Distributed Systems Architectures

https://genuinenotes.com
Objectives

• To explain the advantages and disadvantages of different distributed systems architectures
• To discuss two principal models of distributed systems architecture - client-server and distributed object architectures
• To understand the concept of object request brokers and the principles underlying the CORBA standards
• To introduce peer-to-peer and service-oriented architectures as new models of distributed computing.
Topics covered

• Multiprocessor architectures
• Client-server architectures
• Distributed object architectures
• Inter-organisational computing
Distributed systems

• Virtually all large computer-based systems are now distributed systems.
• Here, Information processing is distributed over several computers rather than confined to a single machine.
• Distributed software engineering is therefore very important for enterprise computing systems.
System types

- **Personal systems** that are not distributed and that are designed to run on a personal computer or workstation.
- **Embedded systems** that run on a single processor or on an integrated group of processors.
- **Distributed systems** where the system software runs on a loosely integrated group of cooperating processors linked by a network.
Distributed system characteristics (Advantages)

- **Resource sharing**
  - A distributed system allows sharing of hardware and software resources such as disks, printers, files & compilers.

- **Openness**
  - Distributed systems are normally open systems,
    - means they are designed around standard protocols that allow equipment and software to be combined from different vendors.

- **Concurrency**
  - Here, several processes may operate at the same time on separate computers on the network called concurrent processing.
  - Concurrent processing to enhance performance.

- **Scalability**
  - It is the capability of the system that can be increased by adding new resources to cope with new demands in the system.
  - Increased throughput by adding new resources.

- **Fault tolerance**
  - The availability of several computers & potential for replicating information means the distributed system can be tolerant of some hardware and software failures i.e.
  - The ability to continue in operation after a fault has occurred.
Distributed system disadvantages

• Complexity
  – Typically, distributed systems are more complex than centralised systems; makes it more difficult to understand their emergent properties & to test these systems.
  – Example- rather than the performance of the system being dependent on execution speed of one processor, it depends
    • on the network bandwidth and
    • speed of the processor on the network

• Security
  – The system may be accessed from several different computers, & the traffic on the network may subject to eavesdropping.
  – This makes it more difficult to ensure that the integrity of the data in the system is maintained &
  – The services are not degraded by the denial of attack. i.e.
  – They are more susceptible to external attack.
Distributed system disadvantages

• Manageability
  – The computers in a system may be different types and run on different versions of operating system.
  – Fault in one machine may propagate to other machines with unexpected consequences.
  – Means more effort required for system management.

• Unpredictability
  – All users of the WWW know, distributed systems are unpredictable in their response.
  – Unpredictable responses depending on the system organisation and network load.
  – As all these may vary over a short period, the time taken to a user request may vary dramatically from one request to another.
Distributed systems architectures

• Client-server architectures
  – Distributed services which are called on by clients.
  – Servers that provide services are treated differently from clients that use services.

• Distributed object architectures
  – No distinction between clients and servers.
  – The server may be thought of as a set of interactive objects whose location is irrelevant.
  – Any object on the system may provide and use services from other objects.
Middleware

- The components of the distributed system may be implemented on different programming language & may execute on the completely different types of processors.
  - Models of data
  - Information representation &
  - Protocols
- Thus, it requires to manage these diverse parts, ensure that they communicate and exchange data.
- Middleware refers to software that manages and supports the different components of a distributed system. In essence, it sits in the middle of the system.
- Middleware is usually off-the-shelf rather than specially written software.
- Examples
  - Transaction processing monitors;
  - Data converters;
  - Communication controllers.
3.6 Multiprocessor architectures

- Simplest distributed system model where
- System composed of multiple processes which may (but need not) execute on different processors.
- This process is common in large real-time systems.
- These systems
  - Collect information
  - Make decision using information &
  - Send signals to actuator to modify the system’s environment
- Distribution of process to processor may be predetermined or may be under the control of a dispatcher.
Example - A multiprocessor traffic control system

Traffic flow sensors and cameras

Sensor processor

Sensor control process

Traffic flow processor

Display process

Operator consoles

Traffic light control processor

Light control process

Traffic lights

https://genuinenotes.com
Example - A multiprocessor traffic control system

• In fig. – a simplified model of the traffic control system is shown.

