Which Scalability Approach is Right for Your App?

In this lesson, you will learn about which type of scaling is best for a given scenario.

We'll cover the following

- Pros and cons of vertical and horizontal scaling
- What about the code? Why does the code need to change when it has to run on multiple machines?
- Which scalability approach is right for your app?

Pros and cons of vertical and horizontal scaling#

This is the part where I talk about the pluses and minuses of both approaches.

Vertical scaling for obvious reasons is simpler in comparison to scaling horizontally because we do not have to touch the code or make any complex distributed system configurations. It takes much less administrative, monitoring, and management efforts as opposed to managing a distributed environment.

A major downside of vertical scaling is availability risk. The servers are powerful but few in number. There is always a risk of them going down and the entire website going offline, which doesn't happen when the

and the entire website going offline, which doesn't happen when the system is scaled horizontally. It becomes more highly available.

What about the code? Why does the code need to change when it has to run on multiple machines?#

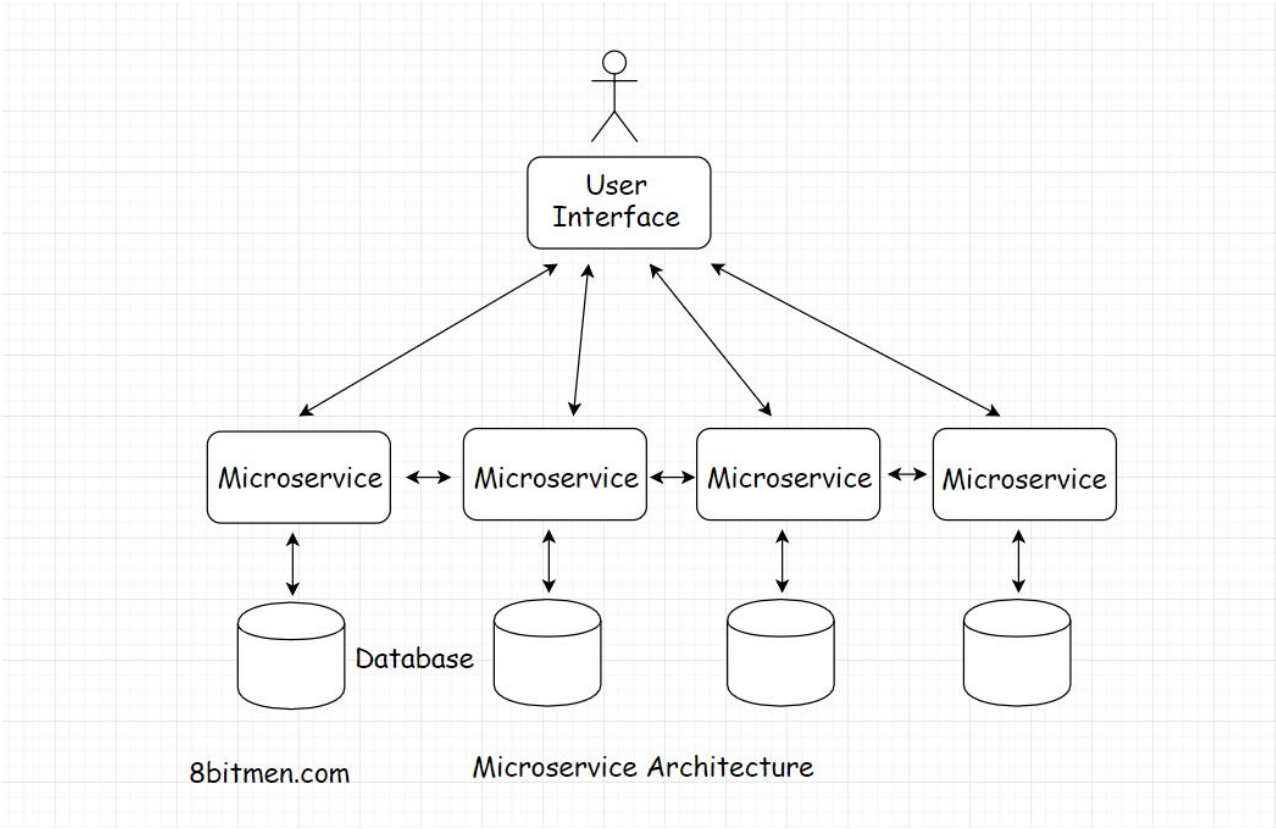
If you need to run the code in a distributed environment, it needs to be *stateless*. There should be no state in the code. **What do I mean by that?**

There can be no *static instances* in the class. Static instances hold application data and if a particular server goes down all the static data/state is lost. The app is left in an inconsistent state.

Rather, use a persistent memory like a *Key-value* store to hold the data and remove all the state/static variable from the class. This is why functional programming became so popular with distributed systems. The functions don't retain any state.

Always have a ballpark estimate on mind when designing your app. **How much traffic will it have to deal with?**

Today, development teams are adopting a distributed micro-services architecture right from the start, and workloads are meant to be deployed on the cloud. So, inherently the workloads are horizontally scaled out on the fly.



The upsides of horizontally scaling include no limit to augmenting the hardware capacity. Data is replicated across different geographical regions as nodes and data centers are set up across the globe.

I'll discuss cloud, serverless, and microservices in the upcoming lessons. So, stay tuned.

Which scalability approach is right for your app?#

If your app is a utility or a tool that is expected to receive minimal consistent traffic, it may not be mission critical. For instance, an internal tool of an organization or something similar.

Why bother hosting it in a distributed environment? A single server is enough to manage the traffic, so go ahead with vertical scaling when you

know that the traffic load will not increase significantly.

If your app is a public-facing social app like a social network, a fitness app, or something similar, then traffic is expected to spike exponentially in the near future. Both high availability and horizontal scalability is important to you.

Build to deploy it on the cloud, and always have horizontal scalability in mind right from the start.

← Back

Types of Scalability

Next →

Primary Bottlenecks That Hurt the Sca...

Mark as Completed

Report an Issue