Cloud Computing

Deploying your application

- You just wrote a website or backend, and it works great on your laptop
- How do you get it to the public?

Deploying your application

- You just wrote a website or backend, and it works great on your laptop
- How do you get it to the public?
- Old solution: buy a server. Run your application there.
    - What are some problems with this?

Deploying your application

- You just wrote a website or backend, and it works great on your laptop
- How do you get it to the public?
- Old solution: buy a server. Run your application there.
  - What are some problems with this? Space, heat, upfront cost, etc
  - What do you do when you have too many users for one server to handle?

Deploying your application

- You just wrote a website or backend, and it works great on your laptop
- How do you get it to the public?
- Old solution: buy a server. Run your application there.
    - What are some problems with this? Space, heat, upfront cost, etc
    - What do you do when you have too many users for one server to handle?
    - Scale *vertically* or *horizontally*: vertical scaling means making your one server more powerful, and horizontal scaling means adding more servers and sharing load between them

Deploying your application

- You just wrote a website or backend, and it works great on your laptop
- How do you get it to the public?
- Old solution: buy a server. Run your application there.
    - What are some problems with this? Space, heat, upfront cost, etc
    - What do you do when you have too many users for one server to handle?
    - Scale *vertically* or *horizontally*: vertical scaling means making your one server more powerful, and horizontal scaling means adding more servers and sharing load between them
    - Problem: this is expensive! And demand is variable!

Why let cloud providers handle scaling?

- Scaling and hardware failure are transparent to your application
    - Servers can fail and you won’t know, and neither will your users!
- Physical hardware and datacenter space is *expensive*, and demand fluctuates
- More time to focus on improving your project!
- There are times you should handle this yourself
    - Strong latency/performance requirements (i.e. HFT)
    - Consistent load, or you already have datacenter space

Cloud Networking - Load Balancing

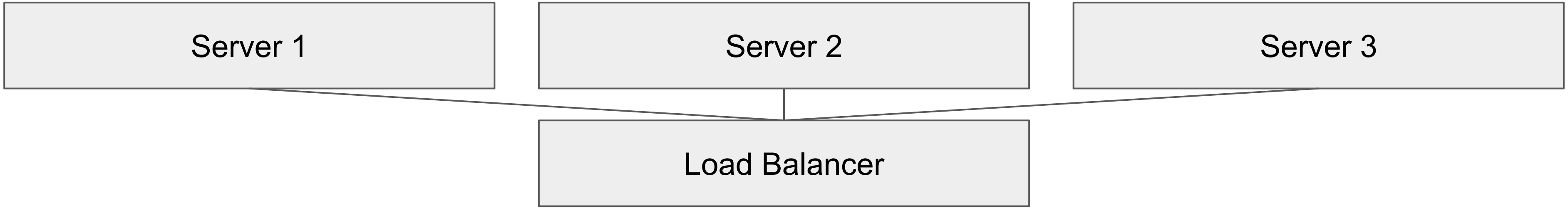
- Problem: we have a bunch of users but only one computer

Cloud Networking - Load Balancing

- Problem: we have a bunch of users but only one computer
- Solution: multiple computers. **But how?**

Cloud Networking - Load Balancing

- Problem: we have a bunch of users but only one computer
- Solution: multiple computers. **But how?**
- A load balancer runs on one computer and sends users to multiple computers “behind” it
    - Load balancer is extremely efficient so it can run on only one (sometimes powerful) computer and serve many users
- Popular load balancers: NGINX, Traefik, Apache, …



“As a Service”

- Offerings are generally in one of X subsets:
    - Infrastructure as a Service
        - Rent a server (+network access/etc) for $x per month
        - Most flexible, generally you need to scale yourself
        - Examples: EC2, DigitalOcean
    - Platform as a Service
        - Rent server(s) + software running on top of them for $x per month
        - Generally handles scaling your software across the servers for you
        - Examples: AWS EKS
    - Software as a Service
    - Functions as a Service

“As a Service”

- Offerings are generally in one of X subsets:
    - Functions as a Service
        - Pay a small price (<1 cent) each time somebody talks to your application’s server
        - No control over hardware, scales automatically
        - Easiest to set up, least versatile
    - Software as a Service
        - Pay for a company to run their software for you (i.e. Trello)

Cloud Networking

- Every computer on a network has a unique IP address
  - Two types: *local* and *public* (public addresses are internet accessible, local are not!)
  - Two versions: IPv4 and IPv6. We ran out of IPv4 addresses so IPv6 is gaining adoption
- *DNS* takes a domain name (like security.azure.com) and converts it to an IP address
  - Question: how do we balance load geographically? We don’t want Netflix users in South America to load their movies all the way from Canada

Cloud Networking

- Every computer on a network has a unique IP address
  - Two types: *local* and *public* (public addresses are internet accessible, local are not!)
  - Two versions: IPv4 and IPv6. We ran out of IPv4 addresses so IPv6 is gaining adoption
- *DNS* takes a domain name (like security.azure.com) and converts it to an IP address
  - Question: how do we balance load geographically? We don’t want Netflix users in South America to load their movies all the way from Canada
  - Answer: anycast! One domain can be resolved to *multiple* IP addresses, and the closest (by some metric) is chosen
  - Question: how can we use the cloud to take advantage of this?

Cloud Networking - Regions

- Large cloud providers operate data centers across the world, and group each area into a region
  - You should put your code close to your users for better response times!
  - Start with one region, and expand as your application grows
  - Note it is harder to autoscale your application across multiple until you are operating at a very large scale
- Regions make your deployment more complicated
  - Each region generally has its own local network, so communication between regions is harder
  - Different prices for different regions, costs to transfer data between regions
  - Load balancing is done on a per-region basis (load is shared across regions via anycast)