Learn System Design
Progress 0/121 chapters

Search topics...

Databases & Storage 0/12
Database Scaling Techniques 0/8
Caching 0/6
Networking 0/7
APIs 0/11
Asynchronous Communications 0/3
Tradeoffs 0/9
Distributed System Concepts 0/10
Microservices 0/5
Big Data Processing 0/5
Architectural Patterns 0/6
9 Software Architecture Patterns
Client-Server Architecture
Microservices Architecture
Serverless Architecture
Event-Driven Architecture
Peer-to-Peer (P2P) Architecture
Observability 0/3
Security 0/5
Interview Tips 0/3
Interview Questions 0/17

Client-Server Architecture

Ashish Pratap Singh

4 min read

Get Premium
Subscribe to unlock full access to all premium content
Subscribe Now

Reading Progress 0%

On this page

1. What is Client-Server Architecture?
2. How Client-Server Architecture Works
3. Types of Client-Server Architectures
3.1 Two-Tier Architecture
3.2 Three-Tier Architecture
3.3 N-Tier Architecture
4. Advantages of Client-Server Architecture
5. Challenges and Considerations
6. Real-World Applications
7. Conclusion

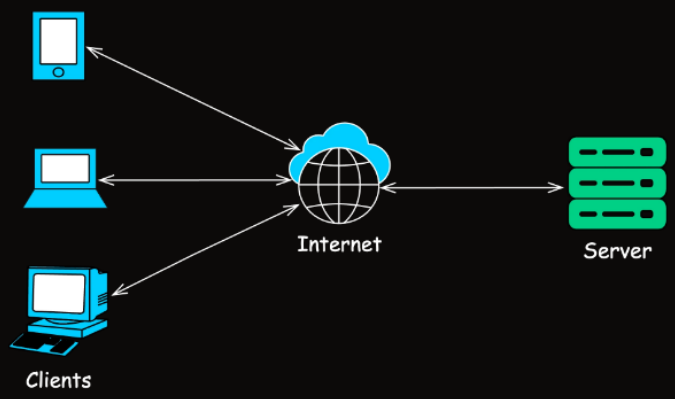
Whether you're browsing the web, sending an email, or using a cloud-based application, you're interacting with a system designed around the client-server model.

In this blog post, we'll explore what client-server architecture is, how it works, the different types of client-server models, and why it's so widely used.

1. What is Client-Server Architecture?

Client-server architecture is a computing model in which multiple clients (users or devices) interact with a centralized server to access data, resources, or services.

The server is responsible for managing resources and serving multiple clients simultaneously, while the client initiates requests and processes the responses from the server.



Key Components:

- **Client**: The client is typically a device or application that initiates a request to the server. This could be a web browser, a mobile app, or a desktop application.
- **Server**: The server is a powerful computer or software application that processes requests from clients, manages resources, and delivers the requested services or data.
- **Network**: The communication medium that allows clients and servers to exchange data.

> **Example:** When you open a web browser and visit a website

Patent Owner, CleveMed - Ex. 2024, p. 1

Learn System Design
Progress 0/121 chapters
Search topics...
Databases & Storage 0/12
Database Scaling Techniques 0/8
Caching 0/6
Networking 0/7
APIs 0/11
Asynchronous Communications 0/3
Tradeoffs 0/9
Distributed System Concepts 0/10
Microservices 0/5
Big Data Processing 0/5
Architectural Patterns 0/6
9 Software Architecture Patterns
Client-Server Architecture
Microservices Architecture
Serverless Architecture
Event-Driven Architecture
Peer-to-Peer (P2P) Architecture
Observability 0/3
Security 0/5
Interview Tips 0/3
Interview Questions 0/17

servers to exchange data.

> **Example:** When you open a web browser and visit a website, your browser (client) sends a request to a web server. The web server processes the request, retrieves the necessary data (such as HTML, CSS, images), and sends it back to your browser, which then renders the webpage.

2. How Client-Server Architecture Works

The client-server model operates on a request-response mechanism, where clients send requests to the server, and the server responds with the requested information or service.

Here's a step-by-step breakdown of how it works:

1. **Client Request:** The client sends a request to the server. This request could be for a webpage, a database query, a file download, or any other service the server offers.
2. **Request Processing:** The server receives the request and processes it. This might involve retrieving data from a database, executing some business logic, or generating a dynamic response.
3. **Server Response:** Once the server processes the request, it sends a response back to the client. The response typically contains the data or service the client requested.
4. **Client Processing:** The client receives the response and processes it. For example, a web browser would render a webpage, or a mobile app might display the requested data to the user.

