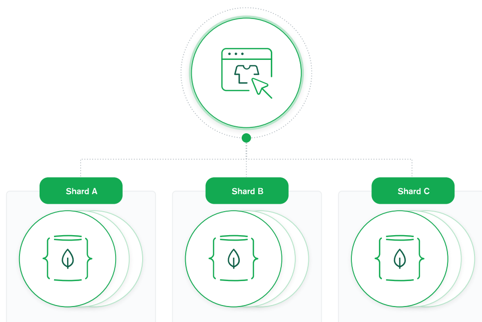
Sharding in MongoDB

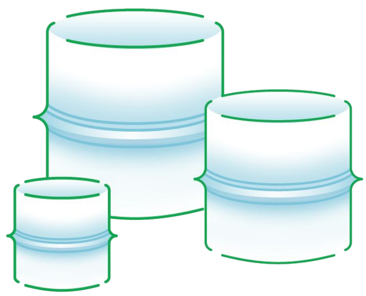
Google slide deck available here

MongoDB Sharding Recap



Data partitioning uses a replica set per partition or shard.

Let's take a moment to recap what we've previously covered on Sharding in the Architecture lesson.
Sharding is also known as data partitioning and it uses a replica set per partition or shard.
Here's a typical sharded cluster with Shard A (pause and click), Shard B (pause and click), and Shard C for a three shard or three partition deployment.



Why you should shard?

Increase volume of persisted data

Increase/distribute throughput to scale both reads & writes

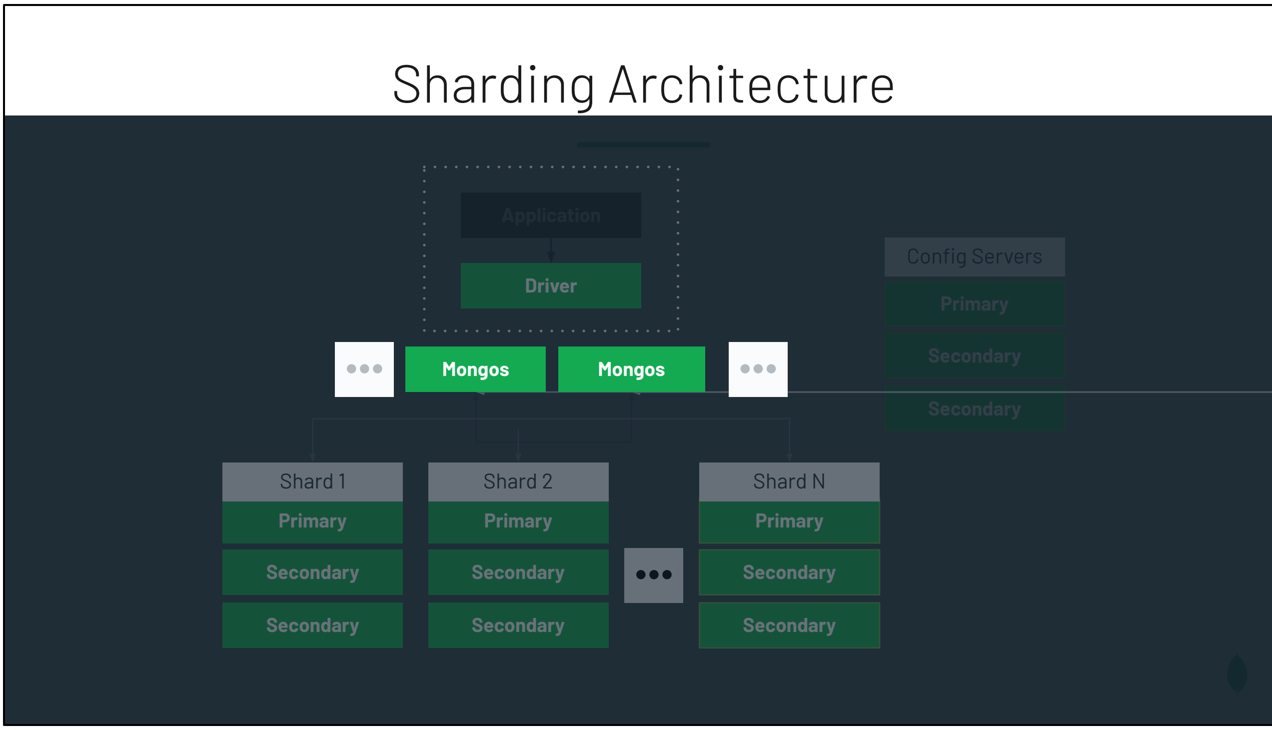
Decrease latency of both reads and writes

Let’s just clarify again the reasons why you should or would want to shard.

