Key Characteristics of Distributed Systems

Key characteristics of a distributed system include:

- Scalability
- Reliability
- Availability
- Efficiency
- Manageability

Let’s briefly review them.

Scalability

> ❗ **Scalability** is the capability of a system, process, or a network to grow and mange increased demand.
>
> Any distributed system that can be continuously evolve in order to support the growing amount of work is considered to be scalable.

A system may have to scale because of many reasons like increased data volume or increased amount of work, e.g. number of transactions.

A scalable system would like to achieve this scaling without performance loss.

Generally, the performance of a system, although designed (or claimed) to be scalable, declines with the system size due to the management or environment cost.

For instance, network speed may become slower because machines tend to be far apart from one another.

More generally, some tasks may not be distributed, either because of their inherent atomic nature or because of some flaw in the system design.

At some point, such tasks would limit the speed-up obtained by distribution

A scalable architecture avoids this situation and attempts to balance the load of all the participating nodes evenly.

Horizontal vs. Vertical Scaling

Horizontal Scaling means that we scale by adding more servers into our pool of resources.

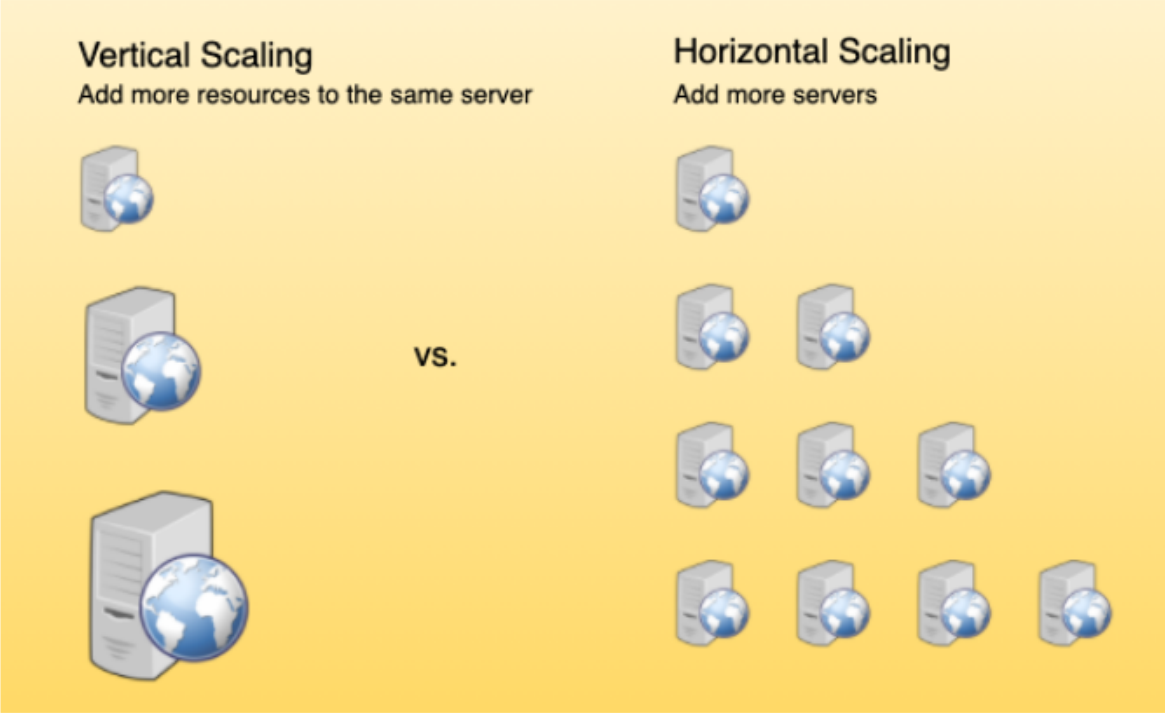
Vertical Scaling means that we scale by adding more power (CPU, RAM, storage etc.) to an existing server.

With horizontal scaling, it is often easier to scale dynamically by adding more machines into the existing pool.

