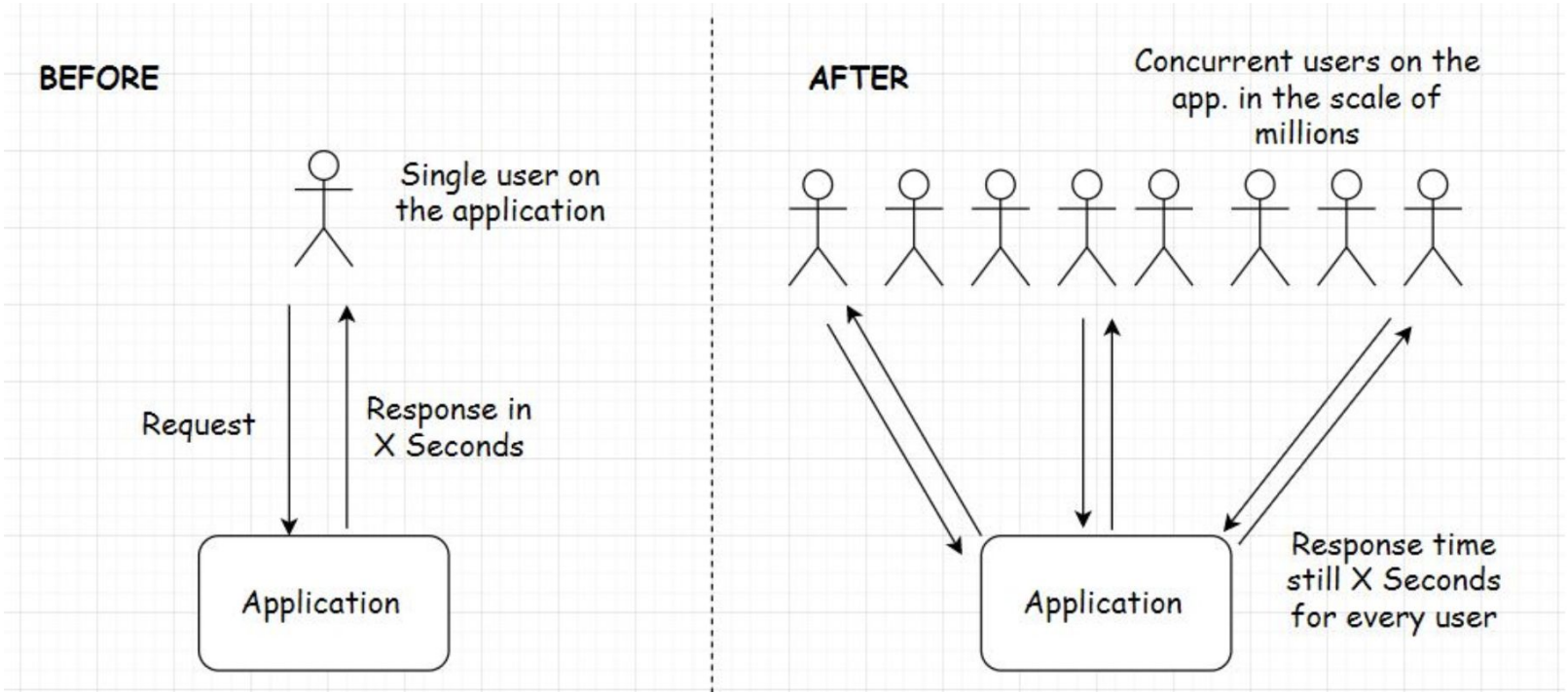
Scalability

...

What is Scalability?

