#### **Scaling** implies adding or removing resources from your machine/application to perform the underlying tasks with optimal cost and processing capability. Scalability is a challenge that every engineering team needs to go through.

Various scenarios require an application to:

Grow/shrink in requests

Increase/decrease in data

Reduce processing power

At this point, you will have two scaling options:

Horizontal scaling

Vertical scaling

#### **Horizontal scaling** (scaling out)

Horizontal scaling implies adding more machines to the existing system. The data is scattered across multiple machines, and each of them has its own capacity. As we are not modifying existing machines, this process involves less downtime.

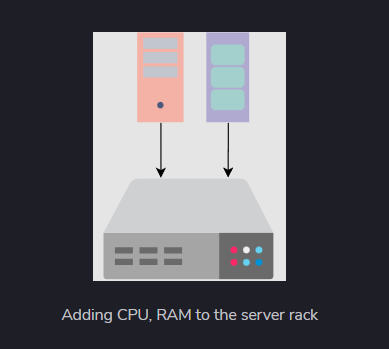
This method allows us to perform distributed programming, which entails distributing jobs across devices. Horizontal scaling can increase the complexity as the address space increases. Therefore, updating and sharing data across machines can be more expensive.



#### 

#### **Vertical scaling** (scaling up)

Vertical scaling implies attaching more resources to the existing machine. Let’s consider a server rack, as before. We add more things like RAM, for example, to the same server rack in this method. The data resides on the same machine and is not distributed as in horizontal scaling. Usually, the activities performed on these machines use multi-threading and in-process data-passing methods. Vertical scaling will have limited capacity within the existing machine. Scaling beyond this capacity will cause downtime.



Links: <https://www.enjoyalgorithms.com/blog/distributed-system-in-system-design/>

<https://www.educative.io/answers/what-is-horizontal-and-vertical-scaling>

#### **Difference between Horizontal and Vertical Scaling**

