


GUIDES

What are Vertical and Horizontal Scaling?

19 MIN READ • [Developer Relations Team](#) on SEP 21, 2023 Share

A Head-to-Head Comparison

Vertical Scaling vs. Horizontal Scaling

These two types of scaling aim to increase the capacity and performance of an application, but they differ in their implementation and impact on the system. In the most basic terms, horizontal scaling involves scaling out, while vertical scaling involves scaling up.

Ask AI 



Cost: Vertical scaling often requires more powerful and expensive hardware. In contrast, horizontal scaling allows for using [commodity hardware](#), which can be more cost-effective.

Scalability: Vertical scaling limits how much a single server can handle, while horizontal scaling allows for virtually unlimited scalability by adding more servers to the system.

Flexibility: Vertical scaling requires downtime or service interruption during the hardware upgrade process. However, horizontal scaling can be done without downtime by adding new servers and distributing the workload.

Fault tolerance: Vertical scaling relies on a single server, so the entire system may go down if that server fails. Conversely, with horizontal scaling, the workload is across multiple servers, so if one server fails, the system can still operate with the remaining servers.

Performance: Vertical scaling can perform better for single-threaded or [single-process applications](#), as all resources are concentrated on a single server. However, horizontal scaling can perform better for multi-threaded or distributed applications, as the workload is distributed across multiple servers.

Maintenance: Vertical scaling requires managing and maintaining a single server, which can be more straightforward. Horizontal scaling, on the other hand, requires managing and maintaining multiple servers, which can be more complex and time-consuming. Furthermore, Vertical scaling is typically handled by the systems administrators via software, whereas horizontal scaling is done by techs within the data center.

Elasticity: Horizontal scaling allows for dynamic scaling, meaning servers can be added or removed based on demand. This provides flexibility and cost efficiency



Overall, the choice between vertical and horizontal scaling depends on the specific requirements and constraints of the application. Vertical scaling may be suitable for applications with limited scalability needs, while horizontal scaling is more suitable for applications that require high scalability, fault tolerance, and flexibility.

What are the cost implications of vertical scaling vs. horizontal scaling?

The pricing implications of horizontal vs vertical scaling depend on several factors, such as the application's needs, the scalability requirements, and the available resources. Let's analyze the cost implications of both approaches:

Vertical Scaling:

Hardware costs: Upgrading hardware components can be expensive, especially if the application requires high-performance servers or specialized equipment. The cost of purchasing new hardware or upgrading existing infrastructure must be considered.

Licensing costs: Some software licenses are based on the number of cores or processors. Vertical scaling often involves increasing the number of cores or processors, which may result in additional licensing costs.

Maintenance and support: As the hardware becomes more powerful, it may require specialized maintenance and support. This can lead to increased costs for maintenance contracts or hiring skilled technicians.

Limited scalability: Vertical scaling has limitations in terms of scalability. Eventually, a single machine may reach its maximum capacity, requiring further upgrades or



Horizontal Scaling:

Hardware costs: When scaling horizontally, the cost of adding more machines is generally lower than upgrading individual machines. However, the total cost can still increase if many machines are needed.

Networking costs: Scaling horizontally requires a robust network infrastructure to enable communication and data synchronization between multiple machines. This may involve additional costs for networking equipment, bandwidth, and network management.

Software costs: Depending on the application, scaling horizontally may require additional software licenses for [load balancing](#), distributed file systems, or other necessary components. These costs need to be taken into account.

Complexity and management costs: Managing multiple machines can be more complex than managing one. It may require additional resources for monitoring, configuration management, and troubleshooting, leading to increased management costs.

Scalability and efficiency: Horizontal scaling offers better scalability than vertical scaling. It allows for adding more machines as the workload increases, resulting in better resource utilization and cost efficiency in the long run.

Considerations for choosing between vertical scaling and horizontal scaling

When choosing between vertical and horizontal scaling, several factors must be considered. These include performance requirements, cost-effectiveness, scalability,



Performance Requirements: One of the most important considerations is the performance requirements of your application. Vertical scaling involves adding more resources, such as CPU or memory, to an existing server. This can significantly boost performance for applications with limited scalability needs. On the other hand, horizontal scaling involves adding more servers to distribute the load, which can provide better scalability for high-traffic applications.

Cost-effectiveness: Cost is another important factor to consider. Vertical scaling usually requires upgrading hardware, which can be more expensive than horizontal scaling. On the other hand, horizontal scaling allows for the use of commodity hardware and can be more cost-effective in the long run, especially when dealing with many users.

