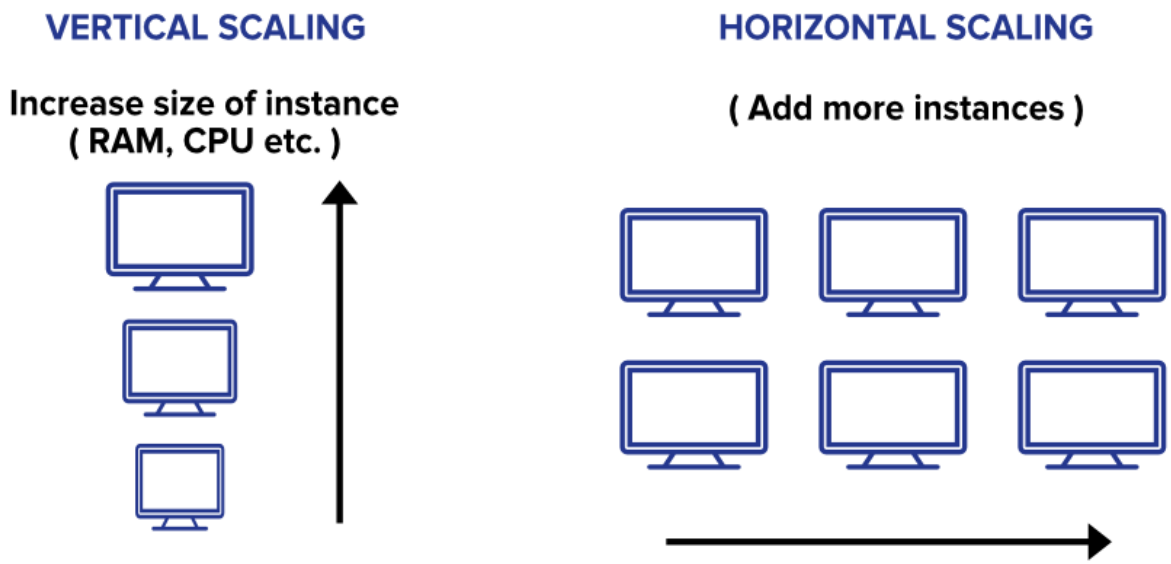
Scaling

- It can be defined as a process to expand the existing configuration (servers/computers) to handle a large number of user requests or to manage the amount of load on the server.
- This process is called scalability. This can be done either by increasing the current system configuration (increasing RAM, number of servers) or adding more power to the configuration. Scalability plays a vital role in the designing of a system as it helps in responding to a large number of user requests more effectively and quickly.

There are two ways to do this :

1. **Vertical Scaling**
2. **Horizontal Scaling**



---

1. Vertical Scaling

In simple terms upgrading the capacity of a single machine or moving to a new machine with more power is called vertical scaling. You can add more powers to your machine by adding better processors, increasing RAM, or other power increasing adjustments. Vertical scaling can be easily achieved by switching from small to bigger machines but remember that this involves downtime. You can enhance the capability of your server without manipulating your code.

Advantage

- This approach is also referred to as the '**scale-up**' approach.
- It doesn't require any partitioning of data and all the traffic resides on a **single node with more capacity**.