> **Example:** When you log into an online banking application, your credentials are sent from the client (your device) to the server. The server validates your credentials, retrieves your account information, and sends it back to the client for display.

3. Types of Client-Server Architectures

3.1 Two-Tier Architecture

In a two-tier architecture, the client directly interacts with the server. The server typically handles both the application logic and

Get Premium
Subscribe to unlock full access to all premium content
Subscribe Now

Reading Progress 30%

On this page

1. What is Client-Server Architecture?
2. How Client-Server Architecture Works
3. Types of Client-Server Architectures
3.1 Two-Tier Architecture
3.2 Three-Tier Architecture
3.3 N-Tier Architecture
4. Advantages of Client-Server Architecture
5. Challenges and Considerations
6. Real-World Applications
7. Conclusion

Patent Owner, CleveMed - Ex. 2024, p. 2

Learn System Design
Progress 0/121 chapters
Search topics...
Databases & Storage 0/12
Database Scaling Techniques 0/8
Caching 0/6
Networking 0/7
APIs 0/11
Asynchronous Communications 0/3
Tradeoffs 0/9
Distributed System Concepts 0/10
Microservices 0/5
Big Data Processing 0/5
Architectural Patterns 0/6
9 Software Architecture Patterns
Client-Server Architecture
Microservices Architecture
Serverless Architecture
Event-Driven Architecture
Peer-to-Peer (P2P) Architecture
Observability 0/3
Security 0/5
Interview Tips 0/3
Interview Questions 0/17

In a two-tier architecture, the client directly interacts with the server. The server typically handles both the application logic and data management.

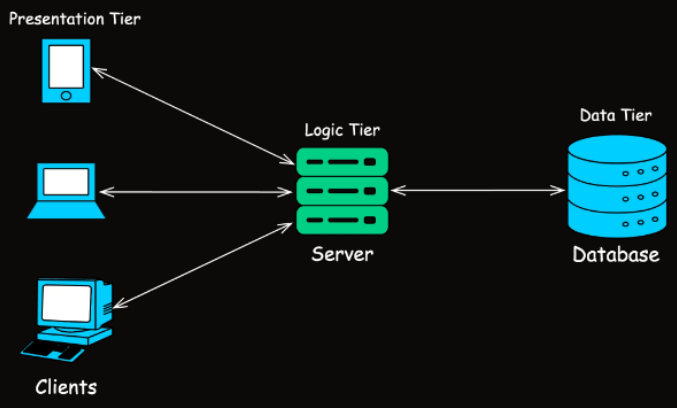
This model is simple but can become inefficient as the number of clients increases, leading to performance bottlenecks.

- **Client:** Handles the presentation layer (UI) and communicates directly with the server.
- **Server:** Manages the application logic and database.

> **Example:** A desktop application that connects directly to a database server to retrieve and display data is an example of a two-tier architecture.

3.2 Three-Tier Architecture

Three-tier architecture introduces an additional layer, known as the application or business logic layer, between the client and the data server. This separation of concerns improves scalability, maintainability, and security.



- **Client:** Handles the presentation layer.
- **Application Server:** Manages business logic and processes client requests.
- **Database Server:** Handles data storage and management.

> **Example:** A web application where the client (browser) interacts with a web server (application server) that then queries a database server to retrieve data follows a three-tier architecture.

3.3 N-Tier Architecture

N-tier architecture is an extension of the three-tier model, where additional layers can be introduced for specific functions such as

Get Premium
Subscribe to unlock full access to all premium content
Subscribe Now

On this page

1. What is Client-Server Architecture?
2. How Client-Server Architecture Works
3. Types of Client-Server Architectures
3.1 Two-Tier Architecture
3.2 Three-Tier Architecture
3.3 N-Tier Architecture
4. Advantages of Client-Server Architecture
5. Challenges and Considerations
6. Real-World Applications
7. Conclusion

Patent Owner, CleveMed - Ex. 2024, p. 3

Learn System Design
Progress 0/121 chapters
Search topics...
Databases & Storage 0/12
Database Scaling Techniques 0/8
Caching 0/6
Networking 0/7
APIs 0/11
Asynchronous Communications 0/3
Tradeoffs 0/9
Distributed System Concepts 0/10
Microservices 0/5
Big Data Processing 0/5
Architectural Patterns 0/6
9 Software Architecture Patterns
Client-Server Architecture
Microservices Architecture
Serverless Architecture
Event-Driven Architecture
Peer-to-Peer (P2P) Architecture
Observability 0/3
Security 0/5
Interview Tips 0/3
Interview Questions 0/17

3.3 N-Tier Architecture

N-tier architecture is an extension of the three-tier model, where additional layers can be introduced for specific functions such as caching, load balancing, or security.