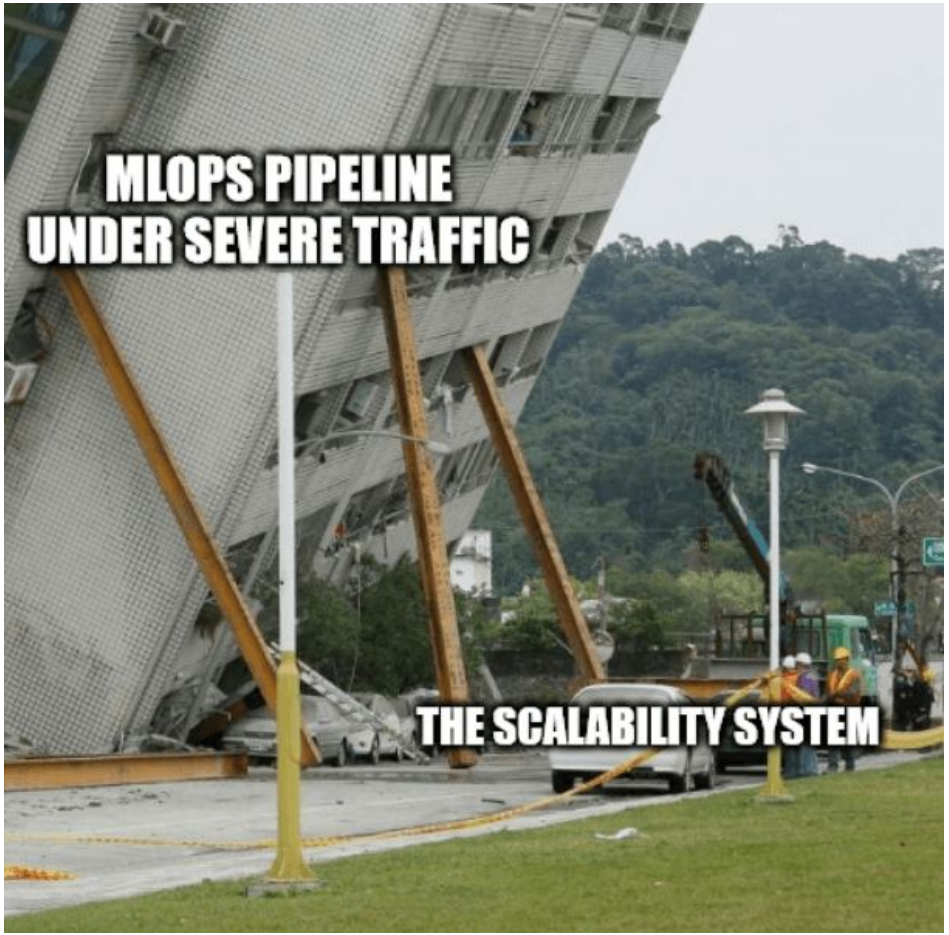
- Scalability: application's ability to handle and withstand increased workload while maintaining performance
- For example, if your app is highly scalable:
  - takes **x** seconds to respond to a user request
  - it should take the same **x** seconds to respond to a million concurrent user requests.
- The app's backend infrastructure should not crumble under an increased load of requests.
  - It should scale well when subjected to a heavy traffic load and maintain the system's latency.

A Highly-Scalable App



What is Scalability?

- Good scalability planning can save you from this:



Latency

- **Latency** is the time a system takes to respond to a user request.
- Minimum latency is what efficient software systems strive for.
    - “Appropriate” latency depends on app and user expectations
    - Producing a PDF vs loading a youtube video vs loading a webpage
- A highly scalable system has the same latency for many requests as it does for few requests

Measuring Latency

- Latency is measured as the time difference between the action that a user takes on the website and the user receiving the system's response in reaction to that action.
- The action can be:
  - clicking a button
  - scrolling down a web page
  - requesting to download a photo

