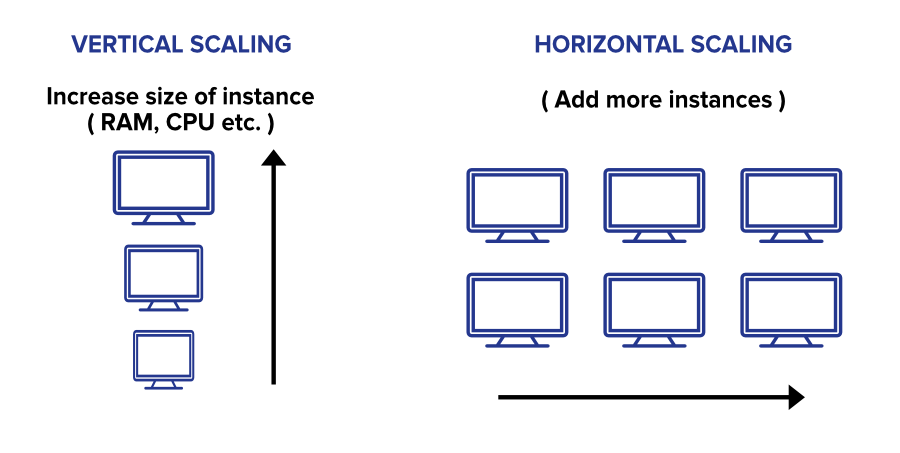
**System Design – Horizontal and Vertical Scaling**

If you’re a developer then building a website and serving it to the users can be the most pleasurable things for you. After all, you worked hard and you tried to create something valuable for the users. When you see a huge number of users have started using your services and day after day the numbers are growing you really feel amazing and you start thinking about taking your services to the next level. You feel good to see the growing numbers but sooner you realize your machine or server can not handle large amounts of requests anymore and it may stop responding anytime. Now you need to find a solution to scale your application for the enormous number of requests. *How would you solve this problem??, what approach you will follow to scale your application??*

It doesn’t matter what company you’re working for or what kind of website you are building, most of the time you will face this kind of scenario, and being a developer it would be your responsibility to find the solution for this problem as well. You will have to expand your app’s accessibility, power, and presence. You can solve this problem by adding extra hardware or by upgrading the current system configuration which is called **scalability**. In this blog, we will talk about scaling an application which is also an important concept of a system design round of interviews. If your aim is to get into a big tech giant company then you should know this concept very well for this round.

Option for scaling your database can be grouped into two major categories…

* Vertical Scaling
* 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.

* 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**.
* Easy implementation.
* Less administrative efforts as you need to manage just one system.
* Application compatibility is maintained.
* Mostly used in small and mid-sized companies.
* MySQL and Amazon RDS is a good example of vertical scaling.

**Drawbacks**

* Limited Scaling.
* Limited potential for improving network I/O or disk I/O.
* Replacing the server will require downtime in this approach.
* Greater risk of outages and hardware failures.
* Finite scope of upgradeability in the future.
* Implementation cost is expensive.

### 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.
* Easy to upgrade
* Instant and continuous availability.
* Easy to size and resize properly to your needs.
* Implementation cost is less expensive compared to scaling-up
* Google with its Gmail and YouTube, Yahoo, Facebook, eBay, Amazon, etc. are heavily utilizing horizontal scaling.
* Cassandra and MongoDB is a good example of horizontal scaling.

**Drawbacks**

* Complicated architectural design
* High licensing fees
* High utility costs such (cooling and electricity)
* The requirement of extra networking equipment such as routers and switches.

### A Short Comparison

We have understood the meaning of both the major categories of scaling an application. We also have discussed some pros and cons of each one of them. Let’s do a quick comparison of these two approaches based on these pros and cons… 

| **Horizontal Scaling** | **Vertical Scaling** |
| --- | --- |
| Load balancing required | Load balancing not required |
| Resilient to system failure | Single point of failure |
| Utilizes Network Calls | Interprocess communication |
| Data inconsistency | Data consistent |
| Scales well | Hardware limit |

* **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 ?)

All the above factors will help you to identify the business goal and requirements for your application. Whatever option you choose, it should ideally be able to answer the above and many other similar questions. You should have a clear understanding of the differences between these 2 scaling approaches. Identify what suits your requirements, and see if the application really fits the model you choose. If your goal is to accomplish superior performance, you can use either vertical scaling or horizontal scaling, or both in a cloud environment.

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.

Simply throwing in new hardware and adding more nodes or machines is not the way to start. You should observe the requirements of your application carefully before making a decision. If the requirements can be met by increasing the capacity or tuning characteristics of a single machine then go with the vertical scaling (especially for start-ups) but once the users start growing rapidly and you need to replace the system quite often, go with horizontal scaling or the combination of both.