Scalability: Scalability refers to an application's ability to handle increasing workloads. Vertical scaling has limitations due to the maximum capacity of a single server, whereas horizontal scaling allows for near-linear scalability by adding more servers to the system. If your application needs to handle unpredictable or rapidly increasing workloads, horizontal scaling may be a better choice.

System Complexity: Another consideration is the complexity of the system. Vertical scaling involves adding more resources to an existing server, which is relatively straightforward. However, horizontal scaling requires additional load balancing, data synchronization, and networking complexity. This increased complexity may require additional expertise and maintenance efforts.

Vertical Scaling: A Deep Dive

How does vertical scaling work?

Vertical scaling, also known as scaling up, refers to increasing the capacity of a single server by adding more computing resources, such as CPU, RAM, or storage, to handle



vertical scaling upgrades the existing server with more powerful hardware components or allocates more resources. This allows the server to handle more traffic and perform more tasks simultaneously.

Vertical scaling is typically easier to implement as it involves changing the existing server rather than adding more servers to a system. It requires shutting down the server temporarily to make the hardware changes and then restarting it with the upgraded resources.

What are the advantages and disadvantages of vertical scaling?

Advantages of vertical scaling:

Increased performance: Vertical scaling involves adding more resources (such as CPU, memory, or storage) to a single server. This increases processing power and can improve performance, especially for resource-intensive applications.

Simplicity: Vertical scaling is relatively easier to implement than horizontal scaling, as it does not require complex distributed systems or load-balancing configurations. It involves upgrading the existing hardware or moving to a more powerful server, which is straightforward.

Cost-effective for small-scale applications: Vertical scaling can be a cost-effective solution for small-scale applications that do not experience high traffic or require extensive scalability. It allows for incremental growth by upgrading the existing infrastructure without additional servers.

Easier management: With vertical scaling, managing a single server is simpler than managing a cluster of servers in a horizontally scaled environment. Maintenance

Disadvantages of vertical scaling:

Limited scalability: Vertical scaling has limitations in terms of scalability. Eventually, there will be a limit to the maximum resources a single server can handle. Once that limit is reached, further scaling can only be achieved through horizontal scaling or other architectural changes.

Higher risk of downtime: When performing vertical scaling, there is a higher downtime risk than horizontal scaling. This is because upgrading the hardware or migrating to a new server requires shutting down the system temporarily, which can result in a period of unavailability for the application. This can be problematic for applications that require high availability or operate in real-time environments.

Single point of failure: A single server handles all the application's resources in a vertically scaled environment. This creates a single point of failure, as any issues or failures with that server can cause the entire application to go down. This increases the risk of system failures and can lead to significant downtime.

Cost inefficiency for large-scale applications: While vertical scaling may be cost-effective for small-scale applications, it becomes less efficient and more expensive as the application grows. Adding more resources to a single server can be costly, and there may come a point where it is more cost-effective to implement horizontal scaling instead.

Limited flexibility: Vertical scaling does not offer the same flexibility as horizontal scaling. Adding more servers in a horizontal scaling approach allows for better distribution of resources and workload, making it easier to handle sudden spikes in traffic or scale up and down based on demand. Vertical scaling, on the other hand, relies on the capabilities of a single server, limiting the ability to adapt to changing needs efficiently.

Vertical Scaling Use Cases

While vertical scaling can be beneficial in certain scenarios, it is unsuitable for all applications. The types of applications that are best suited for vertical scaling are those that have the following characteristics:

- 1 **Monolithic applications:** Vertical scaling works well for monolithic applications where the entire application runs on a single server. These applications typically have a **centralized architecture**, and scaling up the server can effectively handle increased loads.
- 2 **Single-threaded applications:** If an application is single-threaded, meaning it can only execute one task at a time, vertical scaling can help by increasing the server's processing power. This is particularly useful for applications that involve heavy computational tasks or data processing.
- 3 **Applications with limited scalability requirements:** If an application does not have high scalability requirements and is not expected to experience sudden spikes in traffic, vertical scaling can be a cost-effective solution. Adding more resources to a single server can provide the necessary performance boost without the need for complex distributed systems.
- 4 **Resource-intensive applications:** Vertical scaling is suitable for resource-intensive applications that require a significant amount of CPU or memory resources. Applications such as machine learning, scientific simulations, or video rendering can benefit from vertical scaling as it allows them to leverage a single server's increased processing power and memory capacity.
- 5 **Applications with strict data locality requirements:** Vertical scaling is a good fit for applications that require proximity of data and computation. By increasing the resources of a single server, data can be stored and processed locally, **reducing network latency** and improving overall performance.



