Database Scaling

Do you have an application with a growing user base, or do you have an application that you anticipate will grow in the future? If so, then the load on your database is most likely growing as your application saves larger amounts of data. Whether it's the number of connections needed, the amount of data to be stored, or the increased processing power, any database will eventually hit a limit on what it can handle.

Table of contents

What is database scalability?

Database scalability is the ability to expand or contract the capacity of system resources in order to support the changing usage of your application. This can refer both to increasing

and decreasing usage of the application. Increased usage of your application brings three main challenges to your database server:

1. The CPU and/or memory becomes overloaded, and the database server either cannot respond to all the request throughput or do so in a reasonable amount of time.
2. Your database server runs out of storage, and thus cannot store all the data.
3. Your network interface is overloaded, so it cannot support all the network traffic received.

The first action you might take to address the need for increased capacity is application and database *optimization*. Examples include optimizing the application code, caching, and appropriately indexing your query patterns. These optimizations increase the efficiency of your application and should bring some relief. However, there comes a point when system resource limits are reached. At this point, you will want to consider *scaling* your database vertically, horizontally, or both.

What is vertical and horizontal scaling?

Vertical scaling refers to increasing the processing power of a single server or cluster. Both relational and non-relational databases can scale up, but eventually, there will be a limit in terms of maximum processing power and throughput. Horizontal scaling, also known as scale-out, refers to bringing on additional nodes to share the load.

MongoDB Atlas makes it simple to vertically or horizontally scale up or down as needed. You can even enable auto-scaling so your available resources always meet your needs.

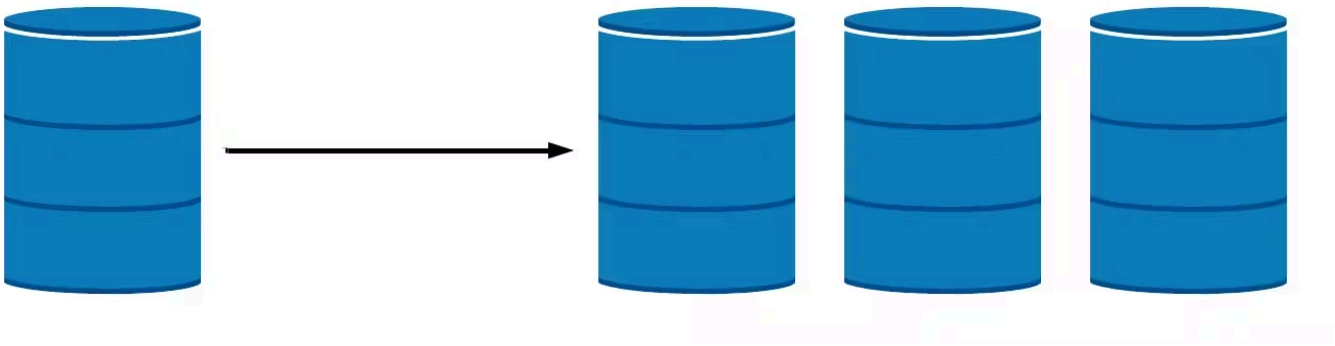
How to increase database performance and make it more scalable

There are a variety of scaling techniques that depend on the database system and what components are used. However, they all use the concept of a **node**, which is an individual machine storing some or all of the data. A group of nodes that work together is called a cluster.

There are two commonly used horizontal database scaling techniques: **replication** and **horizontal partitioning** (or **sharding**). MongoDB is a modern, document-based database that supports both of these.