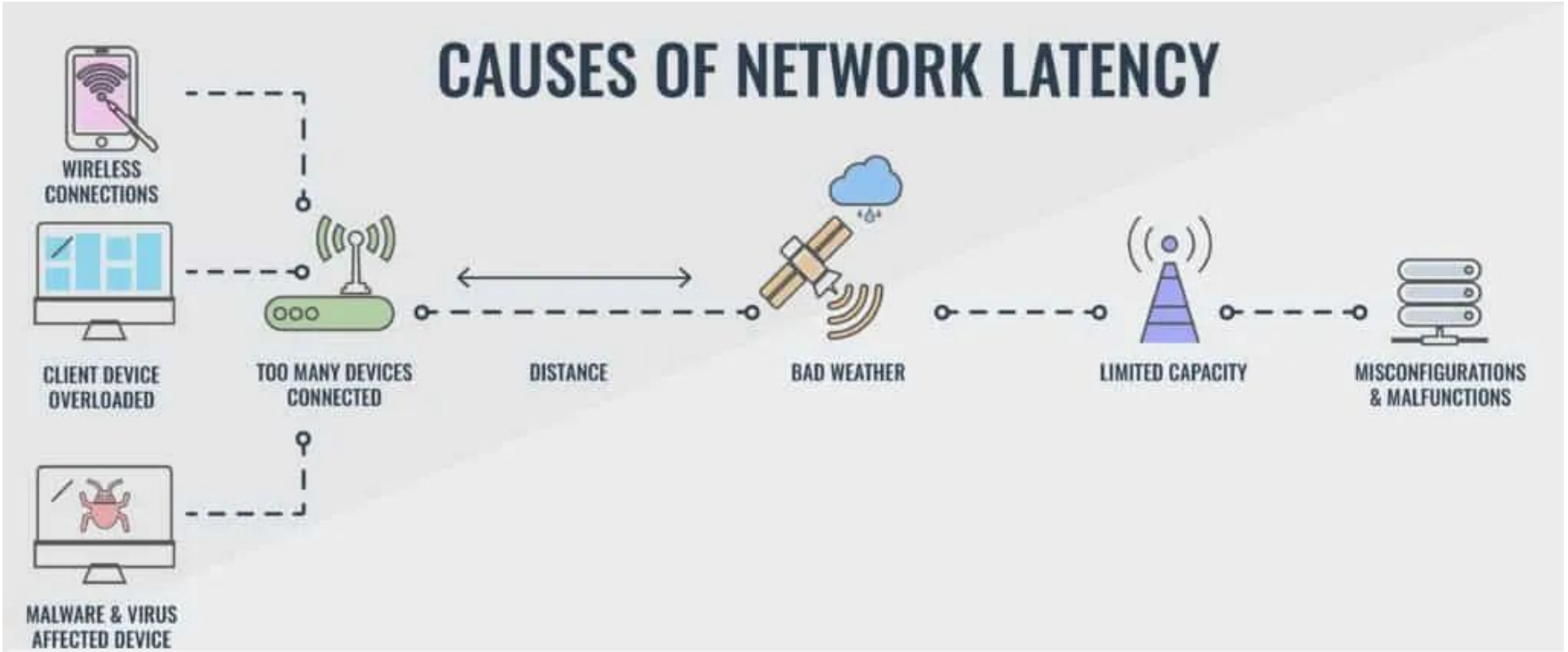
Measuring Latency

- Latency is generally divided into two parts:
  - Network latency
  - Application latency

Network Latency

- Network latency is the time that the **network** takes to send a data packet between client and server, and back again
- The network should be efficient enough to handle the increased traffic load on the website.
- To cut down the network latency, businesses use a CDN (Content Delivery Network) to deploy their servers across the globe as close to the end-user as possible.