There are several ways vertical scaling can improve system performance:

Increased Processing Power: The server can handle more concurrent requests and perform computations faster by upgrading the CPU or adding more cores. This is particularly beneficial for applications that require heavy computational tasks, such as data processing or complex algorithms.

Enhanced Memory Capacity: Increasing the RAM allows the server to store and access more data in memory, reducing the need for disk-based operations. This can significantly improve the overall performance of applications that require frequent data retrieval or manipulation.

Improved Storage Capacity: Upgrading the server's storage capacity allows more data to be stored locally, reducing reliance on external storage systems. This leads to faster data access and retrieval, especially for applications that handle large amounts of data.

Better Scalability: Vertical scaling can handle sudden spikes in traffic or increased workloads by allocating more resources to the server. This flexibility ensures the system can accommodate growth without experiencing performance degradation or downtime.

Simplified Management: Scaling up involves adding more resources to an existing server, simplifying the management and maintenance process compared to managing a cluster of multiple servers. This reduces the complexity and cost associated with managing a distributed system.

It's important to note that vertical scaling has its limitations, however. There is a maximum capacity to which a single server can be scaled. Once that limit is reached, further improvement in system performance can only be achieved through horizontal scaling.



When implementing vertical scaling, there are several key considerations to remember. Here are some of the most important ones:

Cost: Vertical scaling typically involves upgrading the hardware or resources of a single server, which can be expensive. Consider the cost implications and ensure that the benefits gained from vertical scaling justify the investment.

Hardware limitations: Vertical scaling is limited by the maximum capacity of a single server. Before implementing vertical scaling, assess whether the server can handle the increased workload. Consider factors such as CPU, memory, storage, and network capabilities.

Downtime: Vertical scaling often requires taking the system offline or experiencing downtime during the upgrade process. Plan this downtime and communicate it to users or customers to minimize disruption.

Scalability limitations: Vertical scaling may have limitations regarding how much the system can scale. Eventually, the hardware may reach its maximum capacity, and further vertical scaling may not be possible. Consider the long-term scalability needs of your system and whether vertical scaling will meet those needs.

Performance bottlenecks: While vertical scaling increases the resources of a single server, it does not address potential performance bottlenecks within the system architecture. Analyze the system architecture to identify potential bottlenecks and optimize for optimal performance.

Compatibility: When upgrading the hardware or resources of a server, ensure that all software and applications running on the server are compatible with the new configuration. Compatibility issues can cause disruptions and potential downtime,

Horizontal Scaling: A Deep Dive

How does horizontal scaling work?

Horizontal scaling is a method used to increase the capacity and performance of an application by adding more machines or servers to the existing infrastructure. It involves distributing the workload across multiple machines, allowing the system to handle larger traffic volumes or process more data.

The basic principle behind horizontal scaling is to add more machines to the system and divide the workload among them rather than relying on a single machine to handle all the requests. This approach offers several benefits, including improved performance, increased availability, and better fault tolerance.

When a system is horizontally scaled, it typically consists of multiple instances of the application running on different machines or servers. These instances are connected to a load balancer, which evenly distributes incoming requests across the available instances based on predefined algorithms such as round-robin or weighted distribution.

As the number of machines or servers increases, the system can handle more requests simultaneously. This results in improved performance and reduced response times, as the workload is distributed across multiple resources.

To ensure that the workload is evenly distributed, the load balancer monitors the health and performance of each instance and dynamically adjusts the distribution of requests. If an instance becomes overloaded or fails, the load balancer automatically redirects the traffic to other healthy instances, ensuring high availability and minimizing downtime.



Advantages of Horizontal Scaling:

Improved Performance: Horizontal scaling allows workload distribution across multiple servers or machines. This results in improved performance as the workload is divided among multiple resources, reducing the chances of bottlenecks or overloading.

High Availability: By distributing the workload across multiple servers, horizontal scaling enhances the system's availability. If one server fails, the load can be seamlessly transferred to other servers, ensuring uninterrupted service.

