

[Home](#) / [Resources](#) / [Back to Blog](#)

Vertical vs. horizontal scaling: What's the difference and which is better?



Charlie Custer

Last edited on April 10, 2023 | ⌚ 6 minute read

So you need to scale. That's a good problem to have!



But should you scale up or scale out? There's no easy answer, so let's take a closer look at horizontal scaling vs. vertical scaling, how they compare, and what the pros and cons are for each approach.

What is horizontal vs vertical scaling?



Vertical scaling refers to increasing the capacity of a system by adding capability to the machines it is using (as opposed to increasing the overall number of machines). This is also called *scaling up*.

Vertical scaling can be upgrading the physical machine that's running your system, or it can be swapping over to a different, more capable machine. As long as the *number* of machines our system uses isn't changing, that's scaling vertically.

For example, imagine that we have an application with a cloud database that has reached the limits of the server it's running on: a single 8 vCPU GCP instance with 32 GB of RAM.

To scale that database vertically, we might move it to an instance with 32 vCPUs and 64 GB of RAM. The database is still running on a single server, but the server is more powerful, enabling the database to handle heavier workloads.

What is horizontal scaling?

Horizontal scaling refers to increasing the capacity of a system by adding additional machines (nodes), as opposed to increasing the capability of the existing machines. This is also called *scaling out*.



For example, imagine that we again have an application with a cloud database that has reached the limits of the server it's running on, a single 8 vCPU GCP instance with 32 GB of RAM.


To scale that database horizontally, we might partition it over two additional server nodes, each with 8 vCPUs and 32 GB of RAM. While each machine individually has the same capacity as our original server, we can now spread our workload across these three nodes, which in turn will allow us to handle heavier workloads.

Horizontal scaling made easy | Architectural simplification



“Scaling up” vs. “scaling out” [🔗](#)

Before we get into comparing these two options, it's worth making a quick note on terminology.

“Scaling up” and “scaling out” are technically different things – the former refers to  vertical scaling, the latter refers to horizontal scaling. In the real world, however, these

terms are often used imprecisely. For some people, they are essentially interchangeable.

For that reason, in work-related conversations about scaling, it's worth clarifying what somebody means when they say they want to "scale up" or "scale out."

CONTENT

Vertical scaling vs. horizontal scaling [🔗](#)

What is horizontal scaling?

"Scaling up" vs. "scaling out"

Here's a visual demonstration of the difference between vertical vs. horizontal scaling.

Vertical scaling vs. horizontal scaling

(We're using a database icon for the image here, but the same concept applies to up your application layer, too)

Advantages of vertical scaling

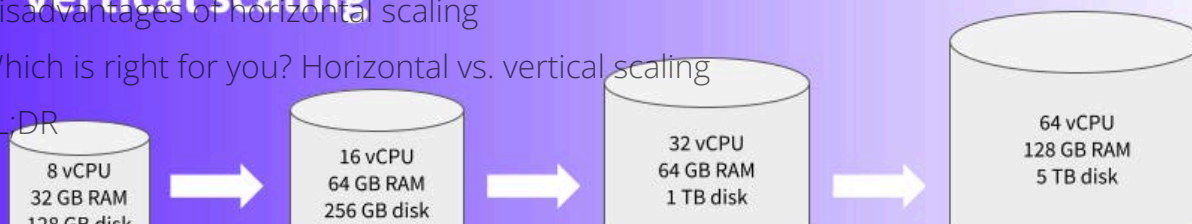
Disadvantages of vertical scaling

Advantages of horizontal scaling

Disadvantages of horizontal scaling

Which is right for you? Horizontal vs. vertical scaling

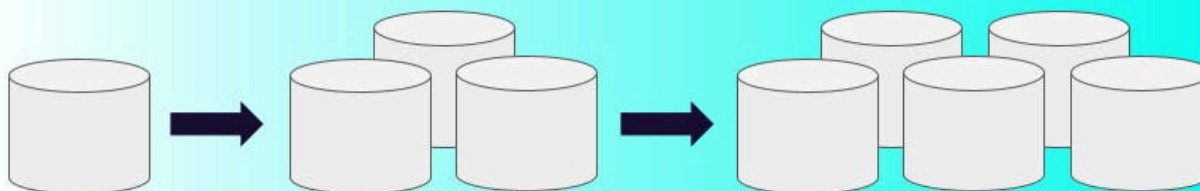
TL;DR



 Cockroach Labs



Horizontal scaling



So which of these approaches is going to be better for you and your workloads? Well, it depends.

Advantages of vertical scaling [🔗](#)

It can sometimes be more cost effective. If you're running an on-prem system, for example, upgrading the components of a server you already own is likely to be cheaper

than buying additional whole servers.

It can sometimes be simpler to work with. Depending on the tools you're using and the approach you take, scaling horizontally can get complicated as you have to work out all the details of how multiple machines will share the load.

It can be easier to maintain. Again, particularly in the case of on-prem systems, it is easier to handle upgrades, maintenance, etc. on a single machine than it is to have to manage multiple machines.