This model is highly scalable and flexible, making it suitable for complex, large-scale applications.

- **Client:** User interface or front-end application.
- **Presentation Layer:** Manages the user interface and presentation logic.
- **Application Layer:** Handles business logic and rules.
- **Data Layer:** Manages data access and storage.
- **Additional Layers (optional):** For caching, logging, security, etc.

> **Example:** A large e-commerce platform with separate services for user authentication, product catalog, shopping cart, and payment processing might use an N-tier architecture.

4. Advantages of Client-Server Architecture

The client-server model offers several advantages, which is why it's so widely used:

- **Centralized Management:** Servers can centrally manage resources, security, and data, making it easier to maintain and update the system.
- **Scalability:** Servers can be scaled vertically (upgrading hardware) or horizontally (adding more servers) to handle increased load and traffic.
- **Resource Sharing:** Multiple clients can share resources like databases, files, and applications hosted on the server.
- **Security:** Centralized control over data and resources allows for better security practices, such as authentication, encryption, and access control.

5. Challenges and Considerations

Despite its advantages, client-server architecture also has some challenges:

- **Single Point of Failure:** If the server goes down, all clients are affected. Redundancy and failover mechanisms are needed to mitigate this risk.
- **Performance Bottlenecks:** A server can become a

Get Premium
Subscribe to unlock full access to all premium content
Subscribe Now

Reading Progress 85%

On this page

1. What is Client-Server Architecture?
2. How Client-Server Architecture Works
3. Types of Client-Server Architectures
3.1 Two-Tier Architecture
3.2 Three-Tier Architecture
3.3 N-Tier Architecture
4. Advantages of Client-Server Architecture
5. Challenges and Considerations
6. Real-World Applications
7. Conclusion

Patent Owner, CleveMed - Ex. 2024, p. 4

Learn System Design
Progress 0/121 chapters
Search topics...
Databases & Storage 0/12
Database Scaling Techniques 0/8
Caching 0/6
Networking 0/7
APIs 0/11
Asynchronous Communications 0/3
Tradeoffs 0/9
Distributed System Concepts 0/10
Microservices 0/5
Big Data Processing 0/5
Architectural Patterns 0/6
9 Software Architecture Patterns
Client-Server Architecture
Microservices Architecture
Serverless Architecture
Event-Driven Architecture
Peer-to-Peer (P2P) Architecture
Observability 0/3
Security 0/5
Interview Tips 0/3
Interview Questions 0/17

- **Single Point of Failure:** If the server goes down, all clients are affected. Redundancy and failover mechanisms are needed to mitigate this risk.
- **Performance Bottlenecks:** A server can become a performance bottleneck if it's handling too many client requests simultaneously. Load balancing and optimization are required to maintain performance.
- **Complexity:** As systems grow, managing and scaling a client-server architecture can become complex, requiring advanced infrastructure and expertise.

6. Real-World Applications

Client-server architecture is ubiquitous in modern computing.

Here are a few examples of real-world applications:

- **Web Browsing:** When you visit a website, your web browser (client) communicates with the web server to fetch and display the webpage.
- **Email Services:** Email clients like Outlook or Gmail connect to mail servers to send and receive emails.
- **Online Banking:** Banking applications use client-server architecture to process transactions, display account balances, and manage user data securely.
- **Cloud Computing:** Cloud services like AWS or Google Cloud provide resources (servers, storage, databases) to clients on demand, using a client-server model.

7. Conclusion

Client-server architecture is a foundational concept in computer science that has enabled the development of scalable, reliable, and efficient systems.

By understanding its components, types, and use cases, you can design systems that meet the needs of modern applications.

Whether you're building a simple two-tier application or a complex n-tier enterprise system, client-server architecture provides the framework to create robust and responsive software that can scale with your users and requirements.

Mark as Complete Star

← Previous: 9 Software Architecture Patterns
Next: Microservices Architecture →

Get Premium
Subscribe to unlock full access to all premium content
Subscribe Now

Reading Progress 100%

On this page

1. What is Client-Server Architecture?
2. How Client-Server Architecture Works
3. Types of Client-Server Architectures
3.1 Two-Tier Architecture
3.2 Three-Tier Architecture
3.3 N-Tier Architecture
4. Advantages of Client-Server Architecture
5. Challenges and Considerations
6. Real-World Applications
7. Conclusion