How to scale a website to support millions of users?

ALEX XU

MAR 29, 2022

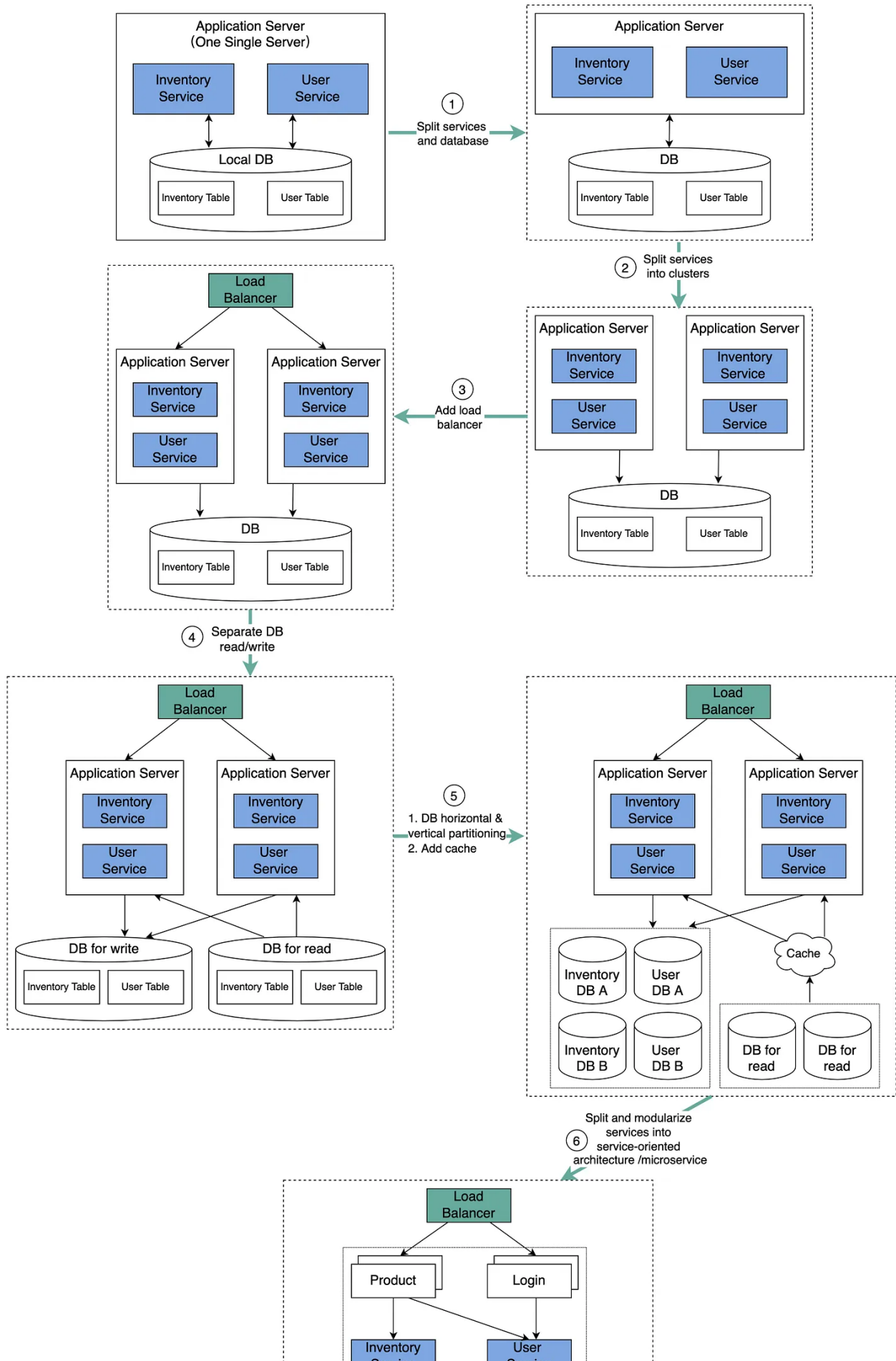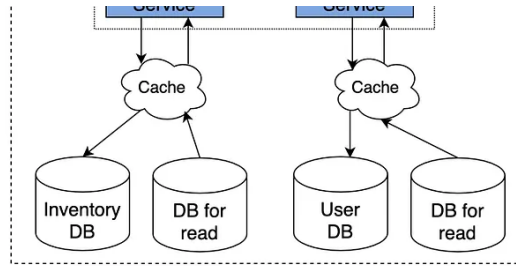♡ 54   💬 3   ↻   Share   •••

How to scale a website to support millions of users? We will explain this step-by-step.

