Design for performance and scalability

10 minutes

Imagine a news story has just been published covering your organization's breakthrough cancer treatment. This is a terrific milestone, and will undoubtedly bring a large influx of traffic to your website. Will the website handle this traffic increase, or will the load cause the site to be slow or unresponsive?

Here, we'll look at some of the basic principles of ensuring outstanding application performance using scaling and optimization principles.

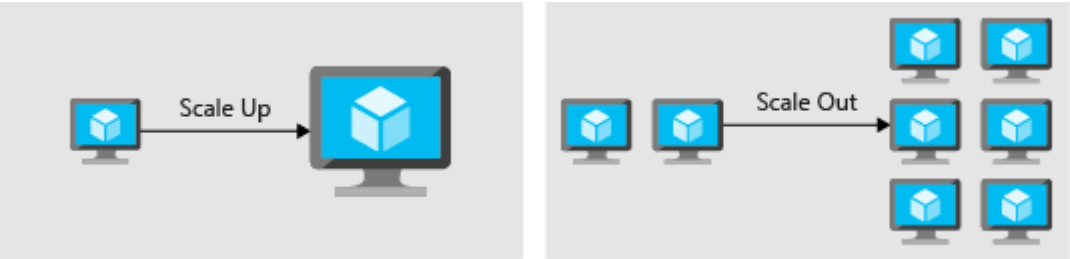
What is scaling and performance optimization?

Scaling and performance optimization are about matching the resources available to an application with the demand it is receiving. Performance optimization includes scaling resources, identifying and optimizing potential bottlenecks, and optimizing your application code for peak performance.

Scaling

Compute resources can be scaled in two different directions:

- Scaling *up* is the action of adding more resources to a single instance.
- Scaling *out* is the addition of instances.



Scaling up is concerned with adding more resources, such as CPU or memory, to a single instance. This instance could be a virtual machine or a PaaS service. The act of adding more capacity to the instance increases the resources available to your application, but it does come with a limit. Virtual machines are limited to the capacity of the host they run on, and hosts

themselves have physical limitations. Eventually, when you scale up an instance, you can run into these limits, restricting your ability to add further resources to the instance.

Scaling out is concerned with adding additional instances to a service. These can be virtual machines or PaaS services, but instead of adding more capacity by making a single instance more powerful, we add capacity by increasing the overall total number of instances. The advantage of scaling out is that you can conceivably scale out forever if you have more machines to add to the architecture. Scaling out requires some type of load distribution. This could be in the form of a load balancer distributing requests across available servers, or a service discovery mechanism for identifying active servers to send requests to.

In both cases, resources can be reduced, bringing cost optimization into the picture.

Performance optimization

When optimizing for performance, you'll look at network and storage to ensure performance is acceptable. Both can impact the response time of your application. Selecting the right networking and storage technologies for your architecture will help you ensure you're providing the best experience for your consumers.

Performance optimization will also include understanding how the applications themselves are performing. Errors, poorly performing code, and bottlenecks in dependent systems can all be uncovered through an application performance management tool. Often, these issues may be hidden or obfuscated for end users, developers, and administrators, but can have adverse impact on the overall performance of your application.

Scalability and performance patterns and practices

Let's take a look at some patterns and practices that can be leveraged to enhance the scalability and performance of your application.

Data partitioning

In many large-scale solutions, data is divided into separate partitions that can be managed and accessed separately. The partitioning strategy must be chosen carefully to maximize the benefits while minimizing adverse effects. Partitioning can help improve scalability, reduce contention, and optimize performance.

Caching

Use caching in your architecture can help improve performance. Caching is a mechanism to store frequently used data or assets (web pages, images) for faster retrieval. Caching can be used at different layers of your application. You can use caching between your application

used at different layers of your application. You can use caching between your application servers and a database, to decrease data retrieval times. You could also use caching between your end users and your web servers, placing static content closer to the user and decreasing the time it takes to return web pages to the end user. This also has a secondary effect of offloading requests from your database or web servers, increasing the performance for other requests.

Autoscaling

Autoscaling is the process of dynamically allocating resources to match performance requirements. As the volume of work grows, an application may need additional resources to maintain the desired performance levels and satisfy service-level agreements (SLAs). As demand slackens and the additional resources are no longer needed, they can be de-allocated to minimize costs.

Autoscaling takes advantage of the elasticity of cloud-hosted environments while easing management overhead. It reduces the need for an operator to continually monitor the performance of a system and make decisions about adding or removing resources.

Decouple resource-intensive tasks as background jobs

Many types of applications require background tasks that run independently of the user interface (UI). Examples include batch jobs, intensive processing tasks, and long-running processes such as workflows. Background jobs can be executed without requiring user interaction--the application can start the job and then continue to process interactive requests from users. This can help to minimize the load on the application UI, which can improve availability and reduce interactive response times.

Use a messaging layer between services

Adding a messaging layer in between services can have a benefit to performance and scalability. Adding a messaging layer creates a buffer for requests between the services so that requests can continue to flow in without error if the application can’t keep up. As the application works through the requests, they will be answered in the order in which they were received.

Implement scale units

Scale as a unit. For each resource, determine the impact that a scaling activity may have on dependent systems. This makes applying scale-out operations easier, and less prone to negative impact on the application. For example, adding x number of web and worker roles might require y number of additional queues and z number of storage accounts to handle the

additional workload generated by the roles. A scale unit could consist of x web and worker roles, y queues, and z storage accounts. Design the application so that it's easily scaled by adding one or more scale units.

Performance monitoring

Distributed applications and services running in the cloud are, by their nature, complex pieces of software that comprise many moving parts. In a production environment, it's important to be able to track the way in which users utilize your system, trace resource utilization, and generally monitor the health and performance of your system. You can use this information as a diagnostic aid to detect and correct issues, and also to help spot potential problems and prevent them from occurring.

Look across all layers of your application and identify and remediate performance bottlenecks in your application. These bottlenecks could be poor memory handling in your application, or even the process of adding indexes into your database. It may be an iterative process as you relieve one bottleneck and then uncover another that you were unaware of.

With a thorough approach to performance monitoring, you'll be able to determine what types of patterns and practices your architecture will benefit from.

Check your knowledge

1. Which of the following is an example of scaling up?

- ○ Updating your application to use a queuing service
- ○ Adding more web servers into a web farm
- ○ Adding another virtual machine into a database cluster
- ○ Updating a virtual machine to a larger size ✓

Channging a virtual machine to a larger size increases the resources available to the instance, and is an example of scaling up.

2. Which of the following is an example of scaling out?

- ○ Updating a virtual machine to a larger size
- ○ Adding more storage to a virtual machine
- ○ Adding more web servers into a web farm ✓

Adding more web servers into a web farm is an example of scaling out.

- ○ Replicating backups to another region

- Replicating backups to another region

Next: Design for availability and recoverability

Continue >