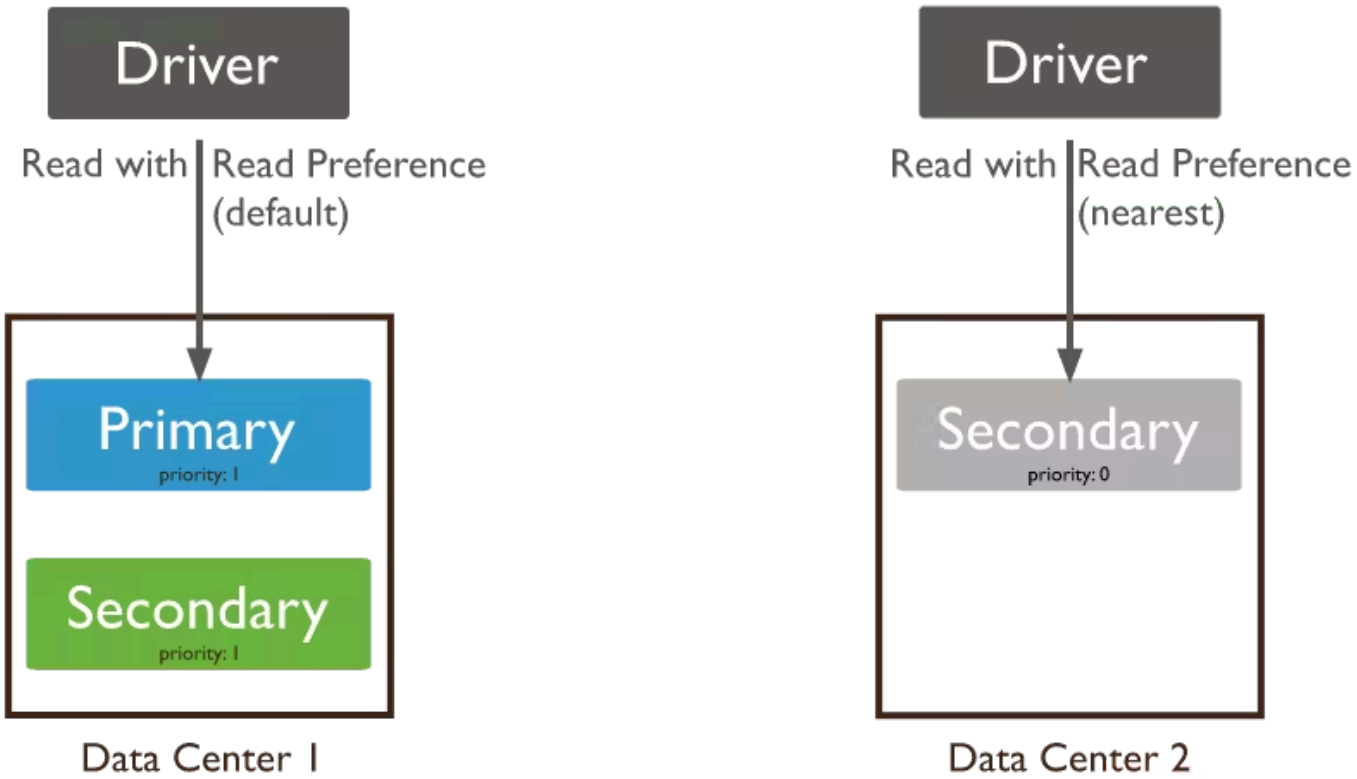
Replication

Replication refers to creating copies of a database or database node. Replication adds fault tolerance to a system. Each node in a cluster contains a copy of the data. If one of the nodes goes down, the cluster is still able to serve client requests because the other nodes in the cluster can respond to the requests.



Replication is also a form of scaling because client requests can be spread across all the nodes in the cluster instead of overwhelming a single node. This increases the capacity of the system to handle more database read requests.

In MongoDB, a set of replicated nodes is called a replica set. One of the nodes in a replica set is the primary node, and the other nodes are secondary nodes. Read requests are distributed between each of the nodes. However, only the primary node can be written to, and updates made to the primary node are then replicated to the other nodes.



Replication is included by default in MongoDB Atlas, providing higher availability and fault tolerance without any additional complexity or setup work.

Replication increases neither the total storage capacity of the system nor its ability to handle write requests. For these, we will need to look to **partitioning**.

