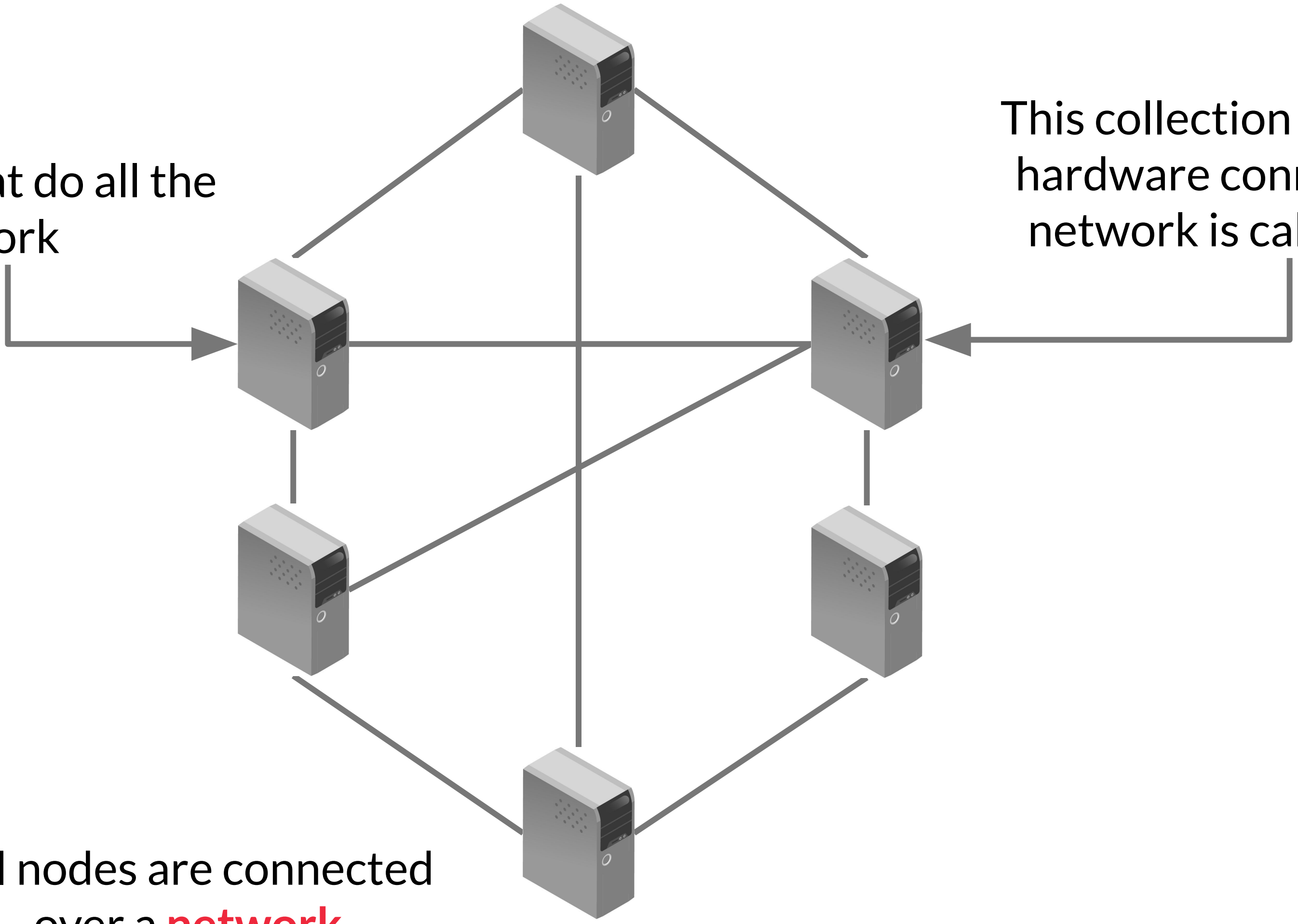
Implementation of Distributed Systems

Session: Implementation of Distributed Systems

Instructor: Vishwa Mohan

DISTRIBUTED SYSTEMS COMPONENTS



Nodes that do all the work

This collection of commodity hardware connected over a network is called a **cluster**

