INTRODUCTION TO Object Oriented Systems
CHAPTER 1 Introduction to Object Oriented Systems

- Preview of Object-orientation.
- Concept of distributed object systems,
- Reasons to distribute for centralized objects.
- Client-server system architecture,
- Multi tier system architectures.
- File Server,
- Database Server,
- Group Server,
- Object Server,
- Web Server

BHUSHAN JADHAV
PROCEDURAL PROGRAMMING

Procedural programming is by far the most common form of programming. A program is a series of instructions which operate on variables. It is also known as imperative programming.

- Examples of procedural programming languages include FORTRAN, ALGOL, Pascal, C, MODULA2, Ada, BASIC. Despite their differences they all share the common characteristics of procedural programming.

Advantages of procedural programming include its relative simplicity, and ease of implementation of compilers and interpreters.

Disadvantages of procedural programming include the difficulties of reasoning about programs and to some degree difficulty of parallelization. Procedural programming tends to be relatively low level compared to some other paradigms, and as a result can be very much less productive.

BHUSHAN JADHAV
Object oriented programming is characterized by the defining of classes of objects, and their properties. Inheritance of properties is one way of reducing the amount of programming, and provision of class libraries in the programming environment can also reduce the effort required. The most widely used object oriented language is C++ which provides object extensions to C, but this is rapidly being overtaken by Java.

Features Of Object-Oriented Programming

Data Abstraction and Encapsulation

Operations on the data are considered to be part of the data type. We can understand and use a data type without knowing all of its implementation details. Neither how the data is represented nor how the operations are implemented.

We just need to know the interface (or method headers) – how to “communicate” with the object. Compare to functional abstraction with methods.
OBJECT ORIENTED PROGRAMMING

Inheritance
Properties of a data type can be passed down to a sub-type – we can build new types from old ones. We can build class hierarchies with many levels of inheritance.

Polymorphism
Operations used with a variable are based on the class of the object being accessed, not the class of the variable. Parent type and sub-type objects can be accessed in a consistent way.

<table>
<thead>
<tr>
<th>OOP’s world</th>
<th>Procedural Language world</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>structure</td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>Objects</td>
<td>structure variables</td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

BHUSHAN JADHAV
Distributed System and Middleware Concepts
A distributed system is a collection of independent computers that appears to its users as a single coherent system.

**Important characteristics of distributed systems**

- Differences between the various computers and the ways in which they communicate are hidden from users.
- Users and applications can interact with a distributed system in a consistent and uniform way, regardless of where and when interaction takes place.

**Goals of Distributed System**

4 important goals that should be met to make building a distributed system worth the effort they are:

1) Easily connect Users to resources, hide the fact that resources are distributed across a network, open, scalable.
2) **Transparency**

A distributed system that is able to present itself to users and applications as if it were only a single computer system is said to be transparent. Which hides whether a implementation i.e. software resource is in main memory or disk

<table>
<thead>
<tr>
<th>Transparency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Hide differences in data representation and how a resource is accessed</td>
</tr>
<tr>
<td>Location</td>
<td>Hide where a resource is located</td>
</tr>
<tr>
<td>Migration</td>
<td>Hide that a resource may move to another location</td>
</tr>
<tr>
<td>Relocation</td>
<td>Hide that a resource may be moved to another location while in use</td>
</tr>
<tr>
<td>Replication</td>
<td>Hide that a resource may be shared by several competitive users</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Hide that a resource may be shared by several competitive users</td>
</tr>
<tr>
<td>Failure</td>
<td>Hide the failure and recovery of a resource</td>
</tr>
<tr>
<td>Persistence</td>
<td>Hide whether a (software) resource is in memory or on disk</td>
</tr>
</tbody>
</table>
3) **Openness**

An open distributed system is a system that offers services according to standard rules.

4) **Scalability**

Scalability of a system can be measured along at least three different dimensions:
- First, a system can be scalable with respect to its size, we can easily add more users and resources to the system.
- Second, a geographically scalable system users and resources may lie far apart.
- Third, a system can be administratively scalable; it can still be easy to manage even if it spans many independent administrative organizations.
Hardware Concepts

• General Classification:
  – Multiprocessor – a single address space among the processors
  – Multicomputer – each machine has its own private memory.

• OS can be developed for either type of environment.

• Bus Based or Switch Based Architecture Of memory and processor.
Hardware Concepts

Basic organizations and memories in distributed computer systems

Shared memory

Private memory

Bus-based

Switch-based

P Processor

M Memory

BHUSHAN JADHAV
Multiprocessors

A bus-based multiprocessor.
Multiprocessors

A crossbar switch

An omega switching network
Software Concepts

- DOS (Distributed Operating Systems)
- NOS (Network Operating Systems)
- Middleware

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Main Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOS</td>
<td>Tightly-coupled operating system for multi-processors and homogeneous multicomputers</td>
<td>Hide and manage hardware resources</td>
</tr>
<tr>
<td>NOS</td>
<td>Loosely-coupled operating system for heterogeneous multicomputers (LAN and WAN)</td>
<td>Offer local services to remote clients</td>
</tr>
<tr>
<td>Middleware</td>
<td>Additional layer a top of NOS implementing general-purpose services</td>
<td>Provide distribution transparency</td>
</tr>
</tbody>
</table>
Uniprocessor Operating Systems

- Separating applications from operating system code through a microkernel
  - Can extend to multiple computers
• OS uses can be different (Windows or Linux)
• Typical services: rlogin, rcp
  – Fairly primitive way to share files
Network Operating System

- Can have one computer provide files transparently for others (NFS)

    Systems, network operating systems do not assume that the underlying hardware is homogeneous
• But no longer have shared memory
  – Provide *message passing*
  – Can try to provide *distributed shared memory*
    • But tough to get acceptable performance
Why Middleware?

Distributed operating system is not intended to handle a collection of independent computers, while network operating system does not provide a view of a single coherent system.

So an additional layer of software that is used in network operating systems to more or less hide the heterogeneity of the collection of underlying platforms but also to improve distribution transparency.
Distributed System as Middleware
CLIENT-SERVER Concepts
Client-Server Architecture Basic Concepts

Client-Server:
The *client* is the entity accessing the remote resource and the *server* provides access to the resource. Operationally, the client is the *caller* and the server is the *callee*.

In Java terms:

The client is the invoker of the method and the server is the object implementing the method.

The client and the server can be heterogeneous:
- Different implementation languages
- Different operating systems

nç The roles can be transient
The definition is with respect to a particular interaction

Client and Server refer both to the code and the system on which the code is running
Client Server Interactions

1. Send message to call $F$ with parameter $X$
2. Receive message that $F$ was called with the given parameter
3. Send message with the result of calling $F$
4. Receive message with the result of calling $F$

$y = F(x)$

$F(x) \{$
  return 5;
$\}$

Client

Server

Network

BHUSHAN JADHAV

23
Tiered architecture

• Single tiered architecture. (Monolithic applications)
  • Ex:- C,Foxpro,ADA etc.

• Two tiered architecture.
  • Client – Server systems
  • Ex:- Unix client & Mainframes Server.
Tiered architecture

- Three tiered architecture.

  - Client
  - Server
  - Database

- Ex: Yahoo Mail Server
Tiered architecture

- **N-Tiered architecture**: Various components that make up the application are logically separated or distributed across network.
  - Client ↔ Server ↔ Server ↔ Database
  - CitiBank ATM implementation

![Diagram of tiered architecture](Image)
N-Tier Architecture

• **Front end (Client):**
  - Viewed and manipulated by the users.
  - It can live in a Web browser or a standalone application.
  - Presents customized information to clients requirements.
  - Servlets and JSP is used as Front end development.

• **Middle:**
  - Contains business logic Ex: Discounts.
  - It may contain two sub-tiers:
    - Web Tier – It handles communication to client.
    - EJB Tier – It manages business logic and access to corporate data.

• **Backend (Database):**
  - Provides access to various corporate data stores (Databases, E-Mail system, Legacy systems...)
N-Tiered architecture

Diagram showing the components of an N-Tiered architecture:
- Presentation tier
- Business tier
- Resource tier
- Database

Workstations connected to the presentation tier, which in turn connects to the business and resource tiers, leading to the database.
FILE SERVER

-> A file server is a device that manages file operations and is shared by each of the client PCs attached to the LAN

-> Each file server acts as an additional hard disk for each of the client PCs

-> Each PC may be called a FAT CLIENT (most processing occurs on the client) Entire files are transferred from the server to the client for processing.
Problems with file server architecture

1. Huge amount of data transfer on the network, because when client wants to access data whole table(s) transferred to PC – so server is doing very little work, network is transferring large blocks of data and client is busy with extensive data manipulation.

2. Each client is authorised to use the DBMS when a database application program runs on that PC. Thus there is one database but many concurrently running copies of the DBMS (one on each active PC) – so heavy resource demand on clients.
File Server Architecture

**FAT CLIENT**

- Requests for data
- Requests to lock data

**File Server**
- File storage
- Record locking
- Acts like extra hard disk to client
- Not very busy
- Significant LAN traffic

**Client**
- Process/scan tables
- Application program
  - user interface
  - database processing
  - generate queries
- Handle integrity and security
- Full DBMS

**LAN**
- Entire file of data
- Lock status
Database server architectures

- After the file-server approach came two-tiered approaches
- Client is responsible for managing user interface, I/O processing logic, data processing logic and some business rules logic (front-end programs)
- Database server performs all data storage and access processing (back-end functions) – DBMS is only on server
- Advantages include: Clients do not have to be as powerful, only the database server requires processing power adequate to handle the database – therefore the server can be tuned to optimise data processing performance
- Greatly reduces data traffic on the network, as only those records (rather than tables) that match the requested criteria are transmitted to the client
- Improved data integrity since it is all processed centrally
Database server architectures

• With a database server, the client passes SQL requests as messages to the database server. The results of each SQL command are returned over the network. The server uses its own processing power to find the request data instead of passing all the records back to the client and then getting it find its own data. The result is a much more efficient use of distributed processing power. It is also known as SQL engine.
Database server architecture

Clients

DBMS only on server
A **Transaction server** is a **software component** that is used in implementing **transactions**. A **transaction** involves multiple steps which must be **completed atomically**.

For example, when paying someone from your bank, the system must guarantee that the money is taken from your account and paid into the other person’s account. It would simply be unacceptable for just one or the other action to take place; both must occur in order for the transaction to have taken place.
TRANSACTION SERVER

With a transaction server, the client invokes remote procedures that reside on the server with an SQL database engine. These remote procedures on the server execute a group of SQL statements. The network exchange consists of a single request/reply message. The SQL statements either all succeed or fail as a unit.

OBJECT SERVER

With an object server, the Client/Server application is written as a set of communicating objects. Client object communicate with server objects using an Object Request Broker (ORB). The client invokes a method on a remote object. The ORB locates an instance of that object server class, invokes the requested method and returns the results to the client object. Server objects must provide support for concurrency and sharing. The ORB brings it all together.
GROUPWARE SERVER

A **SERVER** which manages the interaction between a number of users of a **GROUPWARE** system

Web Servers

Clients and servers communicate using the Hypertext Transfer Protocol (HTTP)

- Client and server establish TCP connection
- Client requests content
- Server responds with requested content
- Client and server close connection (usually)

Examples

<table>
<thead>
<tr>
<th>Apache tomcat Server</th>
<th>Sun Java System Web Server</th>
<th>Glassfish Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Information Services</td>
<td>BHUSHAN JADHAV</td>
<td>Jigsaw Server</td>
</tr>
</tbody>
</table>
APPLICATION SERVER

An Application server is a software framework dedicated to the efficient execution of procedures (scripts, routines, programs, ...) for supporting the construction of applications. The term was created in the context of web applications. In these, the application server acts as a set of components accessible to the software developer through an API defined by the platform itself.

SOME POPULAR APPLICATION SERVER PROVIDER

1) Apache Tomcat (Apache Software Foundation)
2) TIBCO Server (TIBCOsoft)
3) tc Server (SpringSource)
4) WebSphere Application Server (IBM)
5) Sybase Enterprise Application Server (Sybase Inc)
6) WebLogic Server (Oracle)
7) JBoss (Red Hat)
8) JRun (Adobe Systems)
9) Apache Geronimo (Apache Software Foundation)
10) Oracle OC4J (Oracle)
11) Sun GlassFish Enterprise Server (Sun Microsystems)
12) SAP Netweaver AS (ABAP/Java) (SAP)
13) Glassfish Application Server (open source)
14) WebObjects (Apple Inc.)
Middleware Models And Present Technologies

1) Remote Procedure Calls (RPCs).
2) CORBA
3) Remote Method Invocation (RMI).
4) Enterprise Java Beans (EJB).
5) Component Object Modeling & Distributed Component Object Modeling (COM AND DCOM with .net)
6) Web Services And SOA (Service Oriented Architecture)
• ANY QUESTION
THANK YOU