Partitioning (aka sharding)

Partitioning distributes data across multiple nodes in a cluster. Each replica set (known in MongoDB as a shard) in a cluster only stores a portion of the data based on a collection sharding key (sharding strategy), which determines the distribution of the data. This makes it possible to scale the storage capacity of the cluster virtually without limit. Since each node is only responsible for processing the data it stores, overall processing capacity for both reads and writes is increased as well.

For example, in the illustration below, the data shard data subsets were divided by price range.

However, partitioning is a more complex scaling strategy than replication. Because each node only stores part of the data, for each request, the database queries need to determine which node or nodes contain the relevant data. In MongoDB, the client application connects to a sharded cluster through a router that directs the requests to the appropriate nodes.

If the data is stored across multiple nodes, the reads and writes could be done in parallel. For large-volume data reads, performance is improved because each node can read its

section of data in parallel with the other nodes.

There is an overhead to reading from multiple nodes. The data from all the nodes still needs to be transferred over the network and then combined into a query result set. For small data reads, the network latency could be a significant portion of the overall query time. For those scenarios, it's more efficient to query using targeted operations.

MongoDB has the ability to store both sharded and unsharded collections in a sharded cluster. This allows the application to take full advantage of the cluster for large data sets while using a primary shard for small data sets.

The easiest, most convenient, and most cost-effective way to deploy and manage a sharded cluster is via MongoDB Atlas, the Developer Data Platform that simplifies sharded cluster implementation.

MongoDB Atlas scaling

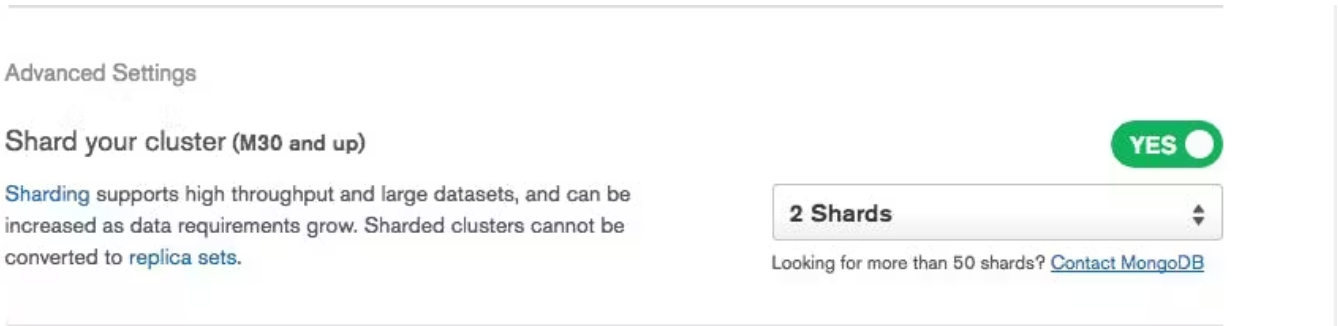
As a service offering, MongoDB Atlas makes scaling as easy as setting the right configuration. Both horizontal and vertical scaling are supported.

Vertical scaling is as simple as configuring a cluster tier. Note that even within a tier, further scaling is possible (including auto-scaling from the M10 tier upwards). We'll look at that later.

Horizontal scaling comes through the deployment of a sharded cluster.

Deploying a sharded cluster

In MongoDB, a sharded cluster consists of shards, routers/balancers, and config servers for metadata. While setting this up manually would require some infrastructure setup and configuration, Atlas makes this quite simple. Just toggle the option on for your MongoDB cluster and select the number of shards.



