Database Management Systems
Client-Server System Architecture

Vikas Thammanna Gowda

01/17/2025

1 Client/Server System Architecture: Hardware Components

A client/server system architecture relies on specific hardware components to support its operation and enable communication between the client and the server. These hardware components are distributed across client devices, servers, and the network infrastructure.

1.1 Client-Side Hardware

- **End-User Devices:** Devices that users interact with to send requests and receive responses from the server. Examples include:
  - **Desktop Computers:** High-performance machines with sufficient processing power and memory for applications.
  - **Laptops:** Portable devices with similar capabilities as desktops.
  - **Mobile Devices:** Smartphones and tablets used for mobile client-server interactions.
- **Input/Output Devices:** Hardware like keyboards, mice, touchscreens, and monitors for user interaction and display of data.
- **Network Interface Card (NIC):** A hardware component that enables devices to connect to a network, typically through Ethernet or Wi-Fi.

1.2 Server-Side Hardware

- **Servers:** High-performance machines designed to handle multiple client requests simultaneously. Types include:
  - **Web Servers:** Serve web pages and manage HTTP/HTTPS traffic.
  - **Application Servers:** Execute business logic and application code.
  - **Database Servers:** Manage and store data, responding to queries from the application server.
- **Central Processing Unit (CPU):** Multi-core or high-performance processors to handle concurrent requests efficiently.
- **Memory (RAM):** Large amounts of RAM are used for caching, processing requests, and running applications efficiently.
- **Storage Devices:**
  - Hard Disk Drives (HDDs) or Solid-State Drives (SSDs) store the server's operating system, application files, and databases.
  - RAID configurations are often used for redundancy and fault tolerance.
- **Power Supply Units (PSUs):** Provide stable and reliable power to the server hardware.
- **Cooling Systems:** Ensure the hardware components maintain optimal operating temperatures.

1.3 Network Infrastructure

- **Routers:** Direct data packets between clients, servers, and other networks, acting as a gateway between local networks and the internet.
- **Switches:** Connect multiple devices on a local area network (LAN) and ensure efficient communication within the same network.
- **Firewalls:** Protect the client/server system by filtering incoming and outgoing network traffic based on security rules.
- **Load Balancers:** Distribute incoming client requests across multiple servers to prevent overloading and ensure high availability.
- **Proxy Servers:** Act as intermediaries between clients and servers to enhance security, caching, and load balancing.

1.4 Communication Hardware

- **Cables and Connectors:** Physical media like Ethernet cables for wired connections.
- **Wireless Access Points:** Provide wireless connectivity for clients in the network.
- **Modems:** Connect the local network to the internet via DSL, cable, or fiber.

1.5 Data Center Hardware (Server-Side Specific)

- **Racks and Enclosures:** Organize and house server hardware, storage devices, and networking equipment in a centralized location.
- **Uninterruptible Power Supplies (UPS):** Provide backup power to servers in case of electrical failures.
- **Backup Devices:** Tape drives, external storage, or cloud-based backup solutions for disaster recovery.
- **Server Clusters:** Multiple interconnected servers that work together to increase reliability and scalability.

1.6 Workflow

1. **Client-Side Interaction:** Users operate client devices to send requests through NICs to the network.
2. **Network Communication:** Routers, switches, and other networking hardware transmit the requests to the server.
3. **Server-Side Processing:** Servers process requests, retrieve data from storage, and use CPUs and RAM to execute application logic.
4. **Response Transmission:** Servers send the processed response back to the client via the network infrastructure.

These hardware components collectively ensure the client/server system operates efficiently, reliably, and securely.

2 Client/Server System Architecture: Software Components

A client/server system architecture divides software into distinct components that perform specialized tasks and interact to achieve a common goal. Below are the primary software components of a client/server system:

2.1 Client-Side Components

- **User Interface (UI):** The client software typically provides the graphical user interface (GUI) or command-line interface (CLI) through which users interact with the system. Examples include:
  - Web browsers.
  - Desktop applications.
- **Application Logic (Partial):** Some application logic may be implemented on the client to reduce server workload and improve responsiveness. Examples include:
  - Input validation.
  - Data preprocessing.
  - Local caching.
- **Communication Module:** This component handles communication with the server, typically through protocols such as HTTP, HTTPS, or WebSocket. It ensures that:
  - Requests are sent to the server.
  - Responses are received from the server.

2.2 Server-Side Components

- **Application Logic (Core):** The main business logic resides on the server, responsible for:
  - Processing client requests.
  - Performing calculations.
  - Managing workflows (e.g., user authentication, applying business rules, executing transactions).
- **Database Management System (DBMS):** Servers often include a DBMS to:
  - Store, retrieve, and manage data.
  - Interact with the application logic to provide persistent data storage.
- **Server Communication Module:** Handles incoming client requests, processes them, and sends responses. It often relies on:
  - APIs or services like REST or GraphQL endpoints to interact with the client.
- **Middleware (Optional):** Middleware facilitates communication and data exchange between different components or between the client and server. Examples include:
  - Message queues.
  - API gateways.
  - Caching systems (e.g., Redis).