Firstly, it allows for your deployment to deal with a greater volume of persistent data. This is achieved by adding more storage associated to the machines/instances you add as you shard.

Secondly, sharding allows for an increase in the throughput for both reads and also for writes. The additional machines/instances added help improve throughput because they add CPU cores, Storage IOPS, and add additional network interfaces to service requests.

Finally, sharding helps decrease the latency of reads and of writes. The addition of RAM (as you increase machines) helps to increase the size of the working set and as such helps decrease these latencies.



We'd previously covered the sharding architecture but it is useful to recap it here to ensure we are clear on the components and their roles before diving deeper into sharding.

The MongoDB Sharding architecture has three primary aspects, the metadata which is held by the config servers.

The data itself which is held by the shard where it resides.

The routing of requests from the application to where the relevant data lives is handled via the mongos routing layer.

What does a shard key do?

The shard key **determines how documents are distributed** amongst a cluster's shards.

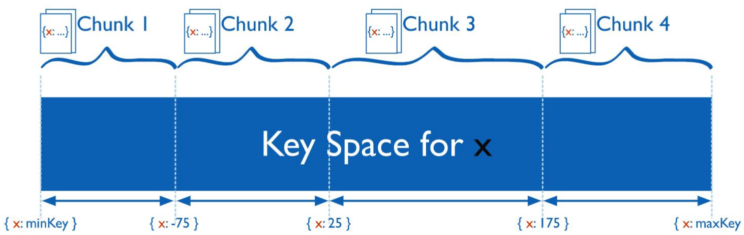
We've discussed partitioning when we are sharding the documents, let's dive a little deeper. The shard key determines how documents are distributed amongst a cluster's shards.

What is a shard key made up of?

The shard key is either an **indexed field** or **indexed compound fields**. It defines a space of values like points on a line. **Key ranges** are segments of that line.

The shard key is either an indexed field or indexed compound fields.
It defines a space of values like points on a line.
Key ranges are segments of that line.

What is shard key range?



Chunks, **contiguous range of shard key values**

Each **shard key range is associated to a chunk.**

Each doc is assigned to a chunk per it's shard key

The database strives to balances these chunks across the shards.

Let's answer what a shard key range is and use the diagram on the slide to help clarify.

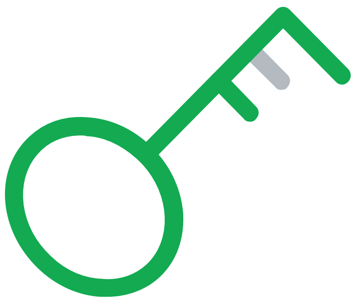
Chunks are a contiguous range of shard key values.

Each shard key range is associated to a chunk, we can see the four ranges and four chunks as the key space for X.

Each doc is assigned to a chunk per the value of it's shard key.

The MongoDB database strives to balances these chunks across the shards to ensure an even distribution.

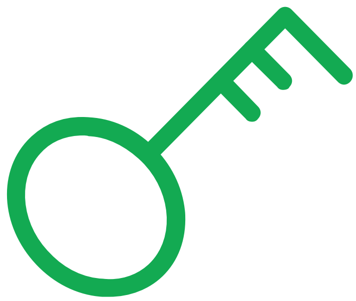
How is a shard key used?



The **shard key is used to route operations to the appropriate shard**. The shard key is used for reads and also for writes.

The shard key is used to route operations to the appropriate shard.
The shard key is used for reads and also for writes.

What is the perfect shard key?



All **queries, inserts, updates, and deletes would each be distributed uniformly across all of the shards.** Operations would only ever go to the relevant shard with the data.

The perfect shard key is one where queries, inserts, updates and deletes are uniformly distributed across all the shards.

It would further have operations only go to the shard with the data and never query any other shard(s).

Sadly, this is an idealised representation and shard key design involves many trade-offs and considerations, which we'll look at next.

Considerations for a good shard key

Cardinality

How well the shard key can be broken into smaller groups

Lower cardinality shard keys will put many documents into a single chunk

See: https://www.mongodb.com/blog/post/on-selecting-a-shard-key-for-mongodb

The first consideration is that of cardinality, specifically a shard key that can be broken up into many smaller groups to give it a high cardinality. A low cardinality shard key means MongoDB will attempt to put many documents into a single chunk.