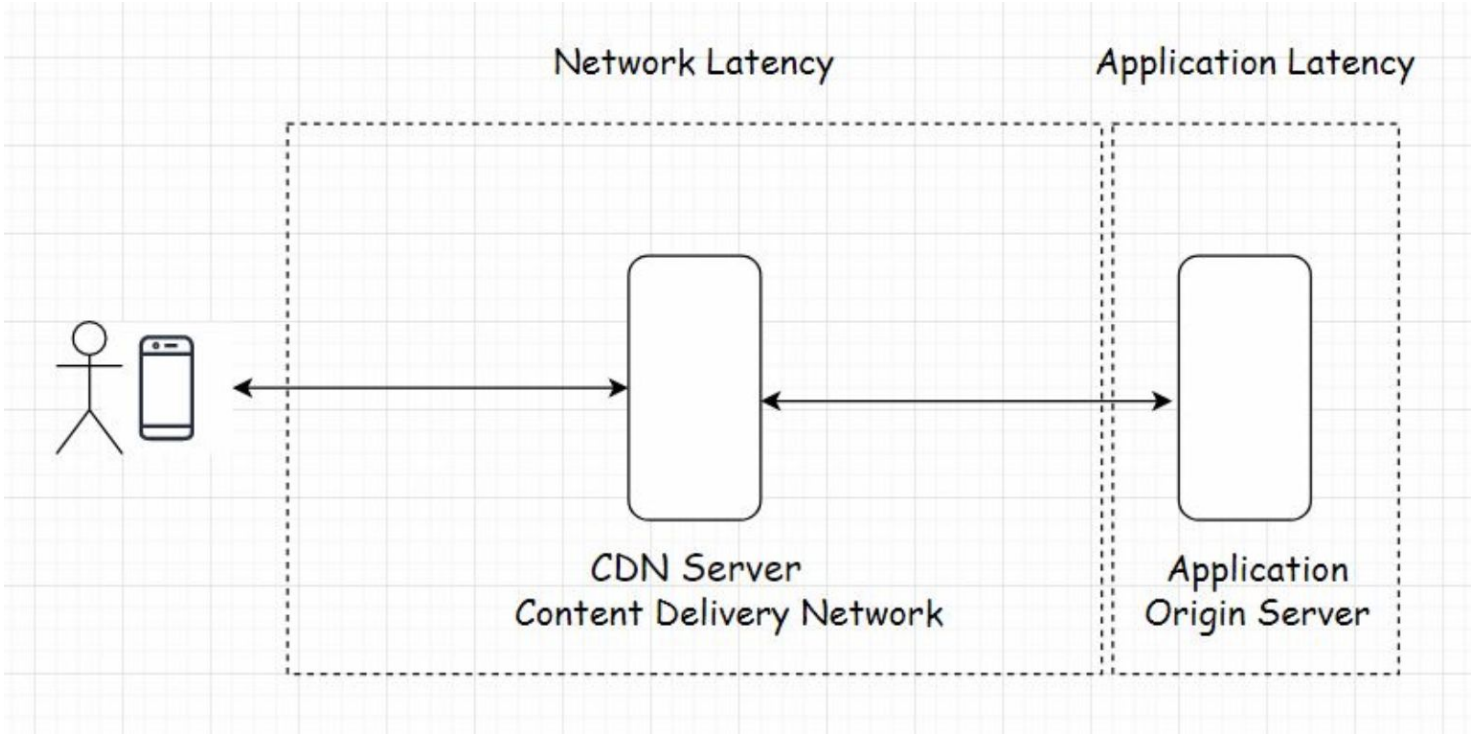
Network Latency



Application Latency

- Application latency refers to the time your application spends processing
  - Time between receiving the request and sending the response
- Your business logic, your internal services, your DB requests, etc
- How to cut down the application latency:
  - Run stress and load tests on the application and scan for the bottlenecks that slow down the system
  - Re-write or reconfigure those bottlenecks

Latency


Network Latency
Application Latency
CDN Server
Content Delivery Network
Application
Origin Server

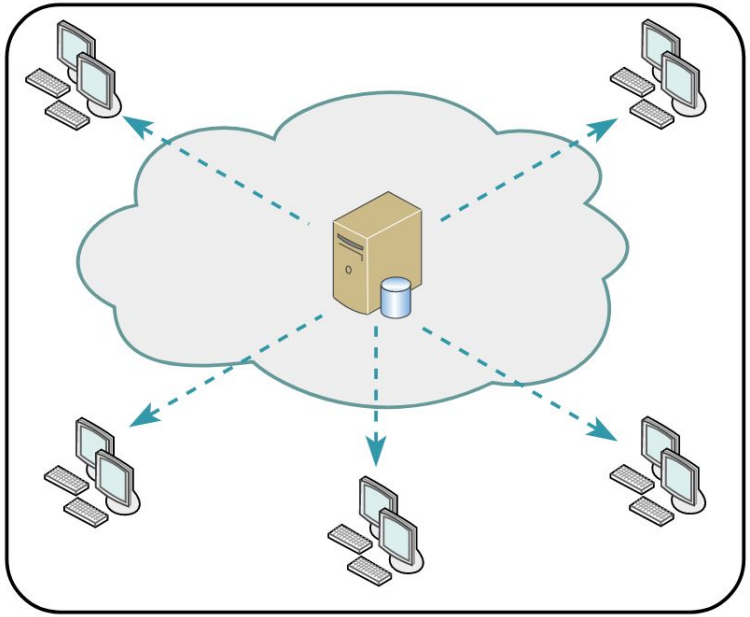
Mini Quiz

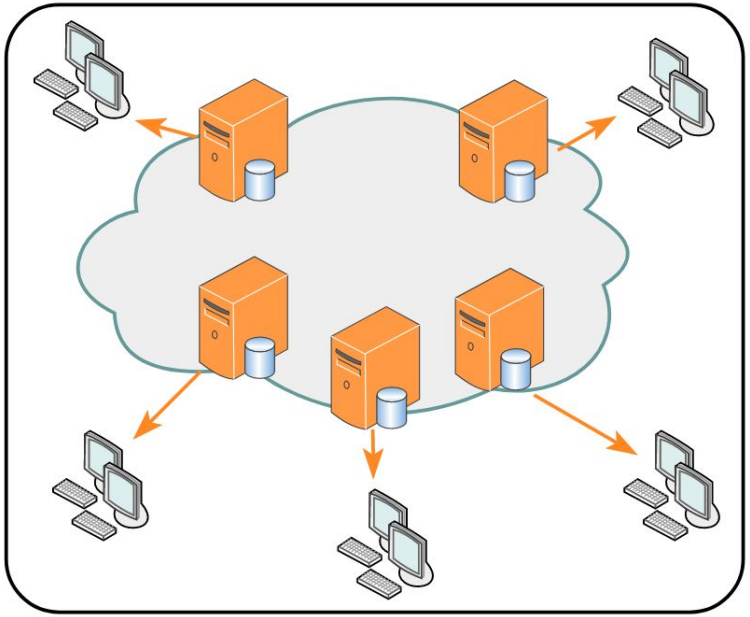
- Which can we control?
  - Application Latency
  - Network Latency

Content Distribution Network (CDN)

- (CDN) is a network of interconnected servers that speeds up webpage loading for data-heavy applications.
- When a user visits a website, data from that website's server has to travel across the internet to reach the user's computer.
- If the user is located far from that server, it will take a long time to load a large file, such as a video or website image.
- Instead, the website content is stored on CDN servers geographically closer to the users and reaches their computers much faster.