1. It is easy to implement

2. Reduced software costs as no new resources are added
3. Fewer efforts required to maintain this single system

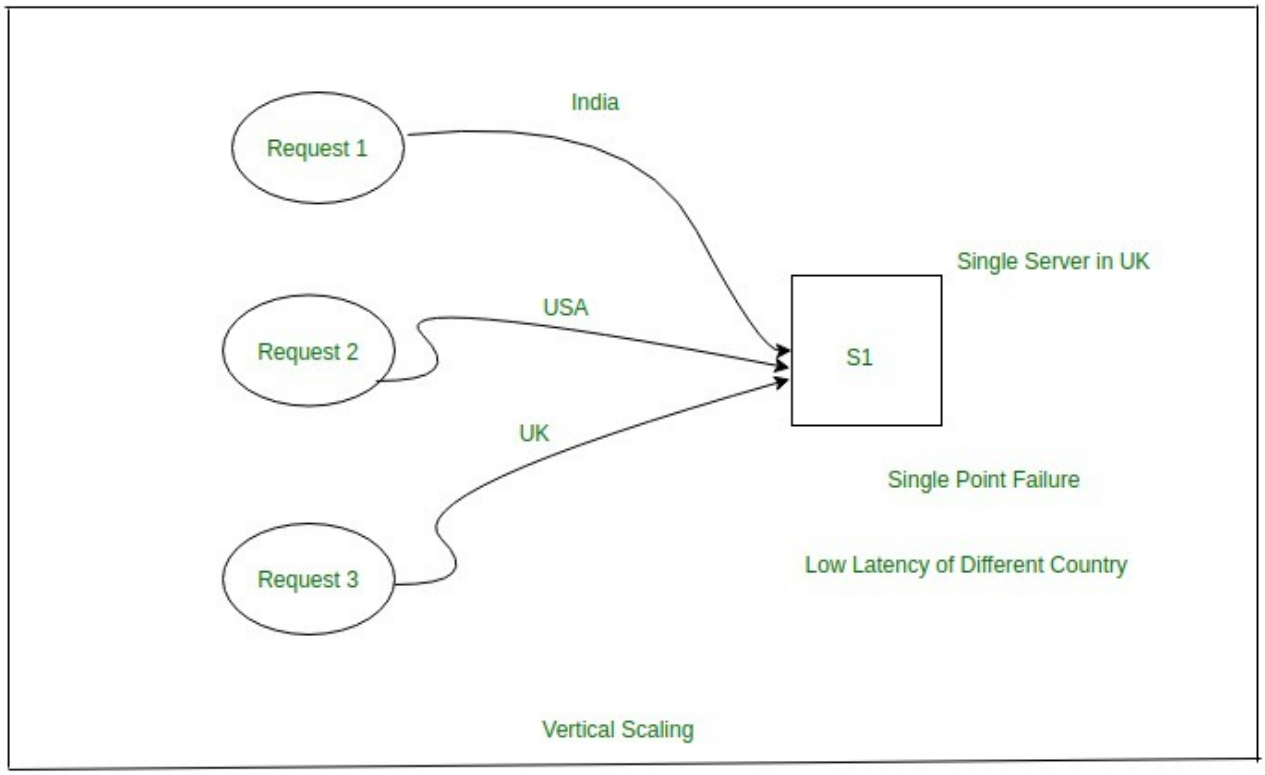
Drawbacks

- Limited Scaling.
- Limited potential for improving network I/O or disk I/O.
- Replacing the server will require downtime in this approach.

1. Single-point failure
2. Since when the system (server) fails, the downtime is high because we only have a single server
3. High risk of hardware failures
4.

A Real-time Example of Vertical Scaling

When traffic increases, the server degrades in performance. The first possible solution that everyone has is to increase the power of their system. For instance, if earlier they used 8 GB RAM and 128 GB hard drive now with increasing traffic, the power of the system is affected. So a possible solution is to increase the existing RAM or hard drive storage, for e.g. the resources could be increased to 16 GB of RAM and 500 GB of a hard drive but this is not an ultimate solution as after a point of time, these capacities will reach a saturation point.



---

2. Horizontal Scaling

- This approach is the best solution for projects which have requirements for high availability or failover. In horizontal scaling, we enhance the performance of the server by adding more machines to the network, sharing the processing and memory workload across multiple devices.

- We simply add more instances of the server to the existing pool of servers and distribute the load among these servers.

- In this approach, there is no need to change the capacity of the server or replace the server. Also, like vertical scaling, there is no downtime while adding more servers to the network.

- Most organizations choose this approach because it includes increasing I/O concurrency, reducing the load on existing nodes, and increasing disk capacity.

- This approach is also referred to as the '**scale-out**' approach.

- Horizontal scalability can be achieved with the help of a distributed file system, clustering, and load–balancing.

- Traffic can be managed effectively.

- Easier to run fault-tolerance.