The link at the bottom of the page links to a post which discusses these criteria in more depth
https://www.mongodb.com/blog/post/on-selecting-a-shard-key-for-mongodb

Considerations for a good shard key

Write Distribution

Design the shard key to evenly distribute writes

Evenly distributing writes should be across all the available shard to optimise the write scalability of the shard key

See: https://www.mongodb.com/blog/post/on-selecting-a-shard-key-for-mongodb

The next consideration is that of write distribution, specifically you want to design the shard key so that writes are evenly distributed across all of the available shards to optimise your write scalability.

Considerations for a good shard key

Read Distribution

Design the shard key to evenly distribute reads

Evenly distributing reads should be across all the available shard to optimise the read scalability of the shard key

See: https://www.mongodb.com/blog/post/on-selecting-a-shard-key-for-mongodb

The third consideration is the distribution of your reads, in a similar fashion to your writes you would also like that your read are evenly spread across all of the available shards in your cluster to optimise the scalability of reads in your cluster.

Considerations for a good shard key

Read Targeting

Design you read to use a shard key that can target an individual shard

The shard key should avoid scatter/gather where all shards need to be queried in order to return a result

See: https://www.mongodb.com/blog/post/on-selecting-a-shard-key-for-mongodb

The fourth aspect to consider is that of read targeting, ideally your read requests should use a shard key that support targeting an individual shard and should not be scatter/gather where all the shards must be queried to return a result.

Considerations for a good shard key

Read Locality

The locality of a read applies to queries involving ranges

Specifically you want to balance read targeting against read locality in certain cases

Social network application - influencers, user name and time of each content item to help locality

See: https://www.mongodb.com/blog/post/on-selecting-a-shard-key-for-mongodb

The fifth aspect to consider when creating a good shard key is read locality. This criteria applies only to the range query strategy, which we'll cover on the next slide. In the case of read locality you may need to balance read targeting against it for specific situations.

Take a social network application, there are influencers who have either huge numbers of followers or significant amounts of content they have created. In such a situation, you will likely need to use a compound shard key that keeps the data you are reading on a single shard. This means for a social media influencer who has generated a large amount of content you would likely have their name or user identified and the time when each content item was posted as the compound shard key to help localise the read to a single shard.

Considerations for a good shard key

Index Locality

How indexes for large data sets are loaded into RAM has an impact

Taking the previous read locality influencer example, using a user name and content time spreads the index entries rather than locates them closely

A related consideration to read locality is index locality. This relates to how RAM needs to be considered with large data sets in terms of how it is indexed and how the index needs to be designed to ensure only portions rather than the entire of the index space needs to be loaded into RAM.

The example for read locality also applied here where the user name and time the content was created would make for a better index as well as locating the document on the same shard versus simply using the user name as the index. In the latter, whilst the user name and documents associated to it will go to a single shard, the index will be spread rather than located closely together in terms of how the index is stored.

Considerations for a good shard key

General considerations

The shard key should be used in the majority of your queries

A shard key should not have more than 64MB sharing the key to avoid jumbo chunks where too many documents have the same key

A shard key should co-locate data you query together (similar to read locality)

Ideally, you should consider a shard key that is in the majority of your queries.

A shard key should not have more than 64MB sharing the key to avoid jumbo chunks where too many documents have the same key

A shard key should co-locate data you query together (similar to read locality)

Quiz

Quiz

Which of the following are criteria you need to consider when picking a good shard key? More than 1 answer choice can be correct.

- A. Write Distribution
- B. Typical number of documents returned by a query
- C. Read Distribution
- D. Read Targeting
- E. Document size

Quiz

Which of the following are criteria you need to consider when picking a good shard key?

- ✓ A. Write Distribution
- ✗ B. Typical number of documents returned by a query
- ✓ C. Read Distribution
- ✓ D. Read Targeting
- ✗ E. Document size

CORRECT: Write Distribution - Being able to spread the load of writes across all the shards in a cluster to avoid bottlenecks is another criteria for a good shard key.
INCORRECT: Typical number of documents returned by a query - This is not a good shard key criteria, it is related to designing good queries which is another aspect to consider within the design of your application.
CORRECT: Read Distribution - Being able to spread the load of reads across all the shards in a cluster to avoid bottlenecks is another criteria for a good shard key.
CORRECT: Read Targeting - Being able to read from one shard with a targeted query rather than needing to query all the shards (broadcast / scatter-gather) is an important criteria for a good shard key.
INCORRECT: Document Size - The size of the document is not related to criteria for a good shard key.

