System Design Fundamentals: Load Balancers, Databases, and Caching (Practical View)

Subtitle: Learn the core building blocks of scalable systems and how they work together in real production environments.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Most developers can build applications that “work”.

But very few understand:
• how those applications survive real traffic
• how systems stay fast under load
• what happens when millions of users arrive at once

This is where system design starts.

System design is not about coding.

It is about:
• architecture decisions
• performance control
• scalability planning

This PDF explains the 3 core pillars:
• Load Balancers
• Databases
• Caching

Chapter 1: What System Design Actually Means

System design is:

How different components of a system work together to handle users efficiently

It focuses on:

• speed
• reliability
• scalability
• fault tolerance

Chapter 2: Load Balancer (Traffic Distributor)

A load balancer is the entry point of large systems.

Problem it solves:

If one server gets too many requests:
• it crashes

Solution:

Instead of one server:

• multiple servers handle traffic
• load balancer distributes requests

Simple Flow:

User → Load Balancer → Server A / Server B / Server C

Why this matters:

Without it:
• system breaks under traffic spikes

Types of Load Balancing (Basic Understanding)

1. Round Robin

Requests are distributed one by one

2. Least Connections

Sends traffic to least busy server

3. IP-based routing

Same user goes to same server sometimes

Chapter 3: Database (Data Storage Engine)

Database is where all data lives:
• users
• posts

• messages
• transactions

Beginner misconception:

Database = simple storage

Reality:
Database is:
• highly optimized system
• designed for fast queries
• structured for scale

Main problem at scale:

Too many requests = slow database

Solutions used in real systems:

1. Indexing

Speeds up search queries

2. Replication

Copies data to multiple servers

3. Sharding

Splits database into parts

Simple idea:

Instead of 1 huge database:
• multiple smaller databases

Chapter 4: Caching (Speed Layer)

Caching is one of the most important concepts in system design.

Problem it solves:

Repeated requests slow down systems

Example:

If 1 million users request same homepage data:
• database gets overloaded

Solution:

Store frequently used data in cache:
• memory-based storage

Result:

• faster response
• reduced database load
• better performance

Where caching is used:

• homepage data
• user profiles
• search results

Chapter 5: How These 3 Work Together

Real systems combine all three:

Flow:

User → Load Balancer → Server → Cache → Database

Step-by-step:

1. user sends request
2. load balancer routes it
3. server checks cache first
4. if not found → database is used
5. response sent back

Chapter 6: Why Systems Fail Without This Structure

Without proper design:

• one server gets overloaded
• database becomes slow
• response time increases
• users experience lag or crash

Chapter 7: Real Engineering Insight

Big companies do NOT rely on:
• single server logic

They rely on:
• distributed architecture

Chapter 8: Beginner vs Real Engineer Thinking

Beginner:

• “My app works locally”

Engineer:

• “Will this work with 10 million users?”

Chapter 9: Why These Concepts Matter

These are not advanced topics.

They are:
• foundation of real systems

If you understand this:

You can understand:
• any scalable application
• any backend system
• any large platform architecture

Chapter 10: Common Misunderstanding

Beginners think:
• system design = memorizing diagrams

Reality:
• system design = understanding tradeoffs

Example Tradeoff:

Cache:
• faster
• but may be outdated

Database replication:
• reliable
• but complex

Chapter 11: Key Engineering Principles

1. Nothing scales by default

Everything must be designed for scale

2. Every system has bottlenecks

You must identify and fix them

3. Tradeoffs always exist

No perfect solution exists

Key Takeaways

• Load balancer distributes traffic across servers
• Database stores structured data but needs scaling strategies
• Caching improves speed and reduces load
• Real systems use all three together
• System design is about structure, not code
• Every architecture has tradeoffs
• Scalability must be planned, not assumed
• Engineering thinking is about system behavior

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app