Vertical scaling is usually limited to the capacity of a single server, and scaling beyond that capacity often involves downtime and comes with an upper limit.

Good examples of horizontal scaling are Cassandra and MongoDB as they both provide an easy way to scale horizontally by adding more machines to meet growing needs.

Similarly, a good example of vertical scaling is MySQL as it allows for an easy way to scale vertically by switching from smaller to bigger machines. However, this process often involves downtime.



Vertical scaling vs. Horizontal scaling

Reliability

> ❗ By definition, **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 on or several of its software or hardware components fail.

Reliability represents one of the main characteristics of any distributed system, since in such systems any failing machine can always be replace by another healthy one, ensuring the completion of the requested task.

Take the example of a large e-commerce store (like Amazon), where one of the primary requirements is that any user transaction should never be cancelled due to a failure of the machine that is running the transaction.

For instance, if a user has added an item to their shopping cart, the system is expected not to lose it.

A reliable distributed system achieves this through redundancy of both the software components and data.

If the server carrying the user's shopping cart fails, another server that has the exact replica of the shopping cart should replace it.

Obviously, redundancy has a cost and a reliable system has to pay that to achieve such resilience for services by eliminating every single point of failure.

Availability

> ❗ By definition, **availability** is the time a system remains operational to perform its required function in a specific period.
>
> It is a simple measure of the percentage of time that a system, service, or a machien remains operational under normal conditions.

As an example, an aircraft that can be flown for many hours a month without much downtime can be said to have a high availability.

Availability takes into account maintainability, repair time, spares availability, and other logistics considerations.

If an aircraft is down for maintenance, it is considered not available during that time.

Reliability is basically availability over time considering the full range of possible real-world conditions that can occur.

An aircraft that can make it through any possible weather safely is more reliable that one that has vulnerabilities to possible conditions.

Reliability vs. Availability

If a system is reliable, it is available.

However, if it is available, it is not necessarily reliable.

In other words, high reliability contributes to high availability, but it is possible to achieve a high availability even with an unreliable product by minimising repair time

and ensuring that spares are always available when they are needed.

Let’s take the example of an online retail store that has 99.99% availability for the first two years after its launch.

However, the system was launched without any information security testing. The customers are happy with the system, but they don’t realise that it isn’t very reliable as it is vulnerable to likely risks.

In the third year, the system experiences a series of information security incidents that suddenly result in extremely low availability for extended periods of time.

This results in reputational and financial damage to the customers.

Efficiency

To understand how to measure the efficiency of a distributed system, let’s assume that we have an operation that runs in a distributed manner and delivers a set of items as a result.

Two standard measures of its efficiency are the response time (or latency) that denotes the delay to obtain the first item and the throughput (or bandwidth) which denotes the number of items delivered in a given unit of time.

The two measures correspond to the following unit costs:

- Number of messages globally sent by the nodes of the system regardless of the message size
- Size of messages representing the volume of data exchanges

The complexity of operations supported by distributed data structure (e.g. searching for a specific key in a distributed index) can be characterised as a function of one of these cost units.

Generally speaking, the analysis of a distributed structure in terms of ‘number of messages’ is over-simplistic.

It ignores many aspects, including the network topology, the network load, and its variation, the possible heterogeneity of the software and hardware components involve in data processing and routing etc.

However, it is quite difficult to develop a precise cost model that would accurately take into account all these performance factors; therefore, we have to live with rough

but robust estimates of the system behaviour.

Manageability (or Serviceability)

Another important consideration while designing a distributed system is how easy it is to operate and maintain.

> ❗ **Manageability** (or **serviceability**) is the simplicity and speed with which a system can be repaired or maintained.
>
> If the time to fix a failed system increases, then availability will decrease.

Things to consider for manageability are the ease of diagnosing and understanding problems when they occur, ease of making updates or modifications, and how simple the system is to operate (i.e. does it routinely operate without failure or exceptions?).

Early detection of faults can decrease or avoid system downtime.

For example, some enterprise systems can automatically call a service center (without human intervention) when the system experiences a system fault.