Quiz

Which of the following are criteria you need to consider when picking a good shard key?

- A. **Write Distribution**
- B. Typical number of documents returned by a query
- C. Read Distribution
- D. Read Targeting
- E. Document size

This is correct. Being able to spread the load of writes across all the shards in a cluster to avoid bottlenecks is another criteria for a good shard key.

CORRECT: Write Distribution - This is correct. Being able to spread the load of writes across all the shards in a cluster to avoid bottlenecks is another criteria for a good shard key.

Quiz

Which of the following are criteria you need to consider when picking a good shard key?

- A. Write Distribution
- B. Typical number of documents returned by a query
- C. Read Distribution
- D. Read Targeting
- E. Document size

This incorrect. This is not a good shard key criteria, it is related to designing good queries which is another aspect to consider within the design of your application.

INCORRECT: Typical number of documents returned by a query - This incorrect. This is not a good shard key criteria, it is related to designing good queries which is another aspect to consider within the design of your application.

Quiz

Which of the following are criteria you need to consider when picking a good shard key?

- ✓ A. Write Distribution
- ✗ B. Typical number of documents returned by a query
- ✓ C. **Read Distribution**
- ✓ D. Read Targeting
- ✗ E. Document size

This is correct. Being able to spread the load of reads across all the shards in a cluster to avoid bottlenecks is another criteria for a good shard key.

CORRECT: Read Distribution - This is correct. Being able to spread the load of reads across all the shards in a cluster to avoid bottlenecks is another criteria for a good shard key.

Quiz

Which of the following are criteria you need to consider when picking a good shard key?

- ✓ A. Write Distribution
- ✗ B. Typical number of documents returned by a query
- ✓ C. Read Distribution
- ✓ **D. Read Targeting**
- ✗ E. Document size

This is correct. Being able to read from one shard with a targeted query rather than needing to query all the shards (broadcast / scatter-gather) is an important criteria for a good shard key.

CORRECT: Read Targeting - This is correct. Being able to read from one shard with a targeted query rather than needing to query all the shards (broadcast / scatter-gather) is an important criteria for a good shard key.

Quiz

Which of the following are criteria you need to consider when picking a good shard key?

- ✓ A. Write Distribution
- ✗ B. Typical number of documents returned by a query
- ✓ C. Read Distribution
- ✓ D. Read Targeting
- ✗ E. **Document size**

This incorrect. The size of the document is not related to criteria for a good shard key.

INCORRECT: Document Size - This is incorrect. The size of the document is not related to criteria for a good shard key.

Exercise

Exercise: Pick a shard key for viewing inboxes

Design a shard key for a web based email service and select which one of the following keys would **be the most suitable key for viewing inboxes** and discuss:

a. `{ from: 1 }`
b. `{ from: 1, to: 1 }`
c. `{ to: 1, sent_on: 1 }`
d. `{ from: 1, message: 1 }`

```
{
 from: "Joe",
 to: ["Bob", "Jane"],
 sent_on: new Date(),
 message: "Hi!"
}
```

(NOTE) In this exercise, take a few minutes and sketch out the various aspects given the example document and in small groups or individually discuss the metrics of the four options presented on the slide.

Design a shard key for a web based email service and select which one of the following keys would be the most suitable key for viewing inboxes and discuss:

A. { from: 1 }
B. { from: 1, to: 1 }
C. { to: 1, sent_on: 1 }
D. { from: 1, message: 1 }

Finally, here is an example document of what is stored in terms of an email for the email service.

Exercise: Pick a shard key for viewing inboxes

Design a shard key for a web based email service and select which one of the following keys would **be the most suitable key for viewing inboxes** and discuss:

a. `{ from: 1 }`

b. `{ from: 1, to: 1 }`

c. `{ to: 1, sent_on: 1 }`

d. `{ from: 1, message: 1 }`

```
{
 from: "Joe",
 to: ["Bob", "Jane"],
 sent_on: new Date(),
 message: "Hi!"
}
```

The most suitable shard key of the suggestions is “c” when we consider it specifically for viewing inboxes.

