Partition around limits

08/30/2018 • 2 minutes to read • Contributors

In this article

Use partitioning to work around database, network, and compute limits

In the cloud, all services have limits in their ability to scale up. Azure service limits are documented in Azure subscription and service limits, quotas, and constraints. Limits include number of cores, database size, query throughput, and network throughput. If your system grows sufficiently large, you may hit one or more of these limits. Use partitioning to work around these limits.

There are many ways to partition a system, such as:

- Partition a database to avoid limits on database size, data I/O, or number of concurrent sessions.
- Partition a queue or message bus to avoid limits on the number of requests or the number of concurrent connections.
- Partition an App Service web app to avoid limits on the number of instances per App Service plan.

A database can be partitioned *horizontally*, *vertically*, or *functionally*.

- In horizontal partitioning, also called sharding, each partition holds data for a subset of the total data set. The partitions share the same data schema. For example, customers whose names start with A–M go into one partition, N–Z into another partition.
- In vertical partitioning, each partition holds a subset of the fields for the items in the data store. For example, put frequently accessed fields in one partition, and less frequently accessed fields in another.
- In functional partitioning, data is partitioned according to how it is used by each bounded context in the system. For example, store invoice data in one partition and product inventory data in another. The schemas are independent.

For more detailed guidance, see Data partitioning.

Recommendations

Partition different parts of the application. Databases are one obvious candidate for partitioning, but also consider storage, cache, queues, and compute instances.

Design the partition key to avoid hot spots. If you partition a database, but one shard still gets the majority of the requests, then you haven't solved your problem. Ideally, load gets distributed evenly across all the partitions. For example, hash by customer ID and not the first letter of the customer name, because some letters are more frequent. The same principle applies when partitioning a message queue. Pick a partition key that leads to an even distribution of messages across the set of queues. For more information, see Sharding.

Partition around Azure subscription and service limits. Individual components and services have limits, but there are also limits for subscriptions and resource groups. For very large applications, you might need to partition around those limits.

Partition at different levels. Consider a database server deployed on a VM. The VM has a VHD that is backed by Azure Storage. The storage account belongs to an Azure subscription. Notice that each step in the hierarchy has limits. The database server may have a connection pool limit. VMs have CPU and network limits. Storage has IOPS limits. The subscription has limits on the number of VM cores. Generally, it's easier to partition lower in the hierarchy. Only large applications should need to partition at the subscription level.