Cost Efficiency: Horizontal scaling provides a cost-effective solution for growing traffic or resource demands. Instead of investing in expensive high-end hardware, organizations can add low-cost commodity servers to scale horizontally, reducing upfront infrastructure costs.

Flexibility and Scalability: Horizontal scaling offers flexibility and scalability to meet changing business needs. It allows organizations to easily add or remove servers based on demand, providing the ability to scale up or down as required. This enables businesses to adapt to fluctuating traffic patterns and handle sudden spikes in user activity.

Disadvantages of Horizontal Scaling:

Complexity of Distributed Systems: Horizontal scaling introduces complexities associated with distributed systems. Managing multiple servers, ensuring data consistency, and coordinating inter-server communication can be challenging. It requires additional expertise and effort to design and maintain a distributed architecture.



overhead due to the need for inter-server communication. Network latency and bandwidth constraints may impact the overall performance of the system.

Data Consistency: Maintaining data consistency across multiple servers can be challenging in horizontally scaled systems. Synchronization and replication mechanisms must be in place to ensure all servers have consistent and up-to-date data. Failure to achieve data consistency can lead to issues like data corruption or conflicts.

Single Point of Failure: While horizontal scaling enhances availability, it also introduces the risk of a single point of failure. If the load balancer, which distributes the workload among servers, fails, the entire system may become inaccessible. Implementing redundancy and failover mechanisms can mitigate this risk but adds complexity to the system architecture.

Resource Utilization: Horizontal scaling may not always utilize resources efficiently. If the workload is not evenly distributed among servers, some servers may be underutilized while others are overloaded. Proper load-balancing algorithms and monitoring systems must be implemented to optimize resource utilization.

Horizontal scaling use cases

While horizontal scaling can benefit various applications, certain applications are especially well-suited for this approach.

Web Applications: Web applications that serve many concurrent users can greatly benefit from horizontal scaling. By distributing the incoming requests across multiple servers, horizontal scaling helps to handle increased traffic and prevent performance bottlenecks. This is particularly useful for [e-commerce](#), [social media](#), and content-heavy websites.



responsiveness during high-traffic periods by horizontally scaling the backend infrastructure.

Real-Time Applications: Applications that require real-time data processing, such as [chat applications](#), [collaboration tools](#), and [online gaming platforms](#), can greatly benefit from horizontal scaling. By distributing the processing load across multiple servers, real-time applications can handle many concurrent connections and ensure low latency.

Big Data Processing: Applications that deal with large volumes of data, such as analytics platforms, [data streaming services](#), and recommendation engines, can use horizontal scaling to process and analyze data in parallel. This allows for faster processing times and improved scalability as the data volume increases.

Content Delivery Networks (CDNs): [CDNs](#) are designed to distribute content geographically close to end users, reducing latency and improving performance. Horizontal scaling is crucial for CDNs as they must handle heavy traffic loads and distribute content efficiently across multiple servers worldwide. Horizontal scaling is particularly suited for these applications because it improves performance, scalability, and reliability. By distributing the workload across multiple servers, horizontal scaling helps to handle increased traffic, prevent performance bottlenecks, and ensure high availability.

Can horizontal scaling improve system performance?

Adding more identical servers to the system increases the overall capacity and processing power, resulting in better performance.

One way horizontal scaling improves system performance is by allowing the system to handle more concurrent requests. As the workload is distributed among multiple



more consistent user experience.

In addition, horizontal scaling can also improve system performance by enabling elastic scaling. [Elastic scaling](#) automatically allows the system to adjust its resources based on the current demand. When the workload increases, additional servers can be added to handle the increased load. Similarly, excess servers can be removed when the demand decreases, saving resources and costs. This flexibility ensures the system can always meet the required performance levels, even during peak times.

Key considerations when implementing horizontal scaling

When implementing horizontal scaling, there are several key considerations to remember. Here are some of the most important ones:

Load balancing: Horizontal scaling involves distributing the workload across multiple servers. To ensure efficient load balancing, consider implementing a load balancer that evenly distributes incoming requests among the servers. This helps prevent any single server from becoming overloaded and ensures optimal performance.

Communication between servers: As the workload is distributed across multiple servers, efficient communication between these servers becomes crucial. Choose a communication protocol or mechanism that allows seamless data exchange and synchronization between servers. This can be achieved through technologies like message queues or distributed databases.