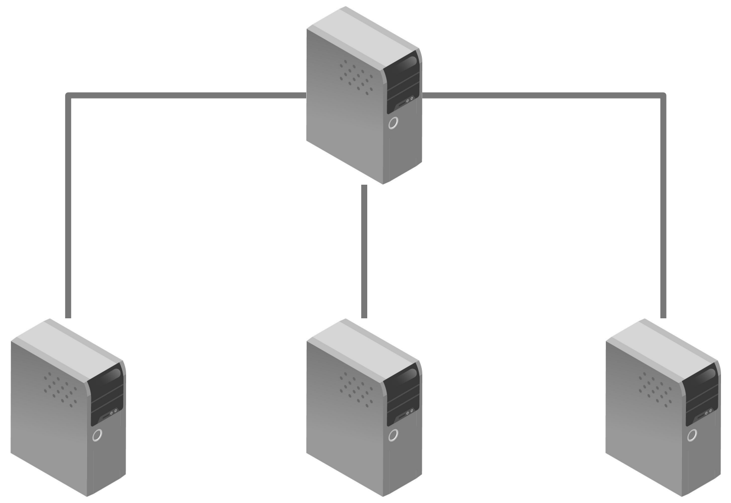
All nodes are connected over a **network.**

DISTRIBUTED SYSTEMS ARCHITECTURE

- A **cluster** is defined as a group of nodes working together and configured in such a way that they appear as a single system.
- Since these machines share the workload, to ensure proper **resource utilisation and bookkeeping,** there needs to be some structure in the arrangement of these clusters.
- The are mainly two ways a cluster of machines are arranged in a distributed system:
  - Master-Slave architecture
  - Master-Master architecture

MASTER-SLAVE ARCHITECTURE

- In this architecture a Master node controls and coordinates all the other nodes(Slaves) in the network.
- Typically the master is used for WRITE operations while the slaves are used for READ operations.
- The master ensures consistency among slaves.
- Typically, there is only one master making it a single point of failure.

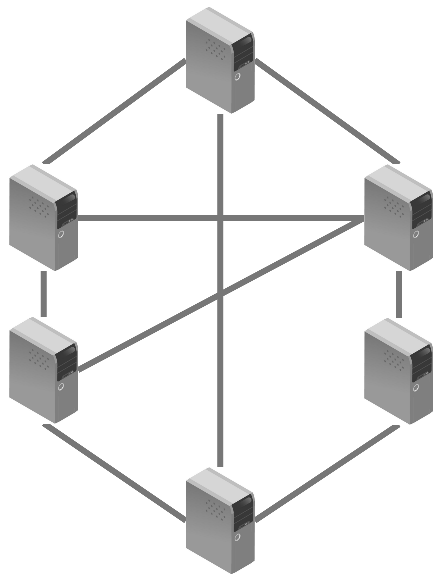


MASTER-SLAVE EXAMPLES

- When MySQL-Postgres is scaled/shared it behaves in the master-slave order.
- Similarly MongoDB follows the master-slave model in the NoSQL world.
- Hadoop makes use of the master-slave model for storage(HDFS) as well as data processing(MapReduce).
- Apache Airflow(A pipeline automation tool) makes use of a distributed Master-Slave architecture using a scheduler as a master node and Workers are slave nodes.

MASTER-MASTER ARCHITECTURE

- In this architecture all the nodes are same. They keep the information about others by different communication protocols.
- Data is equally shared by all nodes. Typically data is distributed by the means of consistent hashing.
- The client can connect to any of the nodes for READ/WRITE irrespective of weather data is present in that node or not.



MASTER-MASTER ARCHITECTURE

- Since all the nodes are equal, they can be easily replaced in case of failure. This makes the model easily fault tolerant.
- On the other hand there is no central coordinator in the master-master model which makes it less consistent.
- **Cassandra** is a NoSQL DBMS that makes use of the Master-Master model.
- Some other examples include:
  - **DynamoDB** – Amazon’s NoSQL database system.
  - **Elasticsearch** – An open-sourced distributed search engine.

CHALLENGES IN IMPLEMENTATION

01 Performance

02 Scalability

03 Fault Tolerance and Reliability

04 Security

PERFORMANCE

- Since distributed system, involve multiple machines connected over a network. The added network hop results in added latency.
- Total Time = Computation time + Network delay
- Bad design/planning or inefficient resource utilisation in distributed systems can result in extremely poor performance or high response time.

WACOM

HOW TO HANDLE THIS?

- Reduce computation time by using concurrent/parallel computation.
- Reduce network delay by increasing bandwidth and redundancy in network.
- Implement proper load balancing and partitioning schemes.
  - **Load balancing** – Distribute the tasks to available resources in an efficient manner.
  - **Partitioning schemes** – Divide data into logical partitions for easier access.
- **Optimal redundancy** - Too much redundancy can increase write time by a lot, thus, only an optimal number of replications must be made.

WACOM

EXAMPLE

- When **starGard** initially moved to a distributed model, the **response time** for individual systems increased.
- To counter this:
  - You leverage the potential of distributed computing to solve parallel problems using multiple machines.
  - You increased the network bandwidth to deal with the network delays.
- This solved the problem more efficiently and managed to bring down the response time to acceptable levels.

SCALABILITY

- A good scalable system should be capable of smoothly increasing the capacity with increasing load.
- Adding a new node to the cluster requires rearrangement/rebalancing of the cluster.
- For example, if the data is distributed among the node by simple hashing, adding a new node to this cluster would require rebalancing.

WACOM

HOW TO HANDLE THIS?

- Proper partitioning by the use of **consistent hashing**.
  - **Simple hashing** is technique used to map key-value pairs when the the size of your data is constant. Any change in size results in all the keys being remapped.
  - In scalable distributed systems though, the size is constantly increasing which makes simple hashing a poor choice.
  - **Consistent hashing** is a special type of hashing in which only a fraction of the keys need to be remapped on average, every time the size changes.
- **Sharding** - Also known as horizontal partitioning, refers to the breaking of a huge dataset into small partitions called shards. Each such shard is stored in a different machine which can easily be scaled with increased demand.
- Choosing Vertical and Horizontal scaling when needed.

EXAMPLE

- As discussed **starGrad** has scaled using horizontal scaling by adding more machines into your cluster.
- This takes care of the computational requirements but size of the datasets your company is working on has also grown too large for any single computer in your network. In this case we can make use of **sharding** to distribute large datasets among different nodes.
- NoSQL database programs like **Cassandra** make use of **consistent hashing** to make their distributed systems elastically scalable.

FAULT TOLERANCE

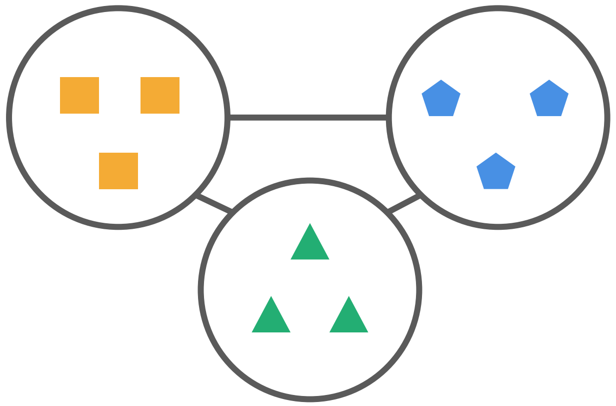
- Distributed systems are based on :
  - Nodes/Commodity hardware
  - Network
- Both of these things make the distributed system highly susceptible to failure.
- Since storage and computation responsibilities is shared by all the nodes in the distributed system. Making the model failure tolerant becomes a challenge.
- Network failures might result in partitions in the network.

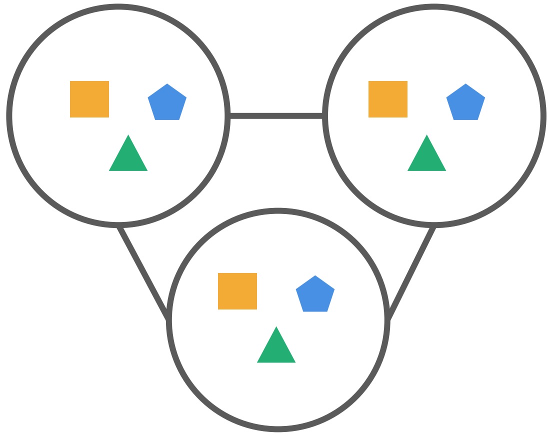
RELIABILITY

- Since distributed systems are susceptible to fault, the reliability of the system is also at stake.
- Reliability = 1 - probability of failure
- Reliability can be ensured by:
  - High fault tolerance
  - Ease of use and maintenance
- Higher reliability results in higher availability but not vice versa.

HOW TO HANDLE THIS?

- **Fault prediction** - Based on the use case some failures can be predicted and prepared for.
- **Proper maintenance** - Since we are using commodity hardware, maintenance is a must.
- **Auto recovery** – Planned/Automated recovery procedures are used to get failed nodes back up or replace the node if recovery isn't possible.
- **Smart Redundancy** – This refers to making multiple copies of the data and storing it in different machines belonging to separate partitions.





EXAMPLE

- Exam season in **starGrad** are particularly traffic heavy. Every learner is doing some last minute preparation. In such a scenario you have to make sure your systems are reliable and fault tolerant so that the learners have access to all the resources they need.
- This can be done by
  - Having backup copies of all the content that are distributed among the nodes in a partition-tolerant manner.
  - Doing maintenance check before exam season when traffic is expected to be heavy.

SECURITY

- Since the components are distributed, managing and maintaining security in a distributed manner is a challenge.
- In centralised systems everything is packed in a single place hence everything can be monitored easily.
- In distributed systems a lot of different systems with different security standards are in a network hence making it vulnerable to attacks in:
  - The storage layer
  - The transfer layer

WACOM

HOW TO HANDLE THIS?

- Proper authentication(Transient token based)should be done between systems during communication.
- Stateless systems - The nodes should not store any unnecessary data related to past communications/computations.
- Proper vigilance/monitoring - All the nodes must be continuously monitored, and any signs of vulnerability/failure should be fixed right away before it gets exploited.

WACOM

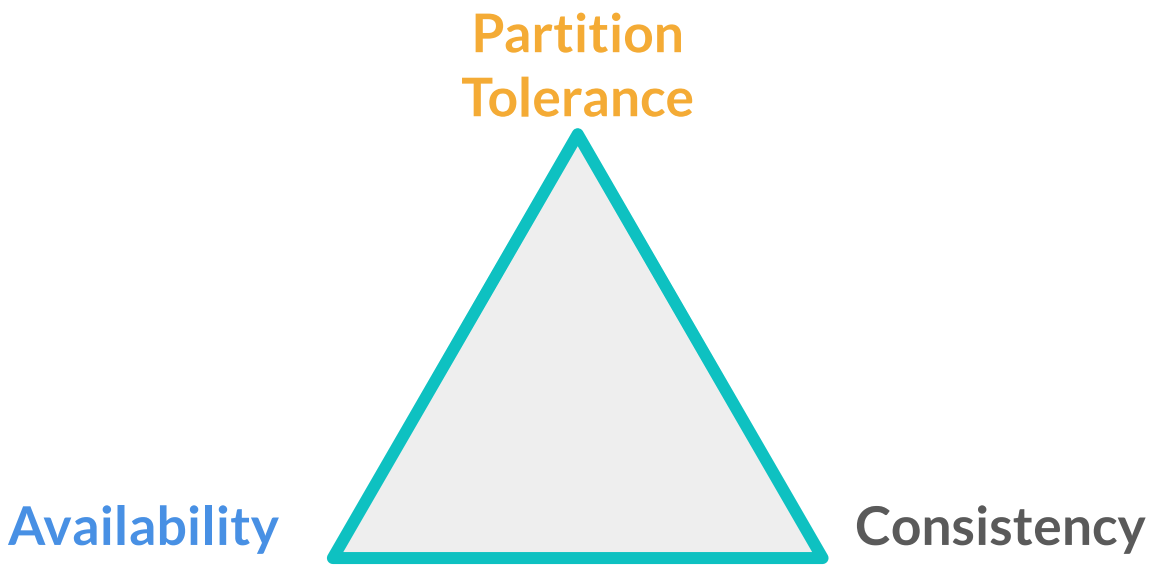
EXAMPLE

- As the starGrad application grows in scale and popularity, it will be more susceptible to attacks trying to steal your resources or malicus learners trying to access classified content like solution documents or results.
- To ensure its continued success you must make sure the platform has consistent security standards and each node makes proper authentication before relaying any data. For example, only authorised IP addresses can read your data.
- To understand the concept of stateless systems, you can take the example of a simple telephone network.

