1. Abstraction

* Abstraction is the art of obfuscating details that we don’t need. It allows us to concentrate on the big picture.

1. Remote Procedure Call (RPC)

* Remote procedure calls (RPCs) provide an abstraction of a local procedure call to the developers by hiding the complexities of packing and sending function arguments to the remote server, receiving the return values, and managing any network retries.
* In the OSI model of network communication, RPC spans the transport and application layers.
* RPC mechanisms are employed when a computer program causes a procedure or subroutine to execute in a separate address space.
* The RPC method is like calling a local procedure, except that the called procedure is usually executed in a different process and on a different computer.

1. Consistency

* Consistency may mean many things:
  + One is that each replica node has the same view of data at a given point in time.
  + The other is that each read request gets the value of the recent write.
* There is a difference between consistency in ACID properties and consistency in the CAP theorem.
  + Database rules are at the heart of ACID consistency. If a foreign key indicates that deleting one row will also delete associated rows, a consistent system ensures that the state can’t contain related rows once the base row has been destroyed.
  + CAP consistency guarantees that, in a distributed system, every replica of the same logical value always has the same precise value.
* Four Main types of Consistencies in the order from weakest to strongest consistent model.

1. Eventual Consistency

* Ensures High Availability
* Example: Domain Name System
* Database: Cassandra

1. Casual Consistency

* Categorizes operation into dependent (causally related) and independent operations. Preserves the order of the causally related operations.
* Example: Commenting System

1. Sequential Consistency

* Preserves the ordering specified by each client’s program. However, sequential consistency doesn’t ensure that the writes are visible instantaneously or in the same order as they occurred according to some global clock.
* Example: Social Media Post

1. Strict Consistency aka linearizability

* A strict consistency or linearizability is the strongest consistency model. This model ensures that a read request from any replicas will get the latest write value. Once the client receives the acknowledgment that the write operation has been performed, other clients can read that value.
* Synchronous replication ensures linearizability, in which an acknowledgment is not sent to the client until the new value is written to all replicas.
* Linearizability affects the system’s availability, which is why it’s not always used. Applications with strong consistency requirements use techniques like quorum-based replication to increase the system’s availability.
* Example: Updating an account’s password
* Database: Amazon Aurora
* Application programmers must compromise performance and availability if they use services with strong consistency models.

1. Failure Models

* Five main types of failure models
* Fail-stop – Distributed System halts permanently but it is detectable
* Crash – Distributed System halts silently and is not detectable
* Omission failure - node fails to send or receive messages
* Temporal failure - node generates correct results, but is too late to be useful
* Byzantine failure - node exhibits random behavior like transmitting arbitrary messages at arbitrary times, producing wrong results, or stopping midway.

1. Non – Functional System Characteristics
2. Availability

* Availability is the percentage of time that some service or infrastructure is accessible to clients and is operated upon under normal conditions.
* A (in percent) = ((total time – amount of time service was down)/ total time) \* 100

1. Reliability

* Reliability, R, is the probability that the service will perform its functions for a specified time
* Reliability is the probability a system will fail in a given period. In simple terms, a distributed system is considered reliable if it keeps delivering its services even when one or several of its software or hardware components fail.
* A reliable distributed system achieves this through redundancy of both the software components and data (i.e. by eliminating every single point of failure).
* The measurement of availability is driven by time loss, whereas the frequency and impact of failures drive the measure of reliability.
* Mean Time Between Failures (MTBF) = (Total Elapsed Time – Sum of Downtime)/ Total Number of failures
* Mean time to repair (MTTR) = Total Maintenance Time/ Total Number of repairs

1. Scalability

* Scalability is the ability of a system to handle an increasing amount of workload without compromising performance.
* Workloads can be of different types – Request WL, Data/ Storage WL
* Approaches of Scalability – Vertical, Horizontal

1. Maintainability

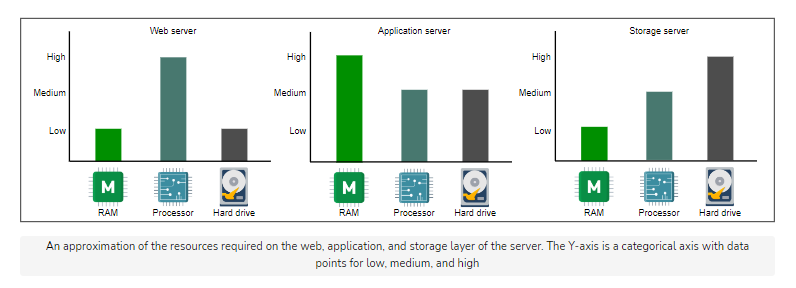
* Maintainability, M, is the probability that the service will restore its functions within a specified time of fault occurrence. M measures how conveniently and swiftly the service regains its normal operating conditions.
* Example, suppose a component has a defined maintainability value of 95% for half an hour. In that case, the probability of restoring the component to its fully active form in half an hour is 0.95.
* We use (mean time to repair) MTTR as the metric to measure M.
* Maintainability refers to time-to-repair, whereas reliability refers to both time-to-repair and the time-to-failure.

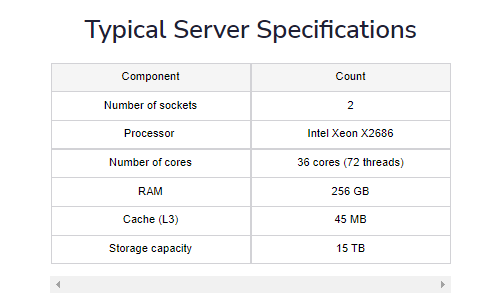
1. Fault tolerant

* Fault tolerance refers to a system’s ability to execute persistently even if one or more of its components fail. Here, components can be software or hardware.
* Techniques to make system fault tolerant

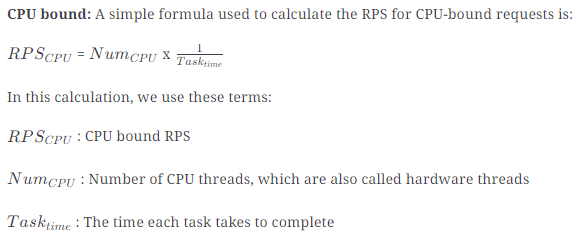
1. Replication
2. Checkpointing
3. Types of Data center servers
4. Web Servers
   * Web servers are the first point of contact after load balancers. Main responsibility is to handle API calls from the clients.
5. Application Servers
   * Application servers run the core application software and business logic.
   * Application servers primarily provide dynamic content, whereas web servers mostly serve static content to the client, which is mostly a web browser.
6. Storage Servers
   * Various types of data are now being stored in different storage units. For instance, YouTube uses the following datastores:
     1. Blob Storage for Encoded Videos
     2. Temporary Processing Queue Storage for few hundred hours of video content uploaded daily to YouTube for processing.
     3. Bigtable for storing a large number of thumbnails of videos.
     4. Relational database management system (RDBMS) for users and videos metadata (comments, likes, user channels, and so on.
     5. Hadoop’s HDFS for analytics.

* The servers described above are not the only types of servers in a data center. Organizations also require servers for services like configuration, monitoring, load balancing, analytics, accounting, caching, and so on.





1. Request Estimation
   1. CPU Bound Requests



* 1. Memory-bound Requests