Data consistency: With horizontal scaling, data may be distributed across multiple servers. Ensuring data consistency becomes essential to prevent data



Fault tolerance: Horizontal scaling inherently increases the number of servers in the system. This provides an opportunity to implement fault-tolerant mechanisms. Consider implementing redundancy, failover mechanisms, and backup strategies to ensure high availability and minimize the impact of server failures.

Scalability limitations: While horizontal scaling allows for adding more servers to handle increased workload, there may still be limitations in terms of scalability. Consider the scalability limits of your system and evaluate whether horizontal scaling will meet the long-term growth needs of your application. It is important to monitor and analyze the performance of your system to identify any scalability bottlenecks and make necessary adjustments.

Application architecture: The architecture of your application plays a crucial role in the success of horizontal scaling. Consider designing your application to be [stateless](#), where each request can be processed independently without relying on previous requests. This allows for easy distribution of workload across multiple servers. Adopting a microservices architecture can further enable horizontal scaling by breaking down your application into smaller, independent services that can be scaled individually.

Monitoring and alerting: With multiple servers in a horizontally scaled system, monitoring becomes critical to ensure optimal performance and identify any issues or bottlenecks. Implement monitoring tools and set up alerts for abnormal behavior or performance degradation. This allows proactive identification and resolution of issues before they impact the user experience.

Cost considerations: Horizontal scaling typically involves adding more servers, which can increase infrastructure and operational costs. Consider the cost implications of scaling horizontally and evaluate whether it aligns with your budget and long-term growth plans. Additionally, consider implementing efficiency

Scaling and Kubernetes

Kubernetes is an open-source [container orchestration](#) platform that can help developers scale their applications. By leveraging [Kubernetes](#), developers can easily manage and scale their application resources, ensuring optimal performance and efficient resource allocation.

One of the key advantages of using Kubernetes for scaling is its ability to automatically distribute the workload across multiple [containers](#) or pods. This horizontal scaling capability allows developers to dynamically add or remove containers based on the application's demand. As the user load increases, Kubernetes can automatically spin up additional containers to handle the increased traffic, ensuring a smooth and uninterrupted user experience.

Additionally, Kubernetes provides a built-in load-balancing mechanism that evenly distributes network traffic across the containers. This ensures that each container receives a fair share of the workload and prevents any container from becoming overwhelmed. Load balancing is essential for high availability and preventing bottlenecks or performance issues.

Another advantage of using Kubernetes for scaling is its integration with cloud providers and its ability to leverage auto-scaling features. With cloud providers such as [Amazon Web Services \(AWS\)](#) or [Google Cloud Platform \(GCP\)](#), developers can configure Kubernetes to automatically scale the number of containers based on predefined metrics, such as CPU utilization or network traffic. As the application's demand increases, Kubernetes can automatically spin up additional containers to handle the increased workload. As the demand decreases, Kubernetes can scale down the number of containers to save resources and reduce costs. This auto-scaling feature ensures that the application always has the necessary resources to meet the



By leveraging Kubernetes, developers can easily scale their applications to meet growing user demands while reducing maintenance efforts and costs. Whether it's horizontal scaling or vertical scaling, Kubernetes offers the necessary tools and features to efficiently manage application scalability.

With over 15 points of presence worldwide supporting 800 million monthly active users and 99.999% reliability, you'll never have to worry about outages, concurrency limits, or any [latency issues](#) caused by traffic spikes. PubNub is [programming language-agnostic](#) and is perfect for any application that requires real-time data.

[Sign up for a free trial](#) and get up to 200 MAUs or 1M total transactions per month included. You can also check out our [GitHub](#), [docs](#), or [tutorials](#) to learn more.

ABOUT

- [Our Company](#)
- [Careers](#)
- [Support](#)
- [Partners](#)
- [News and Press](#)
- [Privacy Policy](#)
- [Terms and Conditions](#)
- [Bug Bounty Policy](#)
- [Contact Us](#)
- [Cookies Policy](#)
- [Cookies Settings](#)

USE CASE

- [Chat](#)

PUBNUB PLATFORM

- [Overview](#)
- [In-App Messaging](#)
- [Mobile Push Notifications](#)
- [Functions](#)
- [Generative AI](#)
- [BizOps Workspace](#)
- [File Sharing](#)
- [Chat SDK](#)
- [Edge Messaging Network](#)
- [Security & Compliance](#)
- [Pricing](#)

PUBNUB ILLUMINATE

- [Illuminate Overview](#)
- [Illuminate for Gaming](#)

RESOURCES

- [Customers](#)