The diagram below illustrates the evolution of a simplified eCommerce website. It goes from a monolithic design on one single server, to a service-oriented/microservice architecture.

How to Scale a Website Step-by-Step?

ByteByteGo

Application Server (One Single Server)
- Inventory Service
- User Service
- Local DB
  - Inventory Table
  - User Table

① Split services and database

Application Server
- Inventory Service
- User Service
- DB
  - Inventory Table
  - User Table

② Split services into clusters

③ Add load balancer

Load Balancer
- Application Server
  - Inventory Service
  - User Service
- Application Server
  - Inventory Service
  - User Service
- DB
  - Inventory Table
  - User Table

Application Server
- Inventory Service
- User Service

Application Server
- Inventory Service
- User Service

- DB
  - Inventory Table
  - User Table

④ Separate DB read/write

Load Balancer
- Application Server
  - Inventory Service
  - User Service
- Application Server
  - Inventory Service
  - User Service
- DB for write
  - Inventory Table
  - User Table
- DB for read
  - Inventory Table
  - User Table

⑤ 1. DB horizontal & vertical partitioning. 2. Add cache

Load Balancer
- Application Server
  - Inventory Service
  - User Service
- Application Server
  - Inventory Service
  - User Service
- Inventory DB A
- User DB A
- Inventory DB B
- User DB B
- Cache
- DB for read
- DB for read

⑥ Split and modularize services into service-oriented architecture /microservice

Load Balancer
- Product
- Login
- Inventory Service
- User Service

Suppose we have two services: inventory service (handles product descriptions and inventory management) and user service (handles user information, registration, login, etc.).

Step 1 - With the growth of the user base, one single application server cannot handle the traffic anymore. We put the application server and the database server into two separate servers.

Step 2 - The business continues to grow, and a single application server is no longer enough. So we deploy a cluster of application servers.

Step 3 - Now the incoming requests have to be routed to multiple application servers, how can we ensure each application server gets an even load? The load balancer handles this nicely.

Step 4 - With the business continuing to grow, the database might become the bottleneck. To mitigate this, we separate reads and writes in a way that frequent read queries go to read replicas. With this setup, the throughput for the database writes can be greatly increased.

Step 5 - Suppose the business continues to grow. One single database cannot handle the load on both the inventory table and user table. We have a few options:

1. Vertical partition. Adding more power (CPU, RAM, etc.) to the database server. It has a hard limit.
2. Horizontal partition by adding more database servers.
3. Adding a caching layer to offload read requests.

Step 6 - Now we can modularize the functions into different services. The architecture becomes service-oriented / microservice.

Question: what else do we need to support an e-commerce website at Amazon's scale?

If you enjoyed this post, you might like our system design interview books as well.

SDI-vol1: https://amzn.to/3tK0qQn

SDI-vol2: https://amzn.to/37ZisW9

---

54 Likes

3 Comments

Write a comment...

Denis Liger Apr 27, 2022

step 7 is event driven architecture with message queue

♡ LIKE (1)   💬 REPLY   ⬆ SHARE   •••

> **1 reply**

Manish Shrestha Oct 12

Are you trying to explain about vertical scaling or vertical partitioning ?

"1. Vertical partition. Adding more power (CPU, RAM, etc.) to the database server. It has a hard limit."

♡ LIKE   💬 REPLY   ⬆ SHARE   •••

1 more comment...

10/14/23, 11:45 AM   How to scale a website to support millions of users?

5/5