2.3 Network Communication

- **Protocol Stack:** Both client and server use a protocol stack (e.g., TCP/IP, HTTP/HTTPS) to enable communication over the network. These protocols:
  - Standardize data exchange.
  - Ensure reliability.
- **Security Layers:** Protect data integrity and user privacy using:
  - Encryption (e.g., TLS/SSL).
  - Authentication protocols (e.g., OAuth, token-based authentication).

2.4 Shared Components

- **Application Programming Interface (API):** APIs define the contract for interaction between the client and server. They enable the server to:
  - Expose specific functions or data for client consumption.
- **Session Management:** Maintains the context of interactions using:
  - Session IDs.
  - Cookies.
  - Tokens.

2.5 Workflow

1. The client sends a request (e.g., HTTP request) to the server through the communication module.
2. The server's communication module receives the request and forwards it to the appropriate application logic.
3. The server processes the request, often querying the DBMS or another data store for necessary information.
4. The server sends the processed result back to the client, which renders the data for the user.

This modular design ensures scalability, security, and efficient division of labor between client and server.

3 Client/Server System Architecture: Implementations

Client/server implementations involve the practical deployment of a client/server architecture to deliver services, process data, and facilitate communication between clients and servers. Below is an overview of common implementation types and their characteristics:

3.1 Two-Tier Client/Server Implementation

Description: In a two-tier architecture, the client directly communicates with the server. The server handles both the application logic and the database.

- **Example:**
  - **Client:** A desktop application like MySQL Workbench.
  - **Server:** A database server like MySQL or Oracle DB.
- **Advantages:**
  - Simple and easy to implement.
  - Suitable for small-scale applications.
- **Disadvantages:**
  - Scalability issues as the number of clients grows.
  - High dependency on server performance.

3.2 Three-Tier Client/Server Implementation

Description: In a three-tier architecture, the system separates the presentation layer (client), application layer (server-side logic), and database layer.

- **Example:**
  - **Client:** A web browser (presentation layer).
  - **Application Server:** Handles business logic (e.g., a backend built with Node.js, Django, or .NET).
  - **Database Server:** Stores data (e.g., PostgreSQL, MongoDB).
- **Advantages:**
  - Better scalability than two-tier systems.
  - Easier to maintain and update application logic independently.
- **Disadvantages:**
  - More complex to implement and manage.
  - Requires more hardware resources.

3.3 Multi-Tier Client/Server Implementation

Description: Extends the three-tier architecture by adding more layers, such as caching, load balancing, or microservices for specific functionalities.

- **Example:**
  - **Client:** Mobile apps, web apps.
  - **Application Servers:** Multiple microservices for different modules (e.g., user authentication, payment processing).

  - **Database Servers:** Separate databases for various microservices.
  - **Load Balancer:** Distributes client requests across multiple servers.

- **Advantages:**
  - Highly scalable and resilient.
  - Supports distributed development and deployment.

- **Disadvantages:**
  - Increased complexity and cost.
  - Requires expertise in deployment and maintenance.

3.4 Peer-to-Peer (P2P) Client/Server Hybrid

Description: Combines client/server and peer-to-peer models, where devices act as both clients and servers, depending on the context.

- **Example:**
  - **Client:** File-sharing apps like BitTorrent.
  - **Server:** Central tracker or coordinator server for peer discovery.

- **Advantages:**
  - Reduced load on central servers.
  - High availability due to distributed nature.

- **Disadvantages:**
  - Security and data consistency challenges.
  - Difficult to manage in large-scale environments.

3.5 Web-Based Client/Server Implementation

Description: A browser acts as the client, and web servers handle the backend processing, often interacting with a database server.

- **Example:**
  - **Client:** Web browser accessing a shopping website.
  - **Server:** Web server (e.g., Apache or Nginx) and application server (e.g., Flask, Spring Boot).

- **Advantages:**
  - Platform-independent (any device with a browser can be a client).
  - Easy to deploy updates on the server-side.

- **Disadvantages:**
  - Dependent on network connectivity.
  - Performance may vary based on browser compatibility.

3.6 Cloud-Based Client/Server Implementation

Description: Cloud services host server-side components, offering flexible and scalable resources.

- **Example:**
  - **Client:** A mobile or desktop app.
  - **Server:** Cloud services like AWS, Azure, or Google Cloud.
- **Advantages:**
  - High scalability and availability.
  - Reduced infrastructure cost for clients.
- **Disadvantages:**
  - Dependency on third-party cloud providers.
  - Data privacy and security concerns.

3.7 RESTful API Client/Server Implementation

Description: Uses RESTful APIs to allow clients to interact with servers via standard HTTP methods like GET, POST, PUT, and DELETE.