The best of this selection of possible shard keys is { to: 1, sent_on: 1} as it provides a nice way to locate the message (to) and order them (sent_on) which facilitates viewing inboxes and messages

Exercise: Pick a shard key for viewing inboxe

Design a shard key for a web based email service and select which one of the following keys would **be the most suitable key for viewing inboxes** and discuss:

a. `{ from: 1 }`
b. `{ from: 1, to: 1 }`
c. `{ to: 1, sent_on: 1 }`
d. `{ from: 1, message: 1 }`

```
{
 from: "Joe",
 to: ["Bob", "Jane"],
 sent_on: new Date(),
 message: "Hi!"
}
```

The other shard keys are possibilities but are not ideally suited for viewing inboxes. In the case of { from: 1 }, the lack of a time field hinders viewing as additional sorting and processing would be required to say return the last 10 emails from a specific sender.

In the case of { from: 1, to: 1 }, it's a similar issue that whilst the key contains the participants in the email there is no time field to help with sorting and presenting emails for viewing in a chronological fashion.

The best of this selection of possible shard keys is { to: 1, sent_on: 1} as it provides a nice way to locate the message (to) and order them (sent_on) which facilitates viewing inboxes and messages.

The final option { from: 1, message: 1} has the same issues as 'a' and 'b'. There is no easy way to sort these messages for viewing the inboxes.

This example is simplified and you would have more fields plus considerations that you would use for a real world system when design a shard key.

Exercise: Pick a shard key for viewing inboxe

Design a shard key for a web based email service and select which one of the following keys would **be the most suitable key for viewing inboxes** and discuss:

a. `{ from: 1 }`

b. `{ from: 1, to: 1 }`

c. `{ to: 1, sent_on: 1 }`

d. `{ from: 1, message: 1 }`

```
{
 from: "Joe",
 to: ["Bob", "Jane"],
 sent_on: new Date(),
 message: "Hi!"
}
```

In the case of { from: 1, to: 1 }, it's a similar issue that whilst the key contains the participants in the email there is no time field to help with sorting and presenting emails for viewing in a chronological fashion.

Exercise: Pick a shard key for viewing inboxe

Design a shard key for a web based email service and select which one of the following keys would **be the most suitable key for viewing inboxes** and discuss:

a. `{ from: 1 }`
b. `{ from: 1, to: 1 }`
c. `{ to: 1, sent_on: 1 }`
d. `{ from: 1, message: 1 }`

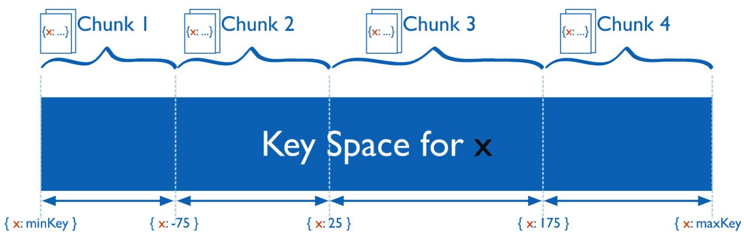
```
{
 from: "Joe",
 to: ["Bob", "Jane"],
 sent_on: new Date(),
 message: "Hi!"
}
```

The final option { from: 1, message: 1} has the same issues as ‘a’ and ‘b’. There is no easy way to sort these messages for viewing the inboxes.

This example is simplified and you would have more fields plus considerations that you would use for a real world system when design a shard key.

Sharding Strategies

Sharding strategies: Range sharding



Contiguous ranges determined by the shard key values.

A **"close" shard key value** should have documents on the same chunk or shard.

Default sharding strategy in MongoDB.

See: https://docs.mongodb.com/manual/core/ranged-sharding/

We've already seen this type of sharding without covering in depth. Here's a few additional aspects to note for range sharding.

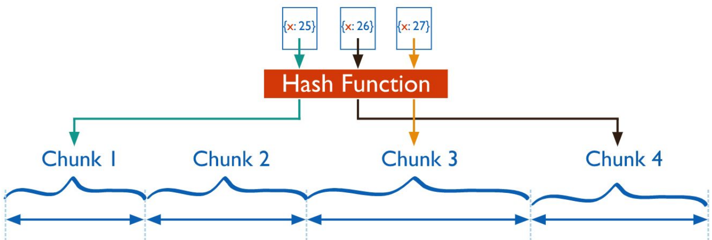
The contiguous ranges are determined by the shard key values.

If a shard key value is 'close' to another document's shard key value then it is likely they will be present on the same chunk or at least within the same shard.

Range based sharding is the default sharding strategy in MongoDB.

The web link has the detailed MongoDB documentation page on range sharding for more information.

