Three-Tier Architecture

Three-tier architecture is a client–server software architecture pattern in which the user interface (presentation), functional process logic ("business rules"), computer data storage and data access are developed and maintained as independent modules, most often on separate platforms.[3] It was developed by John J. Donovan in Open Environment Corporation (OEC), a tools company he founded in Cambridge, Massachusetts.

Apart from the usual advantages of modular software with well-defined interfaces, the three-tier architecture is intended to allow any of the three tiers to be upgraded or replaced independently in response to changes in requirements or technology. For example, a change of operating system in the *presentation tier* would only affect the user interface code.

Typically, the user interface runs on a desktop PC or workstation and uses a standard graphical user interface, functional process logic that may consist of one or more separate modules running on a workstation or application server, and an RDBMS on a database server or mainframe that contains the computer data storage logic. The middle tier may be multitiered itself (in which case the overall architecture is called an "*n*-tier architecture").

Three-tier architecture:

Presentation tier
: This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing and shopping cart contents. It communicates with other tiers by which it puts out the results to the browser/client tier and all other tiers in the network. (In simple terms it is a layer which users can access directly such as a web page, or an operating systems GUI)

Application tier (business logic, logic tier, or middle tier)
: The logical tier is pulled out from the presentation tier and, as its own layer, it controls an application's functionality by performing detailed processing.

Data tier
: The data tier includes the data persistence mechanisms (database servers, file shares, etc.) and the data access layer that encapsulates the persistence mechanisms and exposes the data. The data access layer should provide an API to the application tier that exposes methods of managing the stored data without exposing or creating dependencies on the data storage mechanisms. Avoiding dependencies on the storage mechanisms allows for updates or changes without the application tier clients being affected by or even aware of the change. As with the separation of any tier, there are costs for implementation and often costs to performance in exchange for improved scalability and maintainability.

Three-tier architecture is an architectural deployment style that describe the separation of functionality into layers with each segment being a tier that can be located on a physically separate computer. They evolved through the component-oriented approach, generally using platform specific methods for communication instead of a message-based approach.

This architecture has different usages with different applications. It can be used in web applications and distributed applications. The strength in particular is when using this architecture over distributed systems. In this course work, I will furthermore invest this through the example of three-tier architecture in web applications.

Structure

Using this architecture the software is divided into 3 different tiers: ***Presentation tier, Logic tier, and Data tier***. Each tier is developed and maintained as an independent tier

1-Presentation tier

This is the topmost level of the application. The presentation layer provides the application's user interface (UI). Typically, this involves the use of Graphical User Interface for smart client interaction, and Web based technologies for browser-based interaction. The presentation tier displays information related to such services as browsing merchandise, purchasing, and shopping cart contents. It communicates with other tiers by outputting results to the browser/client tier and all other tiers in the network.

2-Logic tier (called also business logic, data access tier, or middle tier)

The logic tier is pulled out from the presentation tier and, as its own layer; it controls an application's functionality by performing detailed processing. Logic tier is where mission-critical business problems are solved. The components that make up this layer can exist on a server machine, to assist in resource sharing. These components can be used to enforce business rules, such as business algorithms and legal or governmental regulations, and data rules, which are designed to keep the data structures consistent within either specific or multiple databases. Because these middle-tier components are not tied to a specific client, they can be used by all applications and can be moved to different locations, as response time and other rules require. For example, simple edits can be placed on the client side to minimize network round-trips, or data rules can be placed in stored procedures.

3-Data tier

This tier consists of database servers, is the actual DBMS access layer. It can be accessed through the business services layer and on occasion by the user services layer. Here information is stored and retrieved. This tier keeps data neutral and independent from application servers or business logic. Giving data its own tier also improves scalability and performance. This layer consists of data access components (rather than raw DBMS connections) to aid in resource sharing and to allow clients to be configured without installing the DBMS libraries and ODBC drivers on each client. An example would be a computer hosting a database management system (DBMS), such as a Microsoft SQL Server database.