Disadvantages of vertical scaling

You still have a single point of failure. Whether your database is running on 8 vCPUs or 64, if it's all on a single machine and that machine goes down, your entire database – and by extension, most or all of your application – will be offline until you can resolve the problem.

It is inherently limited. Vertical scaling is not infinite; whether you're running on a public cloud or a local machine, you will reach a point where upgrading a single machine's capacity is prohibitively expensive, or straight-up impossible.

It can sometimes cost *more*. While vertical scaling can be more cost effective in some circumstances, the costs can increase rather dramatically as you get closer to the top tier of single-machine capability.

Additionally, vertical scaling can come with “hidden” costs that aren't apparent when upgrading. For example, relying on a system that does have a single point of failure can be very costly. Downtime costs companies \$12,900 per minute on average, according to a 2022 study from analyst firm Enterprise Management Associates (EMA) and AIOps company BigPanda. Those kinds of costs can very quickly wipe out any savings that were associated with opting for vertical scaling to begin with.

Advantages of horizontal scaling

It increases availability and resilience/fault-tolerance. When configured properly, a system that is scaled across multiple machines should be able to survive the loss of one or more of those machines while still operating normally. This means less downtime. In fact, it is possible to come very close to zero downtime with the right horizontally-scaled setup.

It can be more cost-effective. While two or more machines is *generally* more expensive than one in the short term, the increased availability and resilience can quickly pay for themselves by allowing you to avoid the lost business and reputational damage that comes from both planned and unplanned downtime.

It can increase performance. When configured properly, a system that is scaled across multiple machines can route traffic efficiently to spread work evenly and ensure there are no processing bottlenecks. Having multiple nodes also allows for the possibility of multi-region deployments, allowing you to locate services in different geographical locations to place them closest to the users who access them.

Disadvantages of horizontal scaling

It can be more complex. Particularly if you're implementing it manually – for example, **if you're manually sharding a database** – setting up a multi-node system can be quite complex, and it can also add complexity to management and ops work.



Scaling out a traditional RDBMS versus scaling out CockroachDB



However, this complexity can often be avoided by choosing the right tools. In the realm of databases, for example, distributed SQL databases such as CockroachDB handle virtually all of that complexity automatically, under the hood. In some cases, using a tool like CockroachDB can actually lead to *less* work for development and ops teams than what was required when they were managing a single-machine system.

It generally costs more up-front. As previously mentioned, in the long-term horizontal scaling often ends up being more affordable, but the up-front costs tend to be higher since you're paying for two or more machines instead of a single one. If you're scaling out manually rather than using tools that are inherently distributed such as CockroachDB, there will also be costs associated with the setup and management of your manual distribution systems.

Which is right for you? Horizontal vs. vertical scaling



When assessing whether vertical or horizontal scaling is the best choice for your organization, it's important to consider a number of factors, including but not limited to.

- **Costs.** Consider not only the up-front costs of both options, but also the long-term costs. What will managing and operating this system cost? What will an outage cost our organization?
- **Future growth.** If your company is growing quickly, scaling from 32 vCPU machine to a 64vCPU machine may solve your problem, but for how long? For most medium and large companies, there is a point where scaling vertically simply isn't viable.

Similarly, with the right tools, scaling out a multi-node setup can be quite easy. With CockroachDB, for example, this can be as simple as hitting an "Add Node" button. Even in cases where the initial implementation is time-consuming or expensive, it can be worth it in the long run because going from (for example) three nodes to five to seven to dozens doesn't require much additional work.

- **Uptime requirements.** How critical is the system these machines will be running? How available does it need to be? What will it cost if the system becomes unavailable? Mission-critical systems and workloads likely require high availability and thus will be better suited to horizontal scaling; less important workloads may be fine to scale vertically.

(For example, the hardware that's running your internal analytics database may not need five-nines availability. If it goes down for an hour or two, the business impact isn't likely to be massive. On the other hand, if the hardware that's running your *payments* system goes offline, that will have a much higher cost).

- **Performance requirements.** Multi-node systems can be more performant when correctly optimized.
- **Regulatory requirements.** Some locations have specific laws regarding things like geo-locating user data. This may require you to implement a multi-node, multi-region setup to ensure compliance.

TL;DR

Ultimately, the solution that's right for your company is going to involve a lot of company-specific factors, and there are no one-size-fits-all solutions.

In general, though...

Vertical scaling is best for:

- Less important systems and workloads.
- Systems and workloads that aren't likely to need additional scale in the future.
- Keeping initial costs down (but it may increase future costs).
- Systems and workloads that **do not** require high availability, high performance, or multi-region deployment.

Horizontal scaling is best for...

- Mission-critical (tier 0) and important (tier 1) systems and workloads.
- Systems and workloads that are likely to need additional scale in the future as the company grows.
- Systems and workloads that **do** require high availability, high performance, or multi-region deployment.

Have a relational database that you need to scale? CockroachDB is the best solution for mission-critical transactional workloads like payment systems, metadata systems, and much more. [Learn why.](#)

In this short video tutorial you can see what it looks like to scale CockroachDB:

