Introduction to Distributed Systems

M T W T F S S
Page No.:
Date:
YOUVA

A computer is a system.

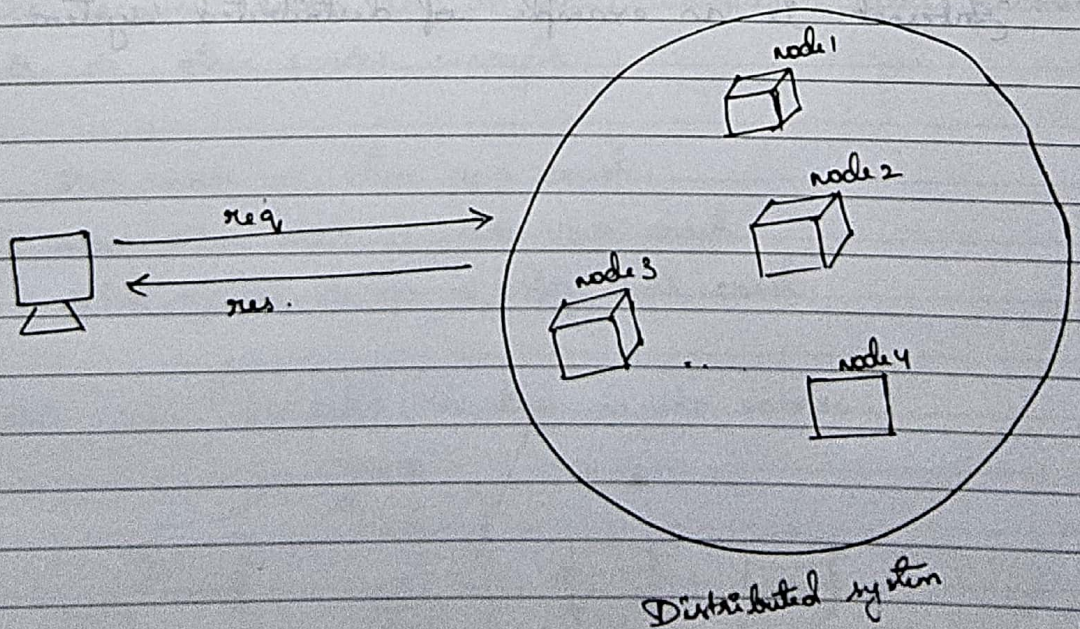
A distributed system means multiple machines that are interconnected.

We are no longer using a single server for processing client requests. As the number of clients grow, the need for adding multiple servers increases. As a result, distributed system which consists of multiple machines (in this case, servers) is getting popular.

A computer comprises of multiple cores which makes it a distributed system as well, however, this is not valid in the context of today's reality.

Definition: A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system.

Usually, computing element is a physical machine, we call it node.



Distributed system

Example Instagram: scrolling the feed, visit profile, checking DMs, uploading photos etc. navigations feel smooth as there is a distributed system beneath which ensures it.

You may notice that part of the system fails - like feed doesn't load but upload photo works. This shows that part of the Instagram system is running on one set of nodes and another part is probably running independently on a different set of nodes.

This is called a "State of incoherence"

Summary:-

A distributed system is a system in which:

* A bunch of nodes run programmatically.
* The nodes have a common goal to achieve, like serving users with some data
* The nodes behave coherently, meaning that users should see what they expect.

"Internet" is an example of distributed system.

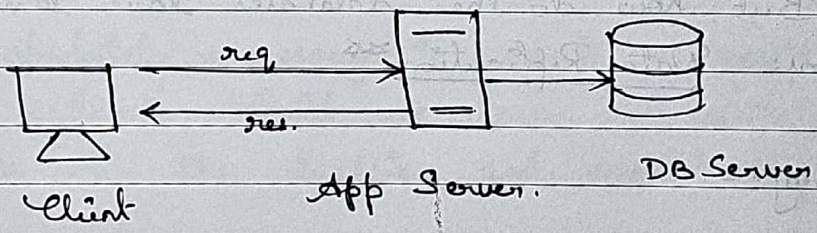
How the single node grows to multiple and why?

