# Web50 Section 9

Scalability

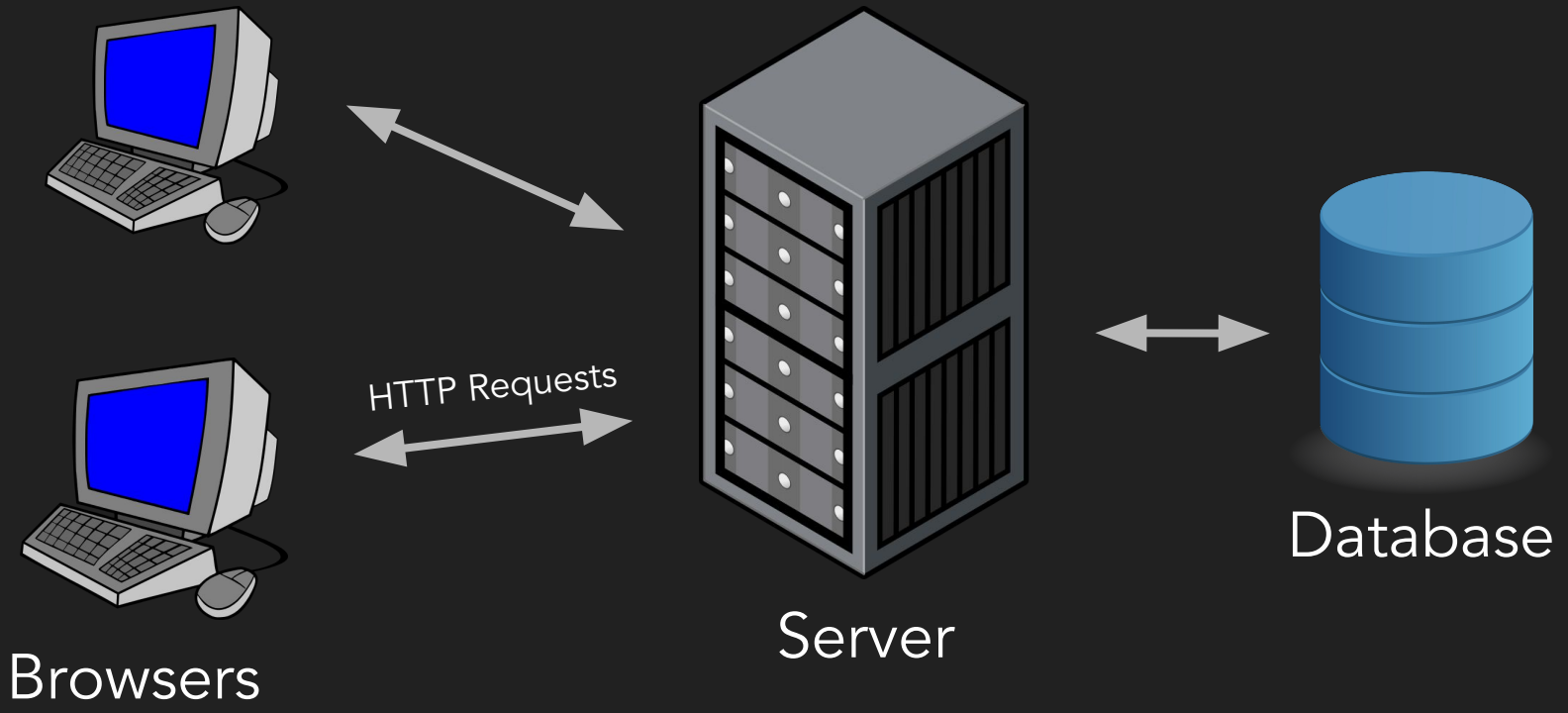
# Agenda

- Notes
- Scaling Basics
- Examples of Scalability
- Questions

# Notes

- Final Project released: project proposal due **Friday April 19th**
- This is the second to last section

# Servers: A review



# Scalability

**Benchmarking**: Determine out how many requests a server can handle

- Load test, stress test

**Vertical Scaling**: Make the server larger with more processing power

- But there is a limit

**Horizontal Scaling**: Add more servers

- **Load balancer**: hardware used to ensure the user goes to the right server
- Load balancing methods:
  - Random choice
  - Round robin
  - Fewest connections
- Use: **session-aware load balancing** to ensure each user uses the same server

# Scalability

**Autoscaling**: Add more servers depending on demand

- Also protects against servers going off-line
- Common feature of **cloud computing:**
    - Servers are located elsewhere and rented out

**Database partitioning**: Split database into smaller tables to make data access faster

- Vertical partitioning: Decrease number of columns
- Horizontal partitioning: Decrease number of rows
- Database shard: store a different database on a different server

# Scalability

**Database replication**: Allows for multiple access points but need to insure that data is the same across all

- Single primary replication:
    - One database treated as the primary database, with write privileges
    - Other databases are read only
- Multi-primary replication:
    - Can read and write across any database

**Caching**: Store data temporarily so you don't need to call repeat requests

- client-side to store page data
- sever-side to store the result of database queries

# Amazon.com

- How would you **benchmark** Amazon.com?
- When would Amazon.com need to vertically scale?
  - How do you actually “vertically scale”? Where does the new “larger server” come from?
- When would Amazon.com horizontally scale?
- Amazon.com usually has people spend an equal amount of time on their website. Each user spends ~5 min from searching to checkout.
- Which load balancing algorithm makes sense?
  - Round-robin, random, direct?

# Netflix

- What load balancing algorithm would Netflix want to use? Some users binge watch 34 hours of TV, while some spend only an hour on the site.
- Netflix's recommendation algorithm is really complicated. When it loads shows for you to watch, how would it want to cache them?
  - Client-side caching?
  - Server-side caching?
- When would you want to reset the cache?

# Gmail Database

- Come up with a scheme to vertically scale the Gmail database
  - Think of users, emails?
- Come up with a scheme to horizontally scale the Gmail database
  - Different users emails?
  - Emails by time?
- When would a slower query time be okay? For what kind of emails?
- What is **sharding**?
- What is a single-primary replication system versus a multi-primary replication system?
- Does Gmail use a lot of writes or reads or both?

# Q&A on Scalability

Questions?

# Q&A on Final Project

- Potential topics?
- Scope?