Content Distribution Network (CDN)





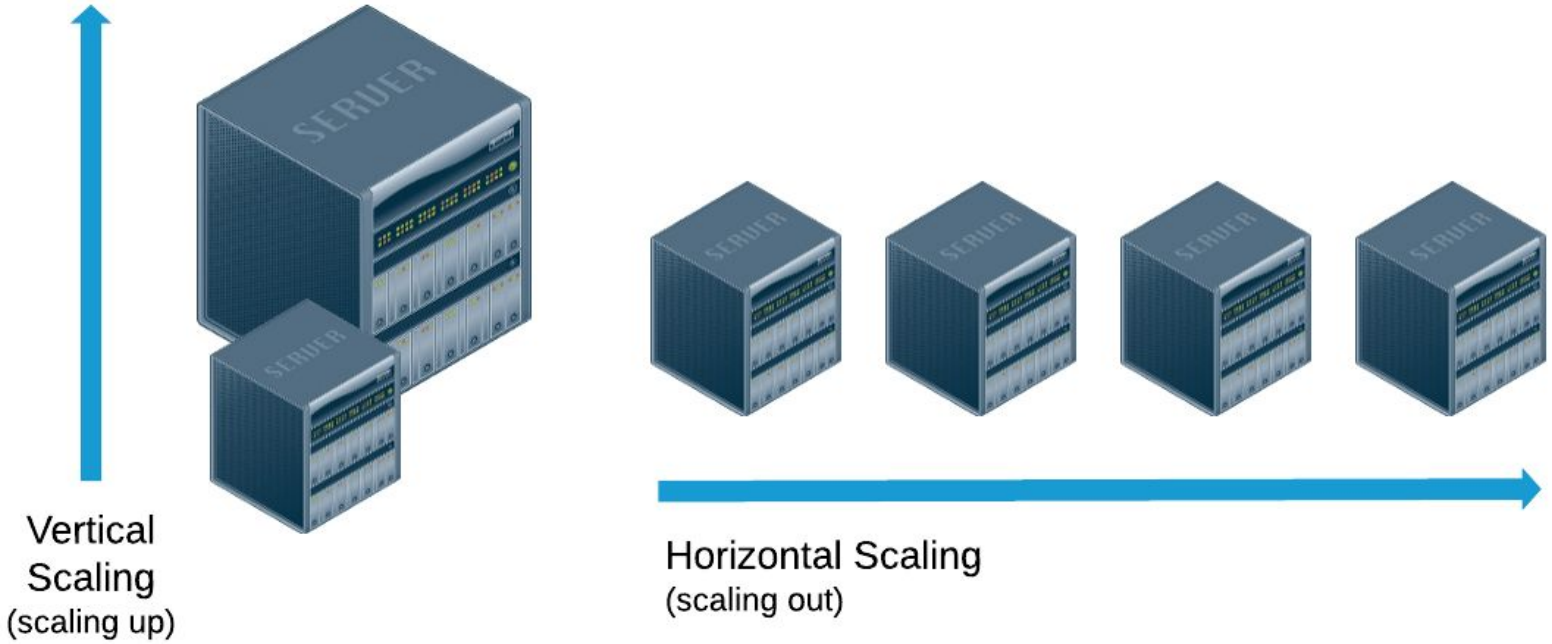
Application Latency

- Why is latency important?
- Latency plays a significant role in determining if an online business wins or loses a customer.
  - Globaldots: 2012 study showed that, for every 100 milliseconds of latency, they lost 1% in sales.
- Massive multiplayer online (MMO) games.
  - A slight lag in an in-game event ruins the whole experience.
  - A gamer with a high latency internet connection will have a slow response time despite having the best reaction time
- Algorithmic trading services need to process events within milliseconds.
  - Fintech companies have dedicated networks to run low latency trading. The regular network just won't cut it.
  - Huawei and Hibernia Atlantic, in 2011, lay a fiber-optic link cable across the Atlantic Ocean between London and New York. Estimated to cost approximately $300M just to save traders six milliseconds of latency.

Application Latency: Real World Example

- Google Speech Recognition
  - Bidirectional vs unidirectional models
- Latency needs affect the design of architecture!