• A set of distributed sensors collects information on the traffic flow & processes locally

• Operators make decisions using this information & give instruction to a separate traffic light control process

• Here, there are separate logical processes for managing sensors, control room & traffic light which run on separate processors.
3.7 Client-server architectures

• The application is modelled as a set of services that are provided by servers and a set of clients that use these services.
• Clients know of servers but servers need not know of clients.
• Clients and servers are separate logical processes as shown in fig below (Fig.1)
• Several server processes can run on a single server processor so there is not necessarily 1:1 mapping between processors & processes.
Example - A client-server system

Fig. 1

https://genuinenotes.com
Example - Computers in a C/S network

Fig. 2

https://genuinenotes.com
Example - Computers in a C/S network

• Fig. 2 shows the physical architecture of the system with six client & two server computers.
• These can run the client & server processes as shown in Fig. 1
Layered application architecture—
a way to look at the application

• Presentation layer
  – Concerned with presenting information (results of a computation) to the user with all the user interaction.

• Application processing layer
  – Concerned with implementing the logic of the application.

• Data management layer
  – Concerned with managing all the database operations.

• In centralized systems, these need not be clearly separated.
Application layers

Presentation layer

Application processing layer

Data management layer

https://genuinenotes.com
Thin and fat clients

- The simplest client server architecture is called two tier client server architecture, where an application is organized as
  - a server (or multiple identical servers) &
  - a set of clients

- The two tier client server architecture can take two forms:
  - Thin-client model
    - In a thin-client model, all of the application processing and data management is carried out on the server.
    - The client is simply responsible for running the presentation software.
  - Fat-client model
    - In this model, the server is only responsible for data management.
    - The software on the client implements the application logic and the interactions with the system user.
Thin and fat clients

Thin-client model

Client

Presentation

Server

Data management

Application processing

Fat-client model

Client

Presentation

Application processing

Server

Data management

https://genuinenotes.com
Thin client model

• Used when legacy systems are migrated to client server architectures.
  – The legacy system acts as a server in its own right with a graphical interface implemented on a client.
  – The application itself act as a server-handles the application processing & data management

• A major disadvantage is that it places a heavy processing load on both the server and the network.

https://genuinenotes.com
Fat client model

• More processing is delegated to the client as the application processing is locally executed.
• The server is essentially a transaction server that manages all database transactions.
• Example- Banking ATM system, where ATM is the client & the server is a mainframe running the customer account database.
• The hardware in the teller machine carries out a lot of customer related processing associated with a transaction.
• More complex than a thin client model especially for management. New versions of the application have to be installed on all clients.
A client-server ATM system

Account server
- Tele-processing monitor
- Customer account database

ATM

https://genuinenotes.com
A client-server ATM system

- Fig. above is the ATM distributed system
- The ATMs are not connected directly to the customer database but to a teleprocessing monitor.
- It is a middleware system that organizes communication with remote clients & serializes the client transaction processing by the database.
- Using serial transaction means that the system can recover from faults without corrupting system data.
Disadvantages-Fat client model

- The fat-client model distributes processing more effectively than thin client model but the system management is more complex
- Application functionality is spread over many computers
- When the application software is to be changed, reinstallation is needed on every computer
- This can be a major cost if there are hundreds of clients in the system.
Disadvantages- Two-tier architecture

• The three logical layers-presentation, application processing & data management must be mapped onto two computer systems-the client & the server.

• This may either be problems with scalability & performance if the thin client model is chosen, or the problems of system management if the fat client model is used.

• To avoid these issues, a three-tier client server architecture is used.
Three-tier architectures

• In a three-tier architecture,
  – the presentation,
  – the application processing &
  – the data management are logically separate processes that
  – execute on a separate processor.
• Allows for better performance than a thin-client approach and is simpler to manage than a fat-client approach.
• A more scalable architecture - as demands increase, extra servers can be added.
A 3-tier C/S architecture

Client → Presentation → Server: Application processing → Server: Data management → Client
Example- An internet banking system

- Client
- Web server
  - Account service provision
  - SQL query
- Database server
  - SQL
  - Customer account database

HTTP interaction
Example- An internet banking system

• Here, the
  – bank’s customer database (usually hosted on a mainframe computer) provides data management services;
  – a web server provides application services such as transferring of cash, generate statements, pay bills etc.
  – The user’s own computer with an internet browser is the client.

• This system is scalable because it is relatively easy to add new web servers as the number of customers increase.
Advantages-A 3-tier C/S architecture

- The use of three-tier architecture is this case allows the information transfer between the web server and the database server to be optimized.
- Network traffic is reduced.
- The communications between these can use fast, low level communication protocols.
- More rapid response to clients.
- Efficient middleware that supports database queries in SQL is used to handle information retrieval from the database.
Multi-tier Architecture

• At times it is appropriate to extend three tier architecture to multi-tier where applications need to access and use the data from different databases.

• In such case, an integration server is positioned between application servers and database servers.