WACOM

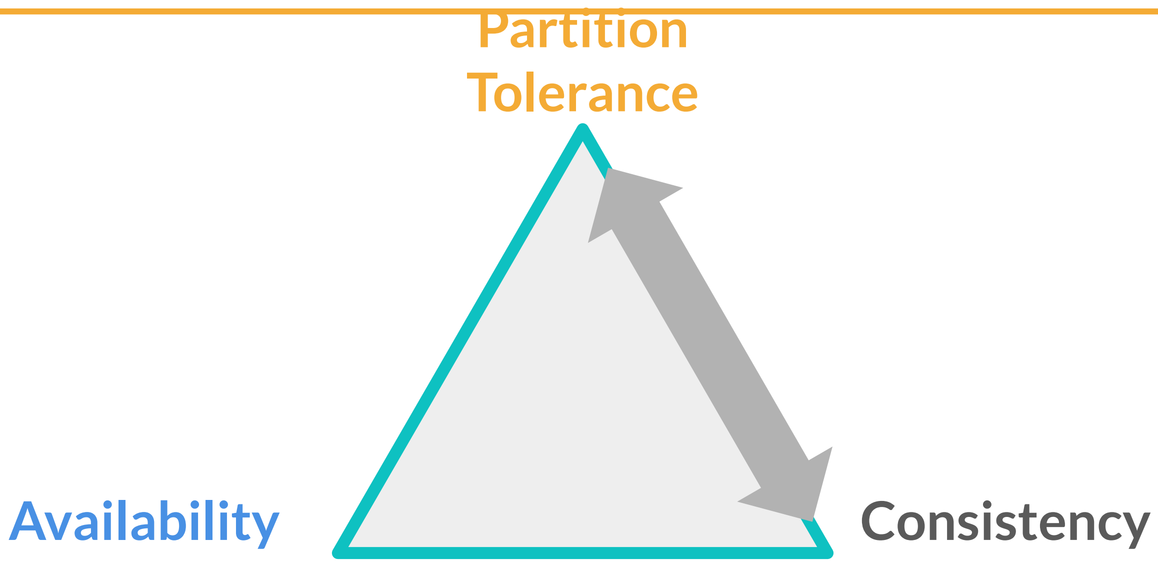
CAP THEOREM

- CAP Theorem states that at any time in a system. We can only support two of these three entities - **Consistency, Availability** and **Partition Tolerance.**
- However in distributed systems, **partition tolerance is a must**. Without partition tolerance neither consistency nor availability can be guaranteed.
- Hence the trade-off is between consistency and availability.



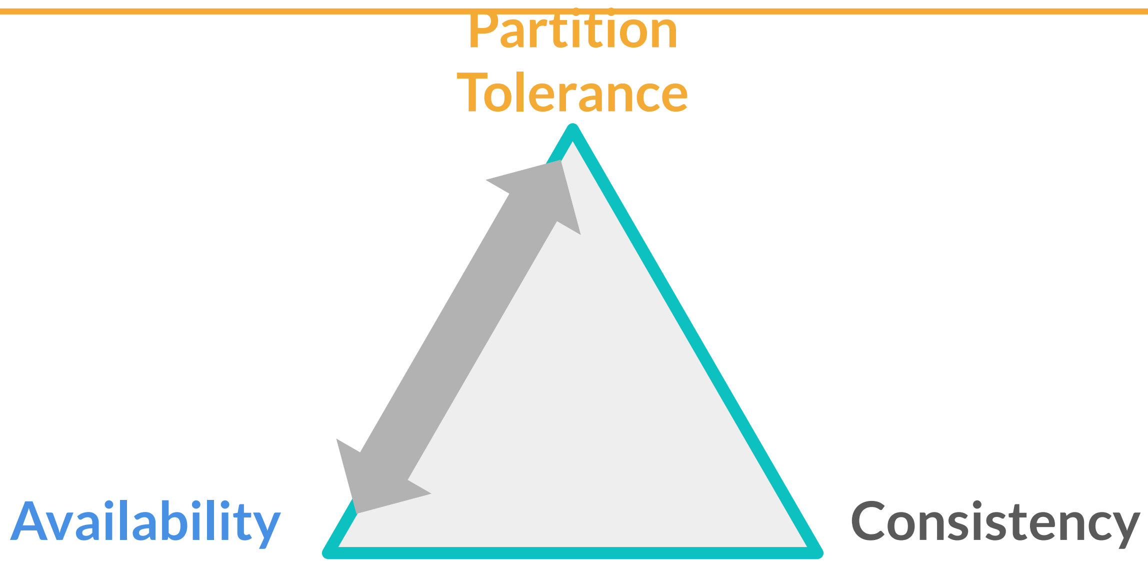
CP SYSTEMS

- CP systems have high **Consistency** and have **Partition tolerance**.
- For example, in a multiplayer game, the position of all the other players needs to be consistent for every player even at the cost of the game not being available to new players during high traffic time. Thus, a CP-based model is more suitable.
- IRCTC ticket booking needs to be highly consistent to avoid multiple bookings. Availability, as we know is compromised frequently.



AP SYSTEMS

- AP systems have high **Availability** and have **Partition tolerance**.
- A social media app like Twitter or Facebook doesn't need the number of likes to be strictly consistent to all the users. On the other hand availability is important in such platforms hence an AP model is more suitable.
- Search engines like Bing and Google are also AP systems as availability is a must, whereas consistency can be compromised.



SUMMARY

01 What are Clusters?

02 Distributed Systems Architecture

03 Challenges in implementation

04 CAP Theorem

Thank You