Advantages of Horizontal Scaling

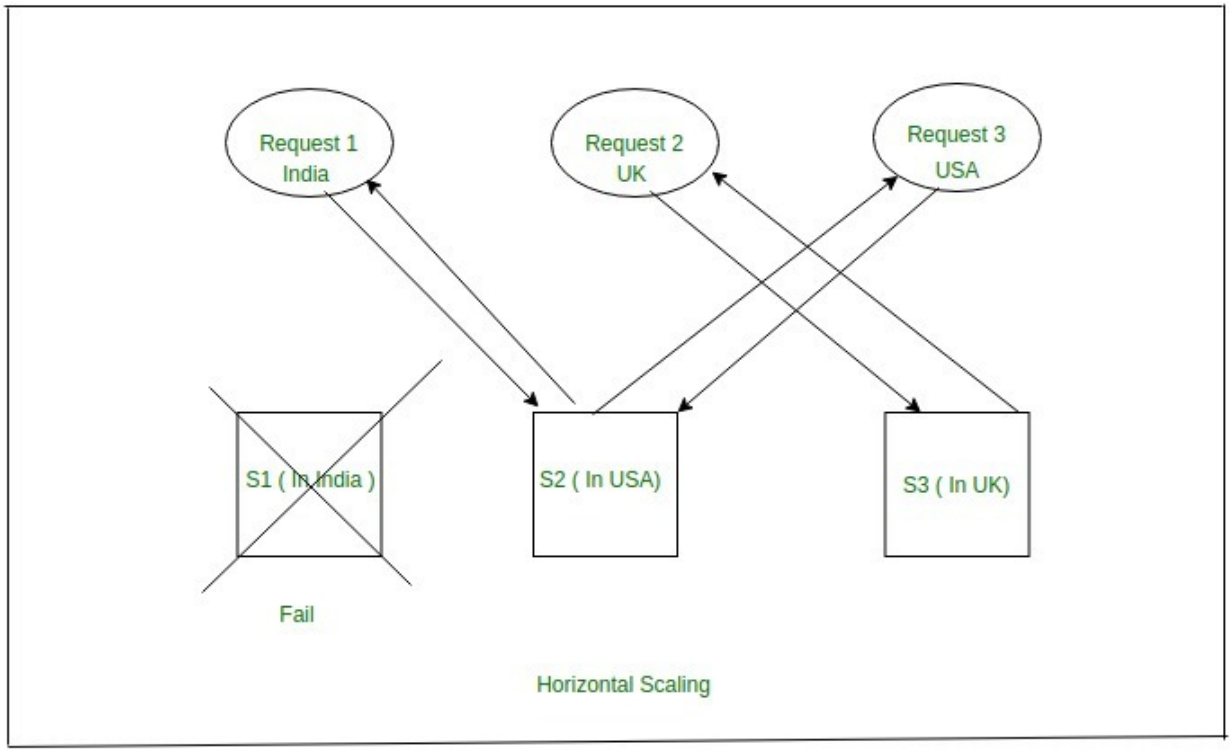
1. **Fault Tolerance** means that there is no single point of failure in this kind of scale because there are 5 servers here instead of 1 powerful server. So if anyone of the servers fails then there will be other servers for backup. Whereas, in **Vertical Scaling** there is single point failure i.e: if a server fails then the whole service is stopped.
2. **Low Latency:** Latency refers to how late or delayed our request is being processed.
3. Built-in backup
4. Google with its Gmail and YouTube, Yahoo, Facebook, eBay, Amazon, etc. are heavily utilizing horizontal scaling.
5. Cassandra and MongoDB is a good example of horizontal scaling.

Disadvantages of Horizontal Scaling

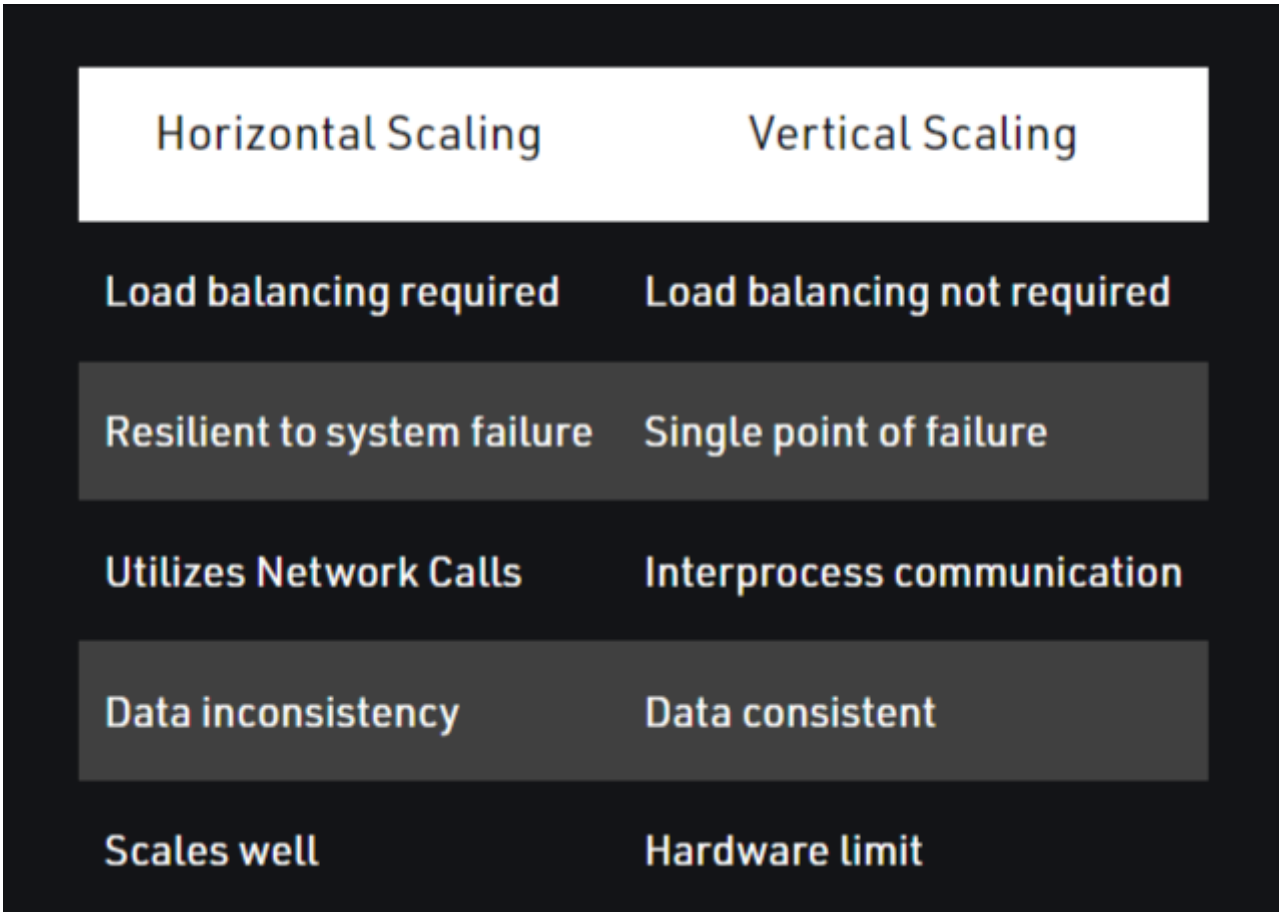
1. Not easy to implement as there are a number of components in this kind of scale
2. Cost is high
3. Networking components like, router, load balancer are required
4.

