

Summary

Learn essential system design concepts to scale applications and excel in interviews, covering networking, APIs, databases, and more.

Highlights

- 📌 **Vertical vs. Horizontal Scaling:** Vertical scaling adds resources to one server, while horizontal scaling adds more servers for better redundancy.
- 🖥️ **Load Balancers:** These distribute incoming requests to servers, ensuring no single server is overwhelmed.
- 🌐 **Content Delivery Networks (CDNs):** CDNs cache static content globally, reducing load on origin servers and speeding access for users.
- 📄 **API Patterns:** Understanding REST, GraphQL, and gRPC is crucial in designing efficient APIs for application communication.
- 🗄️ **Database Types:** SQL databases offer ACID compliance, while NoSQL databases prioritize flexibility and scalability.
- 🔄 **Replication and Sharding:** Techniques for scaling databases by creating read replicas or dividing data across multiple servers.
- 📧 **Message Queues:** Introduced for decoupling app components, they help manage high data loads efficiently.

Key Insights

- 🔄 **Scaling Strategies:** Horizontal scaling is more powerful than vertical scaling, allowing for redundancy and fault tolerance. This is essential for growing applications needing reliability.
- 🖥️ **Load Balancing Techniques:** Effective load balancing is vital for distributing traffic and maintaining performance, especially in large-scale systems.
- 🌐 **Importance of CDNs:** By offloading static assets, CDNs enhance user experience and reduce latency, crucial for global applications.
- 📄 **API Efficiency:** Choosing the right API pattern affects performance and ease of use, with GraphQL offering more efficiency than REST in many cases.
- 🗄️ **Database Management:** Understanding the trade-offs between SQL and NoSQL databases helps in selecting the appropriate data storage solution based on application needs.
- 🔄 **Replication and Sharding Complexity:** These strategies can improve database performance, but they introduce challenges in maintaining consistency and managing data distribution.
- 📧 **Decoupling with Message Queues:** They provide scalability and resilience in applications by allowing asynchronous processing, crucial for handling high traffic and data loads.