- **Example:**
  - **Client:** Mobile app requesting weather data.
  - **Server:** REST API on a server hosting weather data.
- **Advantages:**
  - Language-agnostic and platform-independent.
  - Easily integrates with other systems.
- **Disadvantages:**
  - May require more bandwidth due to HTTP overhead.
  - Lacks built-in state management.

3.8 Key Considerations

- **Scalability:** Choose the appropriate architecture based on the expected number of users and data size.
- **Security:** Implement secure communication (e.g., HTTPS, firewalls) and authentication mechanisms.
- **Performance:** Use load balancers, caching layers, and optimized database queries to improve response times.
- **Cost:** Evaluate infrastructure costs, including hardware, cloud services, and maintenance.
- **Maintainability:** Opt for modular designs to facilitate future updates and debugging.

These implementations cater to various requirements, ranging from small, localized systems to large-scale distributed applications.

4 Multiple Choice Questions

1. What is the primary purpose of a Network Interface Card (NIC)?
   (a) Store application files
   (b) Enable network connectivity
   (c) Process server requests
   (d) Manage client workflows
2. Which of the following is NOT a server-side hardware component?
   (a) RAID configuration
   (b) CPU
   (c) Input/Output devices
   (d) Cooling systems
3. What is the role of load balancers in a client/server architecture?
   (a) Handle concurrent client requests
   (b) Distribute client requests across multiple servers
   (c) Cache frequently accessed data
   (d) Provide backup power in case of failure
4. Which hardware is commonly used for redundancy and fault tolerance?
   (a) SSDs
   (b) RAID configurations
   (c) NICs
   (d) UPS
5. Which network device acts as a gateway between local networks and the internet?
   (a) Switch
   (b) Router
   (c) Proxy server
   (d) Load balancer
6. What is the primary function of a DBMS in server-side components?
   (a) Handle client requests
   (b) Execute business logic
   (c) Store and retrieve data
   (d) Process API calls
7. What protocol is commonly used for secure communication between a client and a server?
   (a) HTTP
   (b) HTTPS
   (c) FTP
   (d) SMTP

8. Which client-side component is responsible for data preprocessing and input validation?

   (a) Communication module
   (b) Middleware
   (c) Application logic (partial)
   (d) Protocol stack

9. What does a middleware component facilitate in a client/server system?

   (a) Direct data communication between clients
   (b) User interaction with the server
   (c) Communication and data exchange between components
   (d) Network security

10. Which shared component maintains the context of interactions between a client and server?

    (a) API
    (b) Session management
    (c) Middleware
    (d) Communication module

11. In a two-tier client/server implementation, the server handles:

    (a) Only the application logic
    (b) Application logic and the database
    (c) Load balancing
    (d) Middleware processes

12. Which implementation is most suitable for small-scale applications?

    (a) Two-tier
    (b) Three-tier
    (c) Multi-tier
    (d) Cloud-based

13. What is a key advantage of a three-tier architecture?

    (a) Simplicity in design
    (b) Scalability and maintainability
    (c) Reduced hardware cost
    (d) Minimal implementation complexity

14. Which architecture adds layers such as caching and microservices?

    (a) Two-tier
    (b) Three-tier
    (c) Multi-tier
    (d) Peer-to-peer

15. What is the primary disadvantage of peer-to-peer (P2P) hybrid models?

    (a) Reduced availability
    (b) High dependency on central servers
    (c) Security and data consistency challenges
    (d) Platform dependency

16. Which communication protocol is commonly used in RESTful API implementations?

    (a) FTP
    (b) SMTP
    (c) HTTP
    (d) SNMP

17. What is a key consideration when selecting a client/server implementation?

    (a) User interface design
    (b) Session management
    (c) Scalability requirements
    (d) Middleware options

18. Which architecture is highly scalable and uses third-party services?

    (a) Cloud-based
    (b) Two-tier
    (c) Web-based
    (d) Peer-to-peer

19. What type of implementation is best for platform-independent clients?

    (a) Peer-to-peer
    (b) RESTful API
    (c) Web-based
    (d) Multi-tier

20. Which type of server is responsible for managing HTTP/HTTPS traffic?

    (a) Web server
    (b) Application server
    (c) Database server
    (d) Proxy server

21. What is the first step in the client/server workflow?

    (a) Server-side processing
    (b) Response transmission
    (c) Client sends a request
    (d) Middleware exchanges data

22. What ensures optimal operating temperatures in servers?

    (a) RAID configurations
    (b) UPS systems
    (c) Cooling systems
    (d) Load balancers

23. Which layer in a three-tier architecture is responsible for user interaction?

    (a) Application layer
    (b) Presentation layer
    (c) Database layer
    (d) Middleware

24. In RESTful API communication, which method is used to retrieve data?

    (a) POST
    (b) PUT
    (c) DELETE
    (d) GET

25. What hardware is used to organize and house server hardware in data centers?

    (a) Switches
    (b) Racks and enclosures
    (c) Routers
    (d) Proxy servers