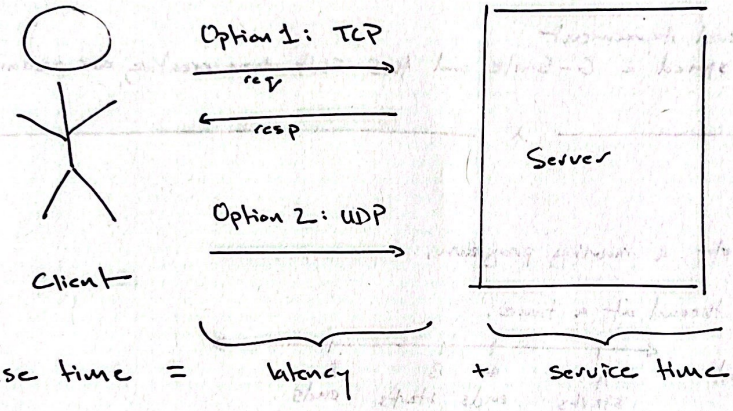
System Architecture

→ Definitions

- ^ System: A collection of components that interact w/ each other to provide some overall functionality at its interface.
  - Vertical Scaling or "Shared Memory Architecture"
  - Horizontal Scaling or "Shared Nothing Architecture"
    - → Sharding
    - → Distributed System

→ Client - Server Relationship

- Client: A user program that connects to a server to access a service.
- Server: Software / Hardware that serves a specific service to a client.



response time = latency + service time

- Transmission time (or the latency in 1 direction) = (size of the msg) / bandwidth

∴ If bandwidth = 10Mbps, what's the transmission time for 1 bit?

Transmission time = 1 bit / 10 Mbps = 1 bit / $1 \cdot 10^6$ bits per sec = $1 \cdot 10^{-6}$ sec

or 1 μsec

⇒ Performance Metrics

- SLI: Service Level Indicators:

1. correctness
   ⇒ Measured by the error rate as a function of all requests
   ⇒ Ex: "99% of requests are correct."

2. Availability
   ⇒ Measured in "9s"
   ⇒ Ex:
   - "2 9s" means service is available 99% of the time.
   - "3.5 9s" means the service is available 99.95% of the time.

3. Throughput
   ⇒ Ex: "Service can handle 3,000 requests per second (rps)"

4. Response Time:
   ⇒ Ex:
   - p50 < 200ms: 50% of response times are less than 200 ms
   - p99 < 1s: 99% of response times are less than 1 sec.

- SLO: Service Level Objectives

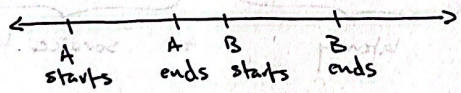
- SLA: Service Level Agreement
  ⇒ Ex: We signed a G-Suite and AWS SLA when creating our accounts.

⇒ Concurrency vs. Parallelism

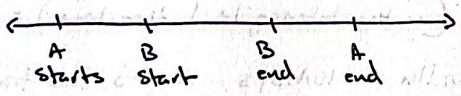
- Thread: A copy of a running program.

- Concurrency: 1 thread at a time.

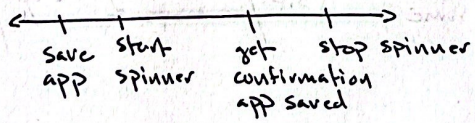
⇒ Sequential:



⇒ Interleved:



⇒ Ex:



- Critical Section: A section that can't be updated by multiple threads at the same time.

  ⇒ Ex: Databases!

  ⇒ Transactions on a Critical Section need to pass the ACID test:

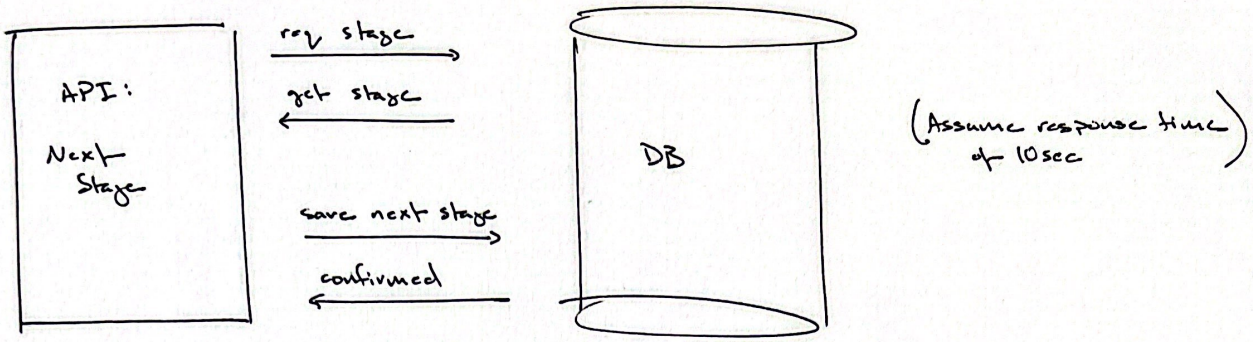
  A: Atomic. It's better to not save the data at all than only save part of the data (Ex: Pizza Order)

  C: Consistent

  I: Isolated

  D: Durable

- Critical Section Problem



- Critical Section Solutions (Synchronization Primitives)

  => Solutions to the critical section problem must satisfy 3 requirements:

  1. Mutual Exclusion: Only 1 process can be executing on the critical section at a time.
  2. Progress: Threads competing for access to a critical section shouldn't have any say on who gets to access a critical section.
  3. Bounded Weight: "Wait" times should be limited ("bounded")

     => This avoids a "dead lock" (when a thread is "starved")

  => Common Synchronization Primitives:

  1. Semaphore: A signal based approach (int)
  2. Mutex: A lock based approach

     => An object that has 2 states: taken (1) or free (0)
  3. Condition Variables
  4. Interlock operations