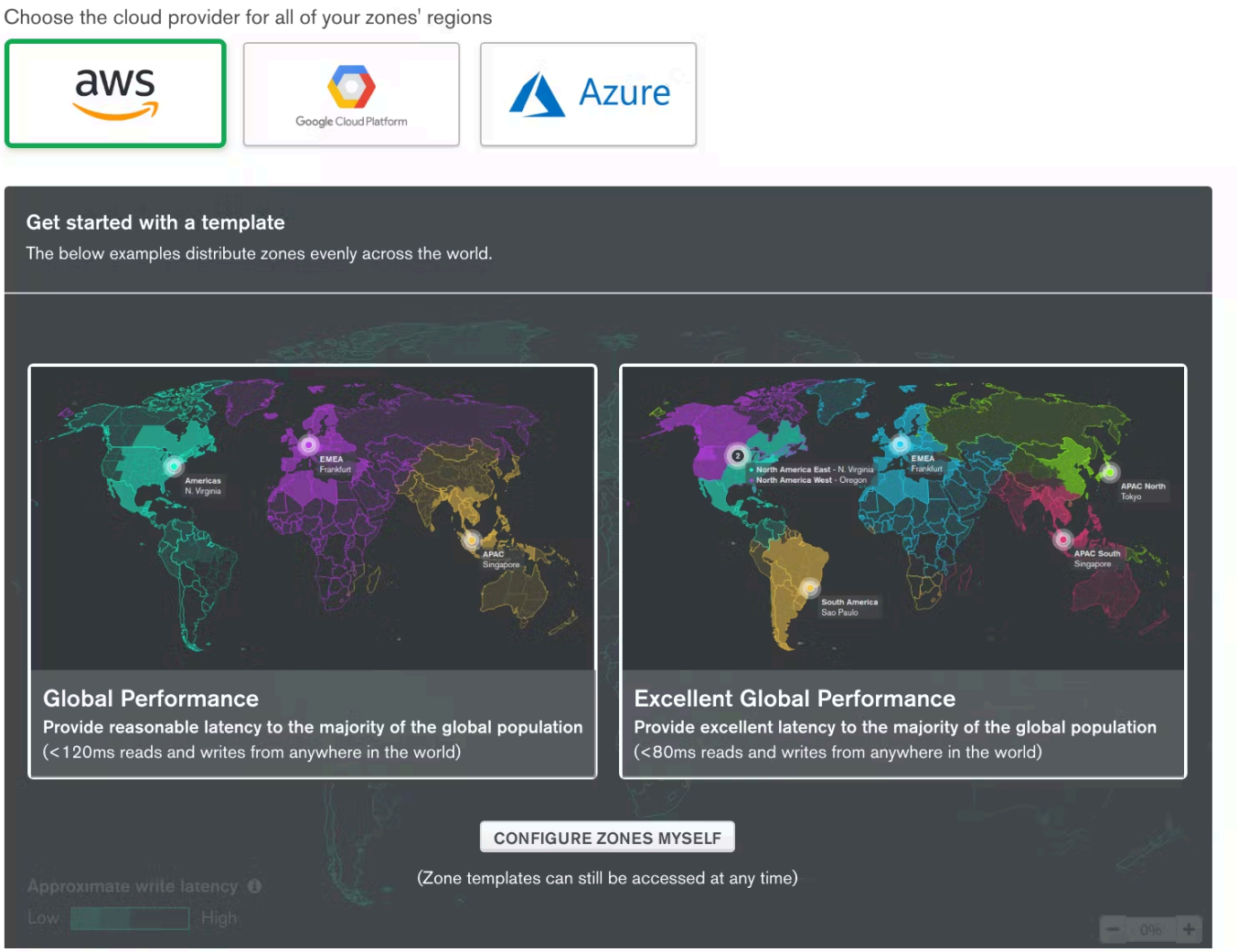
(Please note: This is only available for M30 clusters and up.)

The default setup creates replica sets and mongods for each of the shards and the config servers. This provides high availability, redundancy, and increased read and write performance through the use of both types of horizontal scaling. The routers, or mongos, distribute queries and write operations across the shards according to the data which is on that shard.

Don't forget, a shard key needs to be configured, and there are a few different options available. For more information, see the MongoDB documentation on shard keys.