M T W T F S S
Page No.:
Date:
YOUVA

Let's say there is a new application you created which returns response when client requests.

Initially, you need a system that serve data to the client e.g; a processor which computes data and returns (typically, processor in computer).
This is called server. Your computer is a node.

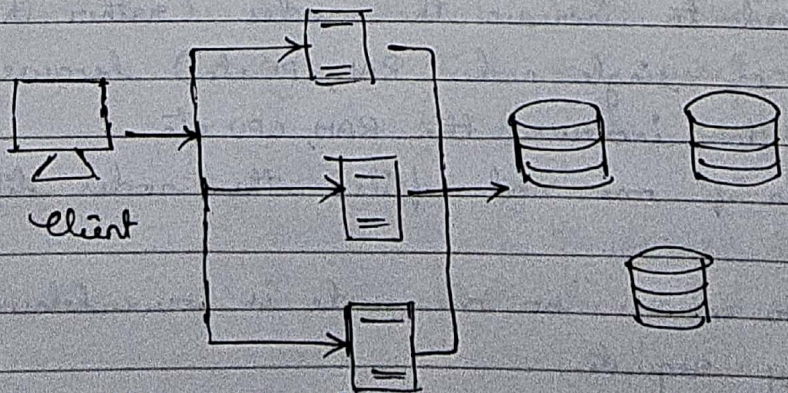
What if the node crashes due to power failure or other reasons? The server is also using data from database.



If the server (either app or DB) crashes, users encounter 503 error or other errors maybe.

503 error → for app crash.
some other error → for DB crash.
entire system down → for both crash.

In this case, we add backup server nodes.



This is also helpful to handle increased load.

Key points :-

* The user request is processed by one of the three app servers and returned response to the user.
When there are multiple user requests, all these requests are balanced among these nodes and the responses are returned to corresponding users
This is called Load Balancing. ***

* The application servers create a query and send it to one of the three database nodes.
All the three of the database nodes have the same copy of the data and each of them is capable of serving the request.
But how do the databases have same data copy?
Ans: Data Replication. ***

Summary :-

To implement the application, handle increased load, (Scalability) handle faults (Fault Tolerance), we implemented multiple nodes. Hence the single server application has become Distributed System.

Scalability and Fault Tolerance are characteristics of Distributed System.

Note :-

We need to increase the nodes (rather than increasing resources of single node RAM, CPU etc.) because there is a limit to increase the RAM, CPU etc.
Also even if one node fails, other nodes still are up & running.
It also ensures no one node is overwhelmed with too many requests.

Evaluate the need for distributed s/m

M T W T F S S
Page No.:
Date:
YOUVA

Following questions to be answered to evaluate need for D.S.

* Did we need to add more nodes or a single reasonably larger node should be enough? What does the traffic look like? For example, we have single node of 20GB RAM, maybe increasing it to 32 GB RAM should be enough? Do we need another node?

* For failed node scenarios, nodes on "standby" can take over? Yes, but that means there will be a downtime in our system. But what if a few minutes of downtime does not make any difference in our system?

* Was it absolutely necessary to create a multi node database system? Could we have one database node to handle all the traffic, and would that be enough?

* When there is a crash in DB node, would it be possible to spin up a new DB node and copy all the data? Maybe from a backup? Would the downtime affect our business?

* Is the cost of building and running a distributed system worth it, given our business requirements and growth?

Bottomline is - be cautious before making a distributed system. Find out whether it is worth it to go for the complexity of a distributed system.
Go for it if the answer to above questions is still that distributed system would be best!

Scalability ⇔ "Growth capacity"

Reliability ⇔ "Trustworthiness"

Availability ⇔ "Uptime"

Consistency ⇔ "Sameness"

Efficiency ⇔ "Resourcefulness" (functions in optimal way with minimum resources)

Robustness ⇔ "Resilience"

Security ⇔ "Safety"

Maintainability ⇔ "Easy care" (Easy to update and upgrade)

Modularity ⇔ "Compartmentalization"

Fault tolerance ⇔ "Forgiveness" (continues to operate even if part of system fails).