A Real-time Example of Horizontal Scaling

For example, if there exists a system of the capacity of 8 GB of RAM and in future, there is a requirement of 16 GB of RAM then, rather than the increasing capacity of 8 GB RAM to 16 GB of RAM, similar instances of 8 GB RAM could be used to meet the requirements.



Scaling Comparison



- **Load Balancing:** Horizontal scaling requires load balancing to distribute or spread the traffic among several machines. In the vertical machine, there is just one machine to handle the load so it doesn't require a load balancer.

- **Failure Resilience:** Horizontal scaling is more resistant to system failure. If one of the machines fails you can redirect the request to another machine and the application won't face downtime. This is not in the case of vertical scaling, it has a single machine so it will have a single point of failure. This simply means in horizontal scaling you can achieve availability but in vertical scaling, DB is still running on a single box so it doesn't improve availability.

- **Machine Communication:** Horizontal scaling requires network communication, or calls, between machines. Network calls are slow and more to prone failure. This is not in the case of vertical scaling, Vertical scaling works on inter-process communication that is quite fast.

- **Data Consistency:** Data is inconsistent in horizontal scaling because different machines handle different requests which may lead to their data becoming out of sync which must be addressed. On the other side, vertical machines have just one single machine where all the requests will be redirected, so there is no issue of inconsistency of data in vertical scaling.

- **Limitations:** Depends on budget, space, or requirement you can add as many servers as you want in horizontal scaling and scales your application as much as you want. This is not in the case of vertical scaling. There is a finite limit to the capacity achievable with vertical scaling.

Scaling beyond that capacity results in downtime and comes with an upper limit. So a single machine can only be improved upon until it reaches the current limits of computing.

Which One is Right For an Application?

After a fair understanding of both the options, we can see that both of them have some pros and cons. There will be always some tradeoffs so it may be a little bit trickier for developers to decide which one is better for an application. You need to make a smart decision here. Firstly, you should identify your requirements, business goals, and areas where we would like to add value. Then make important design decisions by questioning ourselves, developing prototypes, and refining the design. Certain factors are important to consider for a better understanding of your business goal or requirement. Some of them are…

- Performance requirements or performance characteristics of an application.
- System throughput
- System response time
- System availability requirement
- Is the system fault-tolerant? If so, what is the degree of it?
- Is the design reliable?
- What level of consistency do we care about?
- What's the scalability goal of the application (you might have some short-term or immediate one's goal, but what is going to happen in the long run ?)
- `If you're curious to know what big tech real companies use when scaling their systems, then the answer is both. Most of the time in big organizations engineers take some good qualities of vertical scaling and some good qualities of horizontal scaling. They follow the hybrid approach of combining the speed and consistency of vertical scaling, with the resilience and infinite scalability of horizontal scaling`