MongoDB Atlas global clusters

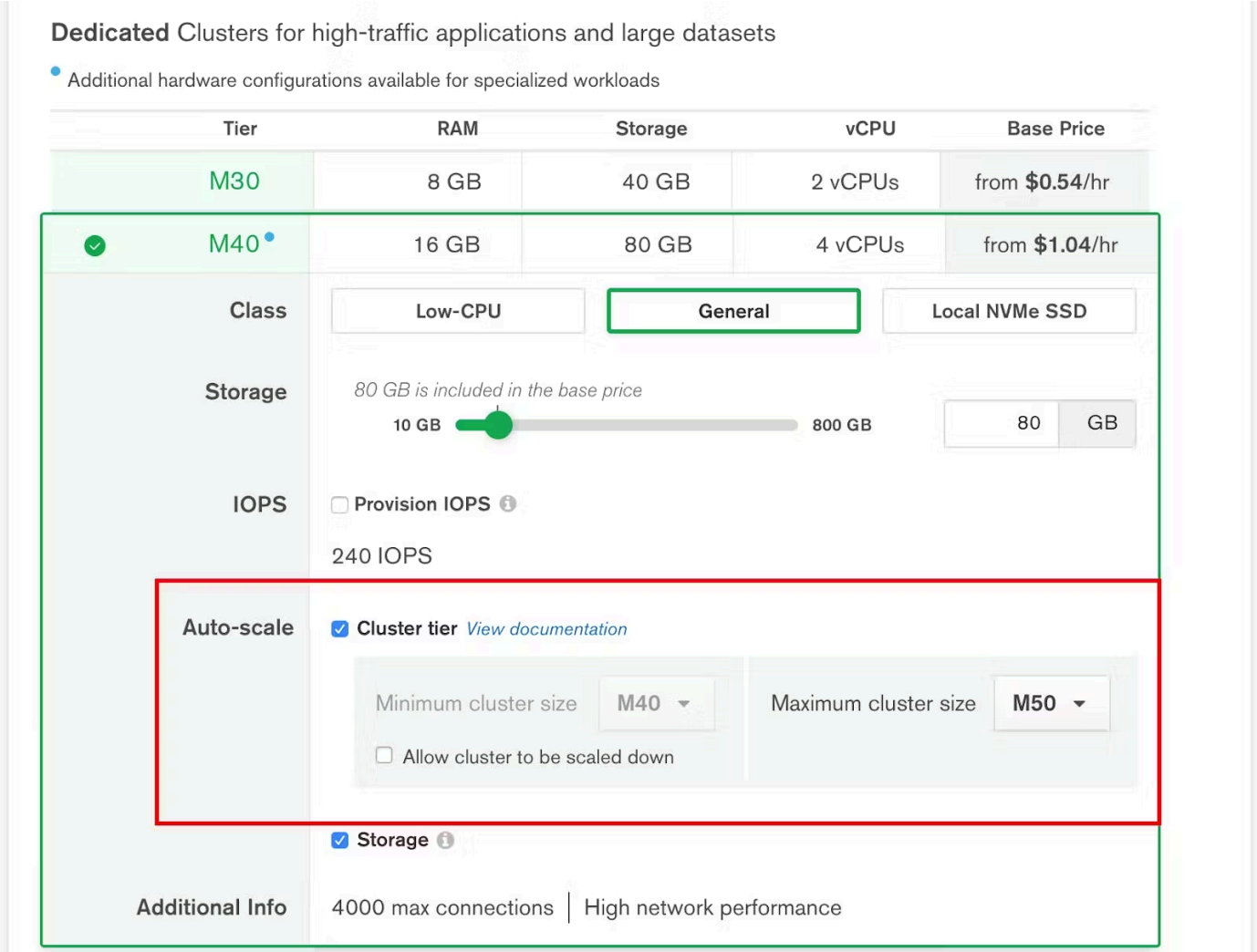
MongoDB Atlas also has the option to create global clusters, which geographically partition (shard) your database to keep relevant data close to end users for regulatory compliance and low-latency reads and writes.



While setting up a global cluster in MongoDB Atlas, you can select from several pre-configured options or customize the configuration of zones to meet the needs of your application and customers.