Components Interconnections

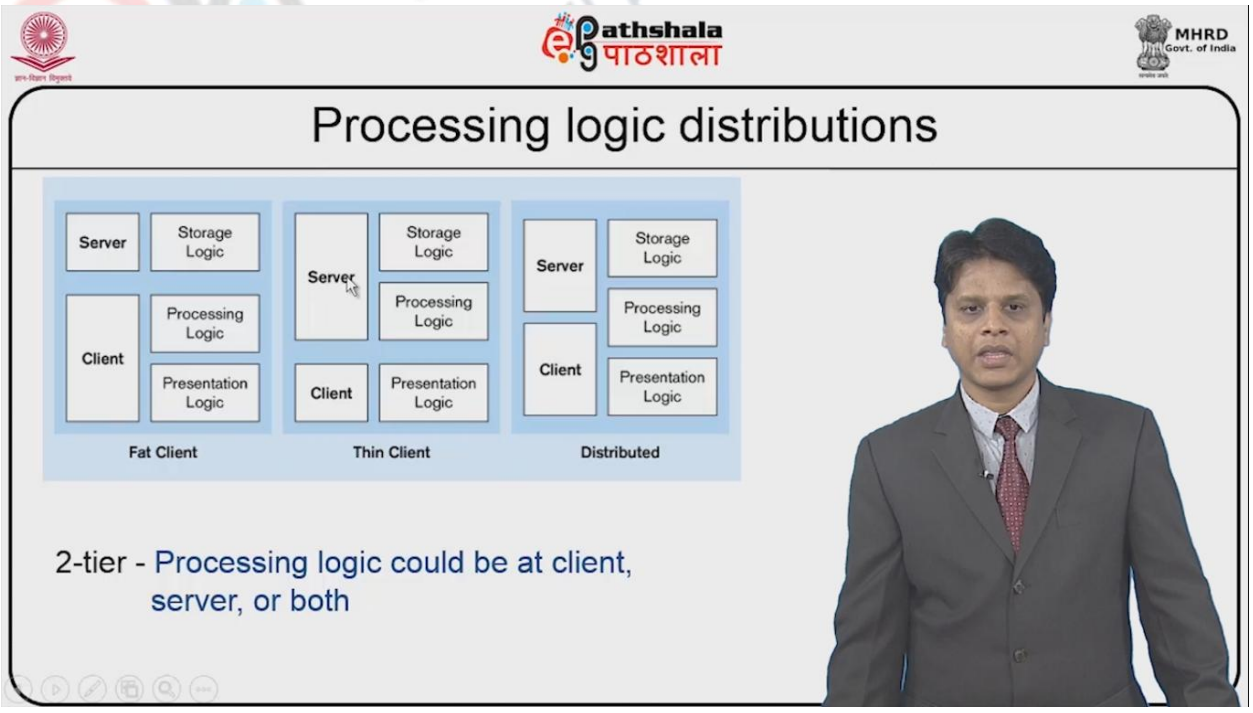
3 tier application architecture is characterized by the functional decomposition of applications, service components, and their distributed deployment, providing improved scalability, availability, manageability, and resource utilization. During an application's life cycle, the three-tier approach provides benefits such as reusability, flexibility, manageability, maintainability, and scalability. Each tier is completely independent from all other tiers, except for those immediately above and below it. You can share and reuse the components and services you create, and you can distribute them across a network of computers as needed. You can divide large and complex projects into simpler projects and assign them to different programmers or programming teams. You can also deploy components and services on a server to help keep up with changes, and you can redeploy them as growth of the application's user base, data, and transaction volume increases.

Logic layer is moved outside the presentation layer and into the business layer as it enhances reuse. As applications grow, applications often grow into other realms. Applications may start out as a web application, but some of the functionality may later be moved to a smart client application. Portions of an application may be split between a web site and a web or windows service that runs on a server. In addition, keeping logic helps aid in developing a good design (sometimes code can get sloppier in the UI).

The main benefits of the 3-tier architectural style are:

- Maintainability. Because each tier is independent of the other tiers, updates or changes can be carried out without affecting the application as a whole.
- Scalability. Because tiers are based on the deployment of layers, scaling out an application is reasonably straightforward.
- Flexibility. Because each tier can be managed or scaled independently, flexibility is increased.
- Availability. Applications can exploit the modular architecture of enabling systems using easily scalable components, which increases availability.

Consider the 3-tier architectural style if the processing requirements of the layers in the application differ such that processing in one layer could absorb sufficient resources to slow the processing in other layers, or if the security requirements of the layers in the application differ. For example, the presentation layer should not store sensitive data, while this may be stored in the business and data layers. The 3-tier architectural style is also appropriate if you want to be able to share business logic between applications, and you have sufficient hardware to allocate the required number of servers to each tier.



