Distributed Systems

Bina Ramamurthy

Introduction

- Distributed system is the one in which hardware and software components at networked computers communicate and coordinate their activity by sharing resources such as information, data, compute cycles, bandwidth and storage.
- Examples: Internet, intranet, grid and mobile computing systems.

Topics for discussion

- The Internet as a distributed system
  - Major challenges
- Client/server distributed systems
  - Issues
  - Expectations of a distributed system
- Scalability: example
- Amazon.com: Dr.Werner Vogels's talk
- Demo: Sample distributed system

Internet

- Internet is a very large distributed system.
- Interconnection of a collection of heterogeneous networks of computers.
- Protocols: IP, TCP, HTTP
- Services: world wide web (www), file transfers (ftp), email, etc.

Fundamental terms: Protocol

- **Protocol** is a set of rules that end points in a telecommunication system use when exchanging information.
  - IP: Internet protocol defines an unreliable packet transfer protocol.
  - TCP: Transmission Control Protocol builds on IP to define a reliable data delivery protocol.
  - LDAP: Lightweight Directory Access Protocol builds on TCP to define a query-response protocol for querying the state of a remote database.
  - HTTP: Hyper Text Transfer Protocol builds on TCP to facilitate hyper-text document exchange.

Fundamental terms: Service

- Service is a network-enabled entity that provides a specific capability.
- Service = Protocol + Behavior
- A service definition permits many implementations.
- Examples: ability to move files, create processes, verify access rights
- An FTP server speaks File Transfer Protocol and supports remote read and write access to a collection of files.

Major Challenges

- Heterogeneity of components
- Security
- Scalability : ability to work well when number of users increases
  - Failure handling
  - Concurrency
  - Transparency
- Reliability
- Interoperability
- Performance
- Openness

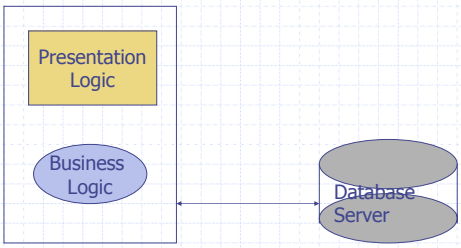
Client/Server

- Server: refers to a process on a networked computer that accepts requests from other (local or remote) processes to perform a service and responds appropriately.
- Client: requesting process in the above is referred to as the client.
- Request and response are in the form of messages.
- Client is said to invoke an operation on the server.
- Many distributed systems today are constructed out of interacting clients/servers.

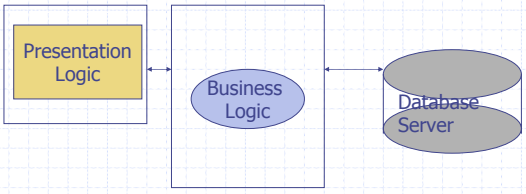
Client/server Issues

- Basic object-technology could not fulfill the promises such as reusability and interoperability fully in the context internet and enterprise level applications. Deployment was still a major problem and as a result portability and mobility were impaired.

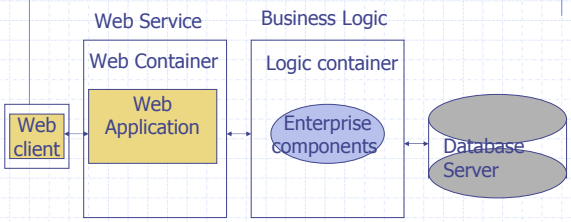
Two-tier applications



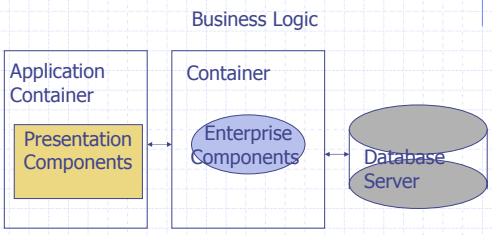
Three-tier Applications



Programming Model for Web-based applications



Application Programming Model for Three-tier Applications



Expectations of a Distributed System

- *Access transparency*: enables local and remote resources to be accessed using identical operations.
- *Location transparency*: enables resources to be accessed without knowledge of their location.
- *Concurrency transparency*: enables several processes to operate concurrently using shared resources without interference between them.
- *Replication transparency*: enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.
- *Failure transparency*: enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.
- *Mobility transparency*: allows the movement of resources and clients within a system without affecting the operation of users or programs.
- *Performance transparency*: allows the system to be reconfigured to improve performance as loads vary. "Scalability"
- *Expansion transparency*: allows the system and applications to expand in scale without change to the system structure or the application algorithms.

Scalability

Amazon.com

- Werner Vogels' talk "Order in the Chaos: Building the Amazon.com Platform."
- 1995: Started out with a single web service on a single server. Today amazon has about 150 web services on its homepage alone.
- 1 million merchant partners; 60 million customers
- One server of customers and inventory grew into two servers; more database servers were added as the business expanded
- 1999: A mistep during this exponential growth period was moving to mainframe from distributed server. Failed to meet scalability, reliability and performance; it was scratched in 2000.

Amazon (contd.)

- Robustness: Shopping cart is tested for 20000 items by a single customer, for example!
- Amazon's secret sauce is "operating reliably at scale".
- After "the denial of service" debacle in 1999, they decided to use Web services to insulate the databases from being overwhelmed by direct interaction with online applications.
- Each web service is the responsibility of a team of developers:
  - "And they are not just responsible for writing the service and then tossing it over the wall for testing and eventual entry into production where some poor maintenance geek has to look after it.
  - The Amazon CTO tells his Web services team members: "You build it. You own it."
  - That means the team is responsible for its Web service's on-going operation. If a Web service stops working in the middle of the night, team members are called to fix it."
- Web services are kept simple: complexity is the notorious enemy of reliability
- No attachment to one technology or standard: what ever customer wants, give it.

Summary

- In this course, we will study different types of distributed systems and learn to design, develop and implement distributed systems.