• The integration server collects the distributed data & presents it to the application as if it were from single database.
## Use of C/S architectures

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-tier C/S architecture with thin clients</td>
<td>Legacy system applications where separating application processing and data management is impractical. Computationally-intensive applications such as compilers with little or no data management. Data-intensive applications (browsing and querying) with little or no application processing.</td>
</tr>
<tr>
<td>Two-tier C/S architecture with fat clients</td>
<td>Applications where application processing is provided by off-the-shelf software (e.g. Microsoft Excel) on the client. Applications where computationally-intensive processing of data (e.g. data visualisation) is required. Applications with relatively stable end-user functionality used in an environment with well-established system management.</td>
</tr>
<tr>
<td>Three-tier or multi-tier C/S architecture</td>
<td>Large scale applications with hundreds or thousands of clients. Applications where both the data and the application are volatile. Applications where data from multiple sources are integrated.</td>
</tr>
</tbody>
</table>
3.8 Distributed object architectures

- There is no distinction in a distributed object architectures between clients and servers.
- Each distributable entity is an object that provides services to other objects and receives services from other objects.
- Objects may be distributed across a number of computers on a network.
- Object communication is through a middleware system called an object request broker.
- Its role is to provide a seamless interface between objects.
- It provides set of services that allows objects to communicate & to be added to & removed from the system.
- However, distributed object architectures are more complex to design than C/S systems.
Distributed object architecture

o1
S(o1)
o2
S(o2)
o3
S(o3)
o4
S(o4)
o5
S(o5)
o6
S(o6)

Object request broker

https://genuinenotes.com
Advantages of distributed object architecture

• It allows the system designer to delay decisions on where and how services should be provided.
  • Service providing objects may execute on any node of the network.
  • There is no need to decide in advance where application logic objects are located.

• It is a very open system architecture that allows new resources to be added to it as required.

• The system is flexible and scalable.

• It is possible to reconfigure the system dynamically with objects migrating across the network as required.
Uses of distributed object architecture

• As a logical model that allows to structure and organise the system. In this case, one thinks about how to provide application functionality solely in terms of services and combinations of services.

• As a flexible approach to the implementation of client-server systems. The logical model of the system is a client-server model but both clients and servers are realised as distributed objects communicating through a common communication framework.
A data mining system
Data mining system

• Here, each database can be encapsulated as a distributed object with an interface that provides read only access to its data.

• Integrator objects are each concerned with specific types of relationships, & they collect from all the databases to try to deduce the relationships.

• There might be integrator object that is concerned with seasonal variations in goods sold & another that is concerned with relationships between different types of goods.
Data mining system

• The logical model of the system is not one of service provision where there are distinguished data management services.
• It allows the number of databases that are accessed to be increased without disrupting the system.
• It allows new types of relationship to be mined by adding new integrator objects.
CORBA

• CORBA is an international standard for an Object Request Broker - middleware to manage communications between distributed objects.

• Middleware for distributed computing is required at 2 levels:
  
  – At the logical communication level, the middleware allows objects on different computers to exchange data and control information;
  
  – At the component level, the middleware provides a basis for developing compatible components. CORBA component standards have been defined.
CORBA application structure-
Object Management architecture (Siegal, 1998)
Application structure

• This architecture above proposes distribution application should be made up of number of components

• **Application objects** - that are designed & implemented for this application.

• **Standard objects** - that are defined by the OMG(Object Management Group) for a specific domain which cover Insurance, health care etc.

• **Fundamental CORBA services** that provides basic distributed computing service such as directories and security management.

• **Horizontal COBRA facilities** such as user interface facilities.
  
  – The term horizontal facilities suggests that these facilities are common to many application domains

https://genuinenotes.com
CORBA standards

• COBRA standards cover all aspect of the above vision.
• There are four major elements to these standards.
• An object model for application objects
  — A CORBA object is an encapsulation of state with a well-defined, language-neutral interface defined in an IDL (interface definition language).
• An object request broker that manages requests for object services.
• A set of general object services likely to be required by many distributed applications (e.g. Directory service)
• A set of common components built on top of these basic services.
CORBA objects

• CORBA objects are comparable, in principle, to objects in C++ and Java.
• They MUST have a separate interface definition that is expressed using a common language (IDL) similar to C++.
• There is a mapping from this IDL to programming languages (C++, Java, etc.).
• Therefore, objects written in different languages can communicate with each other.
Object request broker (ORB)

- The ORB handles object communications. It knows of all objects in the system and their interfaces.
- Using an ORB, the calling object binds an IDL stub that defines the interface of the called object.
- Calling this stub results in calls to the ORB which then calls the required object through a published IDL skeleton that links the interface to the service implementation.
ORB-based object communications