Sharding strategies: Range sharding



Single field hashed index or **compound field hashed index.**

Computes the hash value as the shard key value.

Better **data distribution** at the cost of **potentially more broadcast queries.**

See: https://docs.mongodb.com/manual/core/ranged-sharding/

Hashed sharding was designed for shard keys with fields that change monotonically like ObjectId values or timestamps. These shard keys would cluster on a single shard with range based sharding and create a ‘hot’ shard. Hashed sharding attempts to avoid that situation for these types of shard keys.

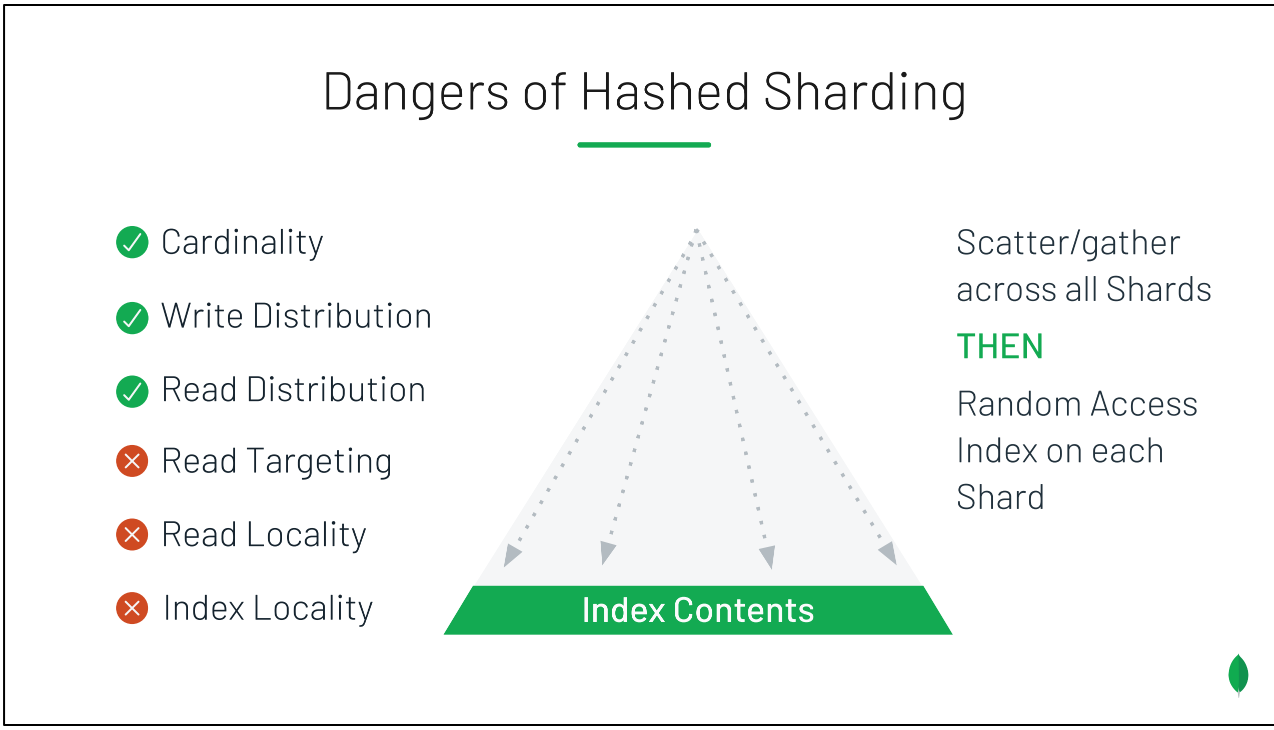
Hashed sharding can use single field hashed index or a compound field hashed index. The compound field hashed index computes the hash value of a single field in the compound index; this value is used along with the other fields in the index as your shard key.

The process of hashed sharding uses a hash function the value of this for the field is used as the shard key value or as part of the shard key if a compound field hashed index.

The idea behind hashed sharding was to provide better data distribution. There is a tradeoff in this case as it is likely to increase the number of broadcast/scatter-gather queries.

Ideally for hashed sharding you should use a low cardinality hash (~720 values) of the field combined with an incrementing value. This provides a good spread of values over the shards with a fast incrementing shard key.

The web link has the detailed MongoDB documentation page on hashed sharding for more information.



Hashed sharding can be a useful technique but there are some caveats and dangers when using this sharding strategy that you should consider when designing your shard key.

Specifically, it can provide good cardinality, write and read distribution with a