications programming interface (API).



Why is Client-Server Different

- Emphasis on user-friendly client applications

- Focus on access to centralized databases
- Commitment to open and modular applications
- Networking is fundamental to the organization



Characteristics Of Client/Server Architectures

1) A combination of a client or front-end portion that interacts with the user, and a server or back-end portion that interacts with the shared resource. The client process contains solution-specific logic and provides the interface between the user and the rest of the application system. The server process acts as a software engine that manages shared resources such as databases, printers, modems, or high powered processors.

2) The front-end task and back-end task have fundamentally different requirements for computing resources such as processor speeds, memory, disk speeds and capacities, and input/output devices.

3) The environment is typically heterogeneous and multivendor.The hardware platform and operating system of client and server are not usually the same. Client and server processes communicate through a well-defined set of standard application program interfaces (APIs) and RPCs.

4) An important characteristic of client-server systems is scalability. Horizontal scaling means adding or removing client workstations with only a slight performance impact.

What are the issues in Client-Server Communication?

Addressing

- Hard-wired address
  - Machine address and process address are known a priori

- Broadcast-based
  - Server chooses address from a sparse address space
  - Client broadcasts request
  - Can cache response for future
- Locate address via name server

Blocking versus non-blocking

- Blocking communication (synchronous)
  - Send blocks until message is actually sent
  - Receive blocks until message is actually received

- Non-blocking communication (asynchronous)
  - Send returns immediately
  - Return does not block either

Buffering Issues

- Unbuffered communication
  - Server must call receive before client can call send

- Buffered communication
  - Client send to a mailbox
  - Server receives from a mailbox

Reliability

- Unreliable channel
  - Need acknowledgements (ACKs)
  - Applications handle ACKs
  - ACKs for both request and reply

- Reliable channel
  - Reply acts as ACK for request
  - Explicit ACK for response
- Reliable communication on unreliable channels
  - Transport protocol handles lost messages

Server architecture

- Sequential
  - Serve one request at a time
  - Can service multiple requests by employing events and asynchronous communication

- Concurrent
  - Server spawns a process or thread to service each request
  - Can also use a pre-spawned pool of threads/processes (apache)
- Thus servers could be
  - Pure-sequential, event-based, thread-based, process-based

Scalability

- Buy bigger machine!
- Replicate
- Distribute data and/or algorithms
- Ship code instead of data
- Cache

Client-Server Pros & Cons

- **Advantages**
- Networked web of computers
- Inexpensive but powerful array of processors
- Open systems
- Grows easily
- Individual client operating systems
- Cost-effective way to support thousands of users
- Cheap hardware and software
- Provides control over access to data
- User remains in control over local environment
- Flexible access to information

- **Disadvantages**
- Maintenance nightmares
- Support tools lacking
- Retraining required
- Complexity
- Lack of Maturity
- Lack of trained developers

Distributed Database Concepts

A distributed computing system consists of a number of processing elements, not necessarily homogeneous that are interconnected by a computer network, and that cooperate in performing certain assigned task. As a general goal, distributed computing systems partition a big, unmanageable problem into smaller pieces and solve it efficiently in a coordinated manner. The eonomic viability of this approach stems from two reasons:1) more computer power is harnessed to solve a complex task and 2) each autonomous processing elements can be managed independently and develop its own application.

We can define a **Distributed Database (DDB)** as a collection of multiple logically interrelated databases distributed over a computer network and a **distributed database management system (DDBMS)** as a software system that manages a distributed database while making the distribution transparent to the user

Advantages of Distributed Database

The advantages of distributed database are as follows:

1. **Management of distributed data with different levels of transparency:**

   Ideally a DBMS should be **Distributed Transparent** in the sense of hiding the details of where each file is physically stored within the system.

- ***Distribution of network transparency:***

This refers to freedom for the user from the operational details of the network. It may be divided into location transparency and naming transparency. **Location Transparency** refers to the fact that the command used to perform a task is independent of the location of the data and the location of the system where the command was issued.

Naming Transparency implies that once a name is specified the named objects can be accessed unambiguously without additional specification.

- ***Replication Transparency:***

It makes the user unaware of the existence of the copies of data.

- ***Fragmentation Transparency:***

Two type of fragmentation are possible. **Horizontal Fragmentation** distributes a relation into sets of tuples.**Vertical Fragmentation** distribute a relation into subrelations where each subrelation is defined by a subset of the column of the original relation. Fragmentation transparency makes the user unaware of the existence of fragments

2. Increased Reliability And Availability

Reliability is broadly defined as the probability that a system is running at a certain time point, whereas **availability** is the probability that a system is continuously available during a time interval. So by judiciously Replicating data and data at more than one site in distributed database makes the data accessible in some parts which is unreachable to many users.

3. Improved performance

A distributed database fragments the database by keeping data closer to where it is needed most. **Data Localization** reduces the contention for CPU and I/O service and simultaneously reduces access delays involved in wide area network. When a large database is distributed over multiple sites smaller database exist at each site .as a result local queries and transactions accessing data at a single site have a better performance

because of the smaller local database. In addition each site has a smaller number of transactions executing than if all transactions are submitted to a single centralized database. Moreover interquery and intraquery parallelism can be achieved by executing multiple queries at different sites or by breaking up a query into a number of subqueries at different sites or by breaking up a query into a number of subqueries that execute parallel. This contributes to improved performance.

4. Easier Expansion:

Expansion of the system in terms of adding more data ,increasing database sizes or adding more processors is much easier.

Additional Features Of Distributed Database

Distribution leads to increased complexity in the system design and implementation. To achieve the potential advantages; the DDBMS software must be able to provide the following function in addition to those of a centralized DBMS:

- ***Keeping Track Of data :*** The ability to keep track of the data distribution, fragmentation and replication by expanding the DDBMS catalog.
- ***Distributed query processing:*** the ability to access remote sites and transmit queries and data among the various sites via a communication network.
- ***Distributed transaction management:*** The ability to devise execution strategies for queries and transaction that access data from more than one sites and to synchronize the access to distributed data and maintain integrity of the overall database.
- ***Replicate data management:*** The ability to decide which copy of a replicated data item to access and to maintain the consistency of copies of a replicated data item.
- ***Distributed database recovery:*** the ability to recover from individual site crashes and from new types of failures such as the failure of a communication links.
- ***Security:*** Distributed transaction must be executed with the proper management of the security of the data and the authorization/access privileges of users.

- ***Distributed Directory Management:*** A directory contains information about data in the database. The directory may be global for the entire DDB or local for each site .The placement and distribution of the directory are design and policy issue.

Types of Distributed Database Systems

The term distributed database management system describes various systems that differ from one another in many respects. The main thing that all such systems have in common is the fact that data and software are distributed over multiple sites connected by some form of communication network. The first factor we consider is the **degree of homogeneity** of the DBMS software. If all servers use identical software and all users use identical software ,the DDBMS is called **homogeneous** ;otherwise it is called **heterogeneous** .Another factor related to the degree of homogeneity is the **degree of local autonomy** .If there is no provision for the local site to function as a stand alone DBMS,then the system has no local autonomy. On the other hand if direct access by local transaction to a server is permitted, the system has some degree of local autonomy.

Review Questions

1. What do you understand by Client-Server architecture?
2. What is the function of middleware in C-S architecture?
3. What are the characteristics of C-S architecture?
4. What are the advantages & disadvantages of C-S architecture?

References:

Date, C, J, Introduction to Database Systems, 7th edition

Database Management Systems, By Alexis Leon & Mathews Leon