Thin Client

A thin client is designed to be especially small so that the bulk of the data processing occurs on the server. Although the term thin client often refers to software, it is increasingly used for the computers, such as network computers and Net PCs, that are designed to serve as the clients for client/server architectures. A thin client is a network computer without a hard disk drive. They act as a simple terminal to the server and require constant communication with the server as well.

Thin clients provide a desktop experience in environments where the end user has a well-defined and regular number of tasks for which the system is used. Thin clients can be found in medical offices, airline ticketing, schools, governments, manufacturing plants and even call centers. Along with being easy to install, thin clients also offer a lower total cost of ownership over thick clients.

Thick Clients

In contrast, a thick client (also called a fat client) is one that will perform the bulk of the processing in client/server applications. With thick clients, there is no need for continuous server communications as it is mainly communicating archival storage information to the server. As in the case of a thin client, the term is often used to refer to software, but again is also used to describe the networked computer itself. If your applications require multimedia components or that are bandwidth intensive, you'll also want to consider going with thick clients. One of the biggest advantages of thick clients rests in the nature of some operating systems and software being unable to run on thin clients. Thick clients can handle these as it has its own resources.

Thick vs. Thin - A Quick Comparison

Thin Clients

- Easy to deploy as they require no extra or specialized software installation

- Needs to validate with the server after data capture

- If the server goes down, data collection is halted as the client needs constant communication with the server

- Cannot be interfaced with other equipment (in plants or factory settings for example)

- Clients run only and exactly as specified by the server

- More downtime

-Portability in that all applications are on the server so any workstation can access

- Opportunity to use older, outdated PCs as clients

Thick Clients

- More expensive to deploy and more work for IT to deploy

- Data verified by client not server (immediate validation)

- Robust technology provides better uptime

- Only needs intermittent communication with server

- More expensive to deploy and more work for IT to deploy

- Require more resources but less servers

- Can store local files and applications

- Reduced server demands

- Increased security issues

- Reduced security threat

Benefits of client/server architecture:

Benefits of the Oracle client/server architecture in a distributed processing environment include the following:

- Client applications are not responsible for performing any data processing. Client applications can concentrate on requesting input from users, requesting desired data from the server, and then analyzing and presenting this data using the display capabilities of the client workstation or the terminal (for example, using graphics or spreadsheets).
- Client applications can be designed with no dependence on the physical location of the data. If the data is moved or distributed to other database servers, the application continues to function with little or no modification.
- Oracle exploits the multitasking and shared-memory facilities of its underlying operating system. As a result, it delivers the highest possible degree of concurrency, data integrity, and performance to its client applications.
- Client workstations or terminals can be optimized for the presentation of data (for example, by providing graphics and mouse support) and the server can be optimized for the processing and storage of data (for example, by having large amounts of memory and disk space).
- If necessary, Oracle can be *scaled*. As your system grows, you can add multiple servers to distribute the database processing load throughout the network (*horizontally scaled*). Alternatively, you can replace Oracle on a less powerful computer, such as a microcomputer, with Oracle running on a minicomputer or mainframe, to take advantage of a larger system's performance (*vertically scaled*). In either case, all data and applications are maintained with little or no modification, since Oracle is portable between systems.
- In networked environments, shared data is stored on the servers, rather than on all computers in the system. This makes it easier and more efficient to manage concurrent access.
- In networked environments, inexpensive, low-end client workstations can be used to access the remote data of the server effectively.
- In networked environments, client applications submit database requests to the server using SQL statements. Once received, the SQL statement is processed by the server, and the results are returned to the client application. Network traffic is kept to a minimum because only the requests and the results are shipped over the network.

Client-Server Security Overview

When configuring the security for a Sun Ray environment, you should evaluate the security requirements. You can choose one of the following security policies between the Sun Ray server and clients:

- Enable encryption for upstream traffic only (client to server)
- Enable encryption for downstream traffic only (server to client)

- Enable bidirectional encryption
- Enable server authentication
- Disable client authentication

Additionally, you must decide whether to enable hard security mode for encryption and client authentication.

You can use the **utcrypto** command or the Admin GUI to configure the encryption option, authentication option, and security mode.

Encryption and Authentication

By default, data packets between the Sun Ray server and client are sent "in the clear." This policy means that outsiders can easily "snoop" the traffic and recover vital and private user information, which malicious users might misuse. To avoid this type of attack, Sun Ray Software administrators can enable traffic encryption through the ARCFOUR encryption algorithm.

The ARCFOUR encryption algorithm, selected for its speed and relatively low CPU overhead, supports a higher level (128-bit) of security between Sun Ray services and clients.

However, encryption alone does not provide complete security. Spoofing a Sun Ray server or a Sun Ray Client and posing as either is still possible, if not necessarily easy. Here are some examples:

- A man-in-the-middle attack, in which an impostor claims to be the Sun Ray server for the clients and pretends to be the client for the server. The imposter then intercepts all messages and has access to all secure data.
- Manipulating a client to pretend to be another client in order to gain access to sessions connected to the spoofed client.

Server and client authentication provided by Sun Ray Software can resolve these types of attacks. Server authentication uses a single pre-configured, public-private key pair in the Sun Ray Software and firmware, and client authentication uses an automatically generated public-private key pair in every client.

Sun Ray Software uses the Digital Signature Algorithm (DSA) to verify that clients are communicating with a valid Sun Ray server and that the server is communicating with a legitimate client. This authentication scheme is not completely foolproof, but it mitigates trivial man-in-the-middle attacks and makes spoofing Sun Ray servers or Sun Ray Clients harder for attackers.

Enabling encryption and authentication is optional. The system or network administrator can configure it based on site requirements. By default only client authentication is enabled.

Problems of Parallel Processing

Effective implementation of parallel processing involves two challenges:

- structuring tasks so that certain tasks can execute at the same time (in parallel)
- preserving the sequencing of tasks which must be executed serially

Characteristics of a Parallel System

A parallel processing system has the following characteristics:

- Each processor in a system can perform tasks concurrently.
- Tasks may need to be synchronized.
- Nodes usually share resources, such as data, disks, and other devices.

Parallel Processing for SMPs and MPPs

Parallel processing architectures may support:

- clustered and massively parallel processing (MPP) hardware, in which each node has its own memory
- single memory systems-also known as symmetric multiprocessing (SMP) hardware, in which multiple processors use one memory resource

Clustered and MPP machines have multiple memories, with each CPU typically having its own memory. Such systems promise significant price/performance benefits by using commodity memory and bus components to eliminate memory bottlenecks.

Database management systems that support only one type of hardware limit the portability of applications, the potential to migrate applications to new hardware systems, and the scalability of applications. Oracle Parallel Server (OPS) exploits both clusters and MPP systems, and has no such limitations. Oracle without the Parallel Server Option exploits single CPU or SMP machines.

Parallel Processing for Integrated Operations

Parallel database software must effectively deploy the system's processing power to handle diverse applications: online transaction processing (OLTP) applications, decision support system (DSS) applications, as well as a mixed OLTP and DSS workload. OLTP applications are characterized by short transactions which have low CPU and I/O usage. DSS applications are characterized by long transactions, with high CPU and I/O usage.

Parallel database software is often specialized-usually to serve as query processors. Since they are designed to serve a single function, however, specialized servers do not provide a common foundation for integrated operations. These include online decision support, batch reporting, data warehousing, OLTP, distributed operations, and high availability systems. Specialized servers have been used most successfully in the area of very large databases: in DSS applications, for example.

Versatile parallel database software should offer excellent price/performance on open systems hardware, and be designed to serve a wide variety of enterprise computing needs. Features such as online backup, data replication, portability, interoperability, and support for a wide variety of client tools can enable a parallel server to support application integration, distributed operations, and mixed application workloads.

Higher Performance

With more CPUs available to an application, higher speedup and scaleup can be attained. The improvement in performance depends on the degree of inter-node locking and synchronization activities. Each lock operation is processor and message intensive; there can be a lot of latency. The volume of lock operations and database contention, as well as the throughput and performance of the IDLM, ultimately determine the scalability of the system.

Higher Availability

Nodes are isolated from each other, so a failure at one node does not bring the whole system down. The remaining nodes can recover the failed node and continue to provide data access to users. This means that data is much more available than it would be with a single node upon node failure, and amounts to significantly higher availability of the database.

Greater Flexibility

An Oracle Parallel Server environment is extremely flexible. Instances can be allocated or deallocated as necessary. When there is high demand for the database, more instances can be temporarily allocated. The instances can be deallocated and used for other purposes once they are no longer necessary.

More Users

Parallel database technology can make it possible to overcome memory limits, enabling a single system to serve thousands of users.