Vertical vs Horizontal Scaling



Vertical Scaling (“Scaling Up”)

- Vertical scaling means adding more power to our server.
  - Let’s say our app is hosted by a server with 16 gigs of RAM.
  - To handle the increased load, we now augment the RAM to 32 gigs.
- Simplest way to scale
  - doesn’t require any code refactoring or complex configurations
- There is a limit to the compute power we can add to a single server.
- Downside of vertical scaling: availability risk.
  - The servers are powerful but few in number. There is always a risk of them going down and the entire website going offline.

Horizontal Scaling (“Scaling Out”)

- Horizontal Scaling adding more hardware to the existing hardware resource pool.
  - This increases the computational power of the system as a whole.
- Horizontal scaling limited only by $$.
  - We can keep adding servers after servers, setting up data centers after data centers.
- Horizontal scaling also allows us to scale dynamically in real-time as the traffic on a single service
  - Cloud computing configurations for this

Horizontal Scaling (“Scaling Out”)

- Often requires refactoring to allow multiple servers to handle requests!
- If you intend to run the code in a distributed environment, it needs to be **stateless** or have its important state managed outside the instance.
- Distributed memory like Redis, Memcache, etc., are used to maintain a consistent state application-wide.
- Eventual consistency, etc from your Databases classes!

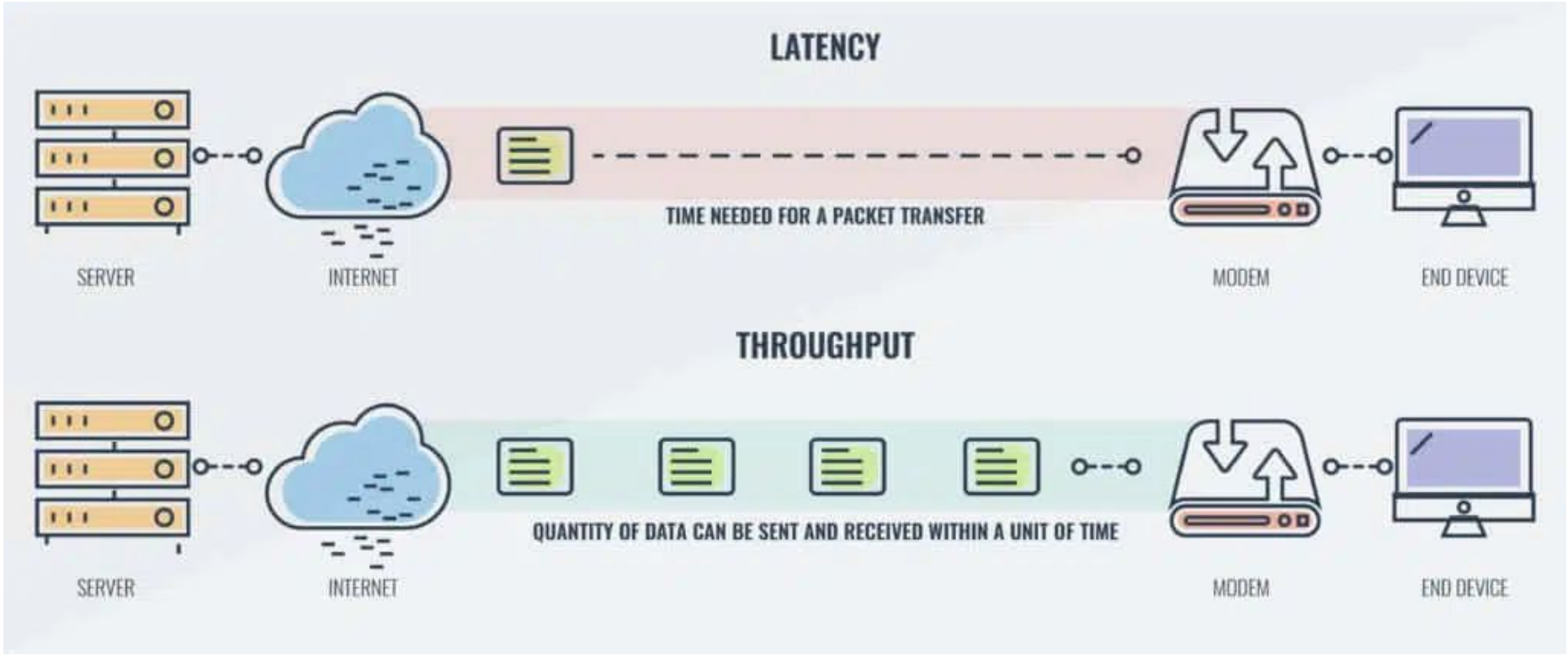
Latency Reduction

- How to reduce latency:
  - Profiling (dynamic analysis of code)
  - Caching
    - Hit the database only when it is really required.
    - Try to serve all the read requests from the cache.
  - CDN
    - Using a CDN further reduces the application's latency due to the proximity of the data from the requesting user.
  - Data compression
    - Since compressed data consumes less bandwidth, the data download on the client will be faster.
  - Avoid unnecessary request-response cycles
    - Simultaneous rather than chained

Throughput

Throughput is the number of data packets being successfully sent per second, and latency is the actual time those packets are taking to get there.