MongoDB auto-scaling

MongoDB Atlas has cluster auto-scaling, which scales vertically based on cluster usage. This is as simple as configuring the cluster tier:



Both cluster tier/CPU power and storage amount can be auto-scaled. This gives you automated and reactive vertical scaling both up and down, without having to worry about setting up new servers, transferring data, or even downtime in between. If necessary, the cluster can also be paused, effectively scaling the whole cluster to 0 except for storage.

Serverless architecture

Serverless is a next-gen cloud-native development model that allows developers to build applications and run code without thinking about server provisioning, management, and scaling. The term “serverless” does not imply the absence of servers but instead reflects the fact that server provisioning and management have been abstracted (or hidden) from the end-user.

MongoDB Atlas includes support for serverless deployments. When considering applications with variable resource needs, both auto-scaling and serverless architecture can optimize performance while minimizing cost.

Summary

In this article, we reviewed different types of database scaling as well as how to implement each of these in MongoDB Atlas. You can try these different scaling options in your free MongoDB Atlas account.

FAQs

Is MongoDB scalable?

Yes. MongoDB allows you to scale your clusters vertically by adding more virtual resources to the cluster, or horizontally by partitioning the data via sharding.

How do you scale a database?

Databases are scaled either vertically (by adding more virtual resources to existing machines) or horizontally (by adding more machines, distributing data, and processing across those machines).

What is scaling in DBMS?

Scaling in DBMS is the ability to expand the capacity of a database system in order to support larger amounts of requests and/or store more data without sacrificing performance.

What are scalability issues?

Scalability issues are problems caused by a system's inability to support growing demand on virtual resources, such as storage/memory, processing, and network bandwidth. These are usually manifested as a system's degraded performance, errors, or unresponsiveness.

How do you test scalability?

Scalability is usually tested through load testing, e.g., simulating a real-life large number of requests and ensuring that the system can support those requests and adapt to differing amounts of load.

Are relational databases scalable?

Yes. However, relational databases are not as easily scaled as more modern, non-relational databases. MongoDB, for example, is built from the ground up to scale massively and has high availability.

Why are NoSQL databases more scalable than RDBMS databases?

NoSQL databases are usually built by design for a distributed database environment, allowing them to take advantage of more availability and partition networking built-in solutions, which sometimes comes as a tradeoff for consistency.

Most relational database management systems (RDBMS), such as SQL Server and Oracle, choose consistency over availability. These systems often focus on storing business transaction information, and so consistency is critical to their operation.

On the other hand, most non-relational and NoSQL databases choose availability over consistency because their main focus is the ability to support large amounts of users and data volume even when some of the database nodes go down. This assists in the support of scalability using the "partition of data" approach. MongoDB has both availability and consistency thanks to replica sets and multi-document transactions. This helps make it an ACID-compliant database and reduces the tradeoffs developers normally have to make.

How do I scale a SQL database?

Generally, you start by scaling vertically by adding more storage, CPUs, and memory. You could also enable replication and serve some of the read requests from different nodes in the cluster. However, this may require that the application be aware of the different nodes.

How do I scale a database horizontally?

Horizontal scaling involves adding additional servers and partitioning the system dataset and load over those servers. Vertical scaling involves expanding the resources used by a single server/replica set.

Which is better: horizontal or vertical scaling?

Neither is better. You should choose the type of database scaling that meets your use case. Vertical scaling is generally simpler but more limited. Horizontal scaling is more complex but supports larger loads in terms of the number of requests served as well as total data storage.

Related Resources

- AWS re:Invent 2022 Presentation - From RDBMS to NoSQL
- Deploy MongoDB Atlas Sharded Cluster
- Sharding Performance Best Practices
- Atlas Global Clusters
- Sharding Reference
- Beginners Guide: MongoDB Basics
- MongoDB Atlas Tutorial
- How does replication work in MongoDB?
- Using ACID transactions in MongoDB