Object Request Broker

IDL stub

S(o1)

o1

IDL skeleton

S(o2)

o2
Inter-ORB communications

• ORBs are not usually separate programs but are a set of objects in a library that are linked with an application when it is developed.
• ORBs handle communications between objects executing on the same machine.
• Several ORBS may be available and each computer in a distributed system will have its own ORB.
• Inter-ORB communications are used for distributed object calls.
CORBA services

• Naming and trading services
  – These allow objects to discover and refer to other objects on the network.
• Notification services
  – These allow objects to notify other objects that an event has occurred.
• Transaction services
  – These support atomic transactions and rollback on failure.
  – Transactions are fault-tolerance facility that supports recovery from errors during an update operation.
  – If an object update operation fails, then the object state can be rolled back to its state before the update was started.
3.9 Inter-organizational computing

- For security and inter-operability reasons, most distributed computing has been implemented at the organizational level.
- An organization has a number of servers & spreads its computation load across these.
- Because these all located within the same organization, local standards, management and operational processes apply.
- Newer models of distributed computing have been designed to support inter-organizational computing rather than intra-organization distributed computing
  - where different nodes are located in different organizations.
- Two of these approaches are discussed here:
  1. Peer to Peer architectures
  2. Service oriented architectures
Peer-to-peer architectures

• Peer to peer (p2p) systems are decentralized systems where computations may be carried out by any node in the network.
• The overall system is designed to take advantage of the computational power and storage of a large number of networked computers.
• Most p2p systems have been personal systems but there is increasing business use of this technology.

https://genuinenotes.com
P2p architectural models

• One can look at architecture of P2P applications from two perspectives:

1. The logical network architecture
   – Decentralized architectures;
   – Semi-centralized architectures.

2. Application architecture
   – The generic organization of components in each architecture type; making up a p2p application.

• Here, focused on network architectures.
Decentralized p2p architecture
Decentralized p2p architecture

• Here, the nodes in the network are not simply functional elements but are also communication switches that can route data & control signals from one node to another
• Fig above represents a decentralized document management system-used by the consortium of researchers to share documents.
• Each member mains own document store
Decentralized p2p architecture

• However, when the document is retrieved, the node retrieving that document makes it available to other nodes.
• Someone who needs the document issues a search command that is sent to nodes in that locality.
• These nodes check whether they have the document.
• If so return to the requestor.
• If they do not have route search to another node.
• When the document is finally discovered, the node can route the document back to the original requestor.

https://genuinenotes.com
Semi-centralized p2p architecture
Semi-centralized p2p architecture

• With the sue of decentralized architecture the are obvious overheads in the system in that
  – same search may be process by many different nodes &
  – there is significant overhead in replicated peer communication

• Alternative- semi centralized where, within the network one or more nodes act as servers to facilitate node communications.
Service-oriented architectures

• Based around the notion of externally provided services (web services).

• A web service is a standard representation for some computational or information resource that are accessible across the web
  – A tax filing service could provide support for users to fill in their tax forms and submit these to the tax authorities.
Definition Web service (Lovelock, et al., 1996) - A generic service

• An act or performance offered by one party to another. Although the process may be tied to a physical product, the performance is essentially intangible and does not normally result in ownership of any of the factors of production.

• Service provision is therefore independent of the application using the service.
Web services

Service registry

Find

Publish

Bind

Service requestor

Service provider

service

https://genuinenotes.com
Services and distributed objects

- Provider independence.
- Public advertising of service availability.
- Potentially, run-time service binding.
- Opportunistic construction of new services through composition.
- Pay for use of services.
- Smaller, more compact applications.
- Reactive and adaptive applications.
Services standards

• Services are based on agreed, XML-based standards so can be provided on any platform and written in any programming language.

• Key standards
  – SOAP - Simple Object Access Protocol;
  – WSDL - Web Services Description Language;
  – UDDI - Universal Description, Discovery and Integration.
Services scenario

• An in-car information system provides drivers with information on weather, road traffic conditions, local information etc.
• This is linked to car radio so that information is delivered as a signal on a specific radio channel.
• The car is equipped with GPS receiver to discover its position and,
• based on that position, the system accesses a range of information services.
• Information may be delivered in the driver’s specified language.

https://genuinenotes.com
Automotive system

Weather info
Facilities info
Receiver
Sends position and information request to services
Translator
Collates information
Mobile Info Service
Transmitter
Collates information
Service discovery
Finds available services
Road traffic info
Facilities info
Traffic info
Translator
Road locator
Road traffic info
Collates information
Translator
Info stream
Language info
Receiver
Receives information stream from services
Translator
Radio
Translates digital info stream to radio signal
Locator
Discovers car position
User interface
Receives request from user

https://genuinenotes.com