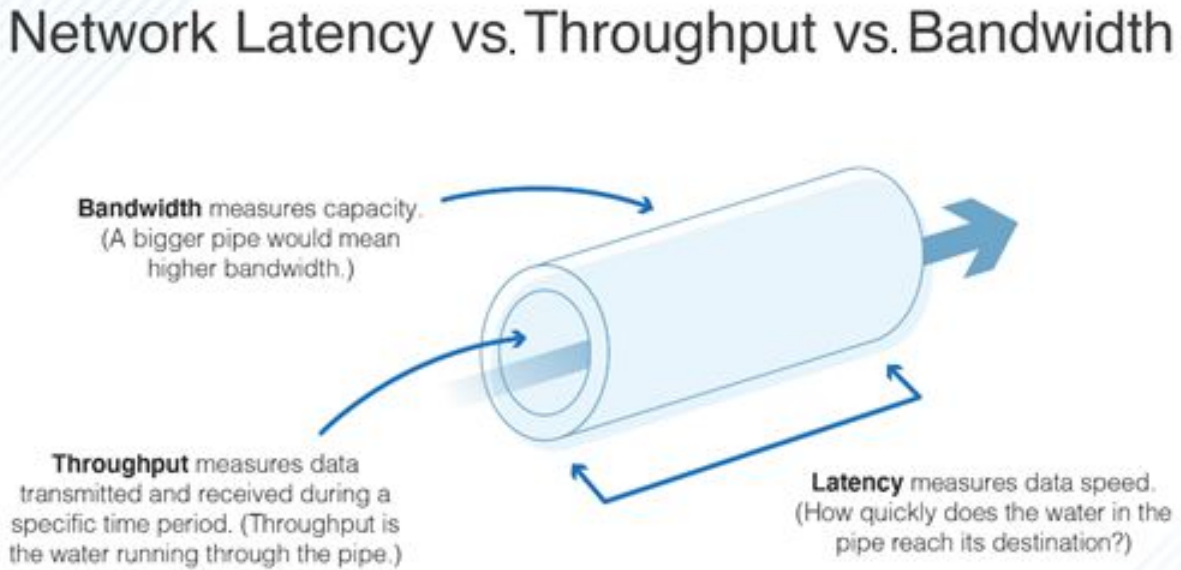
Throughput vs latency vs bandwidth



Throughput vs latency vs bandwidth

Bandwidth is used to refer to the maximum capacity of the network.

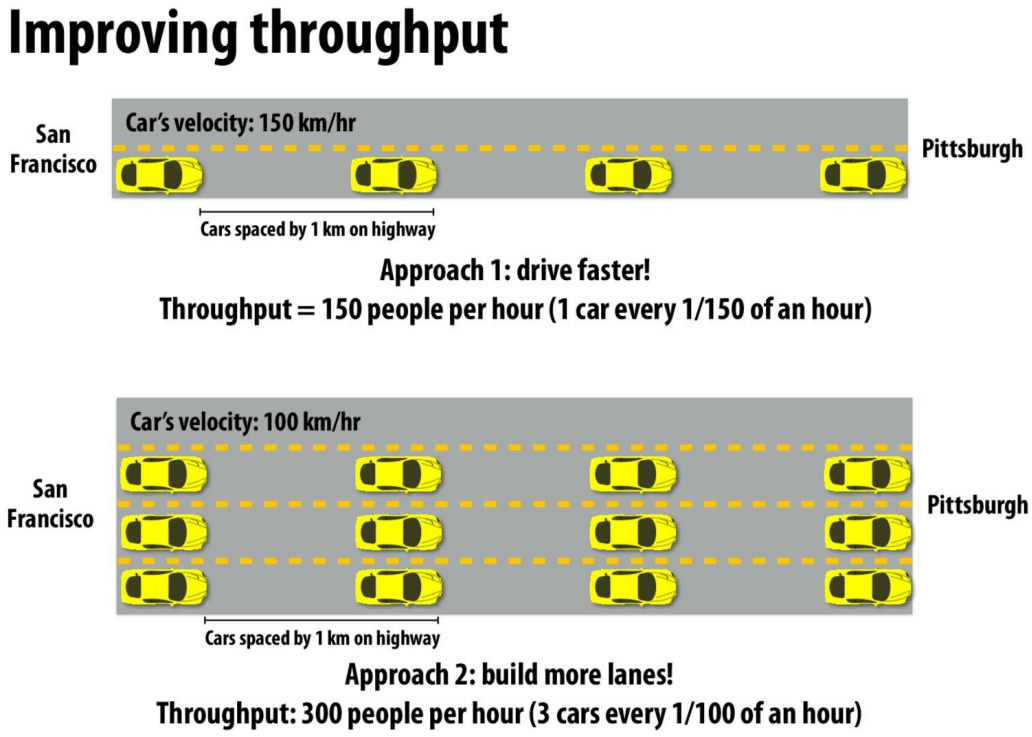
Throughput vs latency vs bandwidth



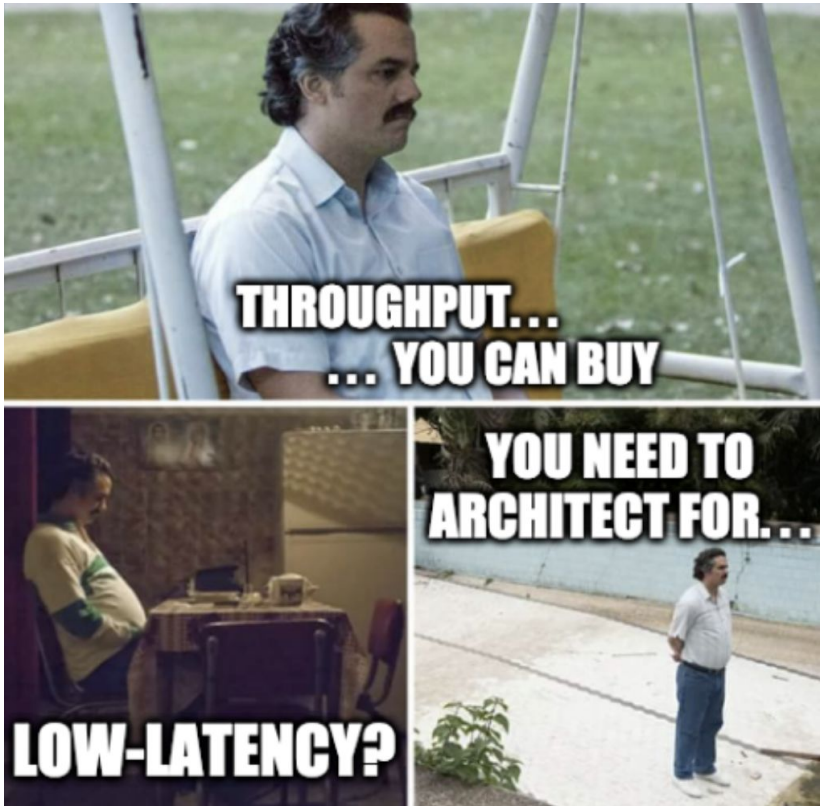
Bandwidth: Another analogy

- Latency is how long it takes you to drive from A to B
- Bandwidth is how wide the roads are
- Throughput is how many cars are on the road.

Bandwidth: Another analogy



Autoscaling



Autoscaling

- Autoscaling is a cloud computing feature that enables organizations to scale cloud services such as server capacities or virtual machines up or down automatically
- Based on defined situations, that you encode as a configuration
  - Min number of machines, max number of machines
  - Triggers such as # requests per minute

Reactive vs Scheduled Autoscaling

- By default, autoscaling is a reactive approach to decision making.
  - scales traffic as it responds in real-time to changes in traffic metrics.
  - However, in certain situations, especially when changes happen very quickly, it may be less effective to take a reactive approach.
- Scheduled autoscaling is a kind of hybrid approach to scaling policy that still functions in real-time, but also anticipates known changes in traffic loads and executes policy reactions to those changes at specific times.
  - Scheduled scaling works best in cases where there are known traffic decreases or increases at particular times of day, but the changes in question are typically very sudden.

Scalability Problems

You can't have scalability problems if nobody uses your app

2/12/17, 10:37 AM

Further Reading (for those interested)

https://engineering.fb.com/production-engineering/how-production-engineers-support-global-events-on-facebook/

https://cloud.google.com/architecture/scalable-and-resilient-apps

http://highscalability.com/blog/2016/9/28/how-uber-manages-a-million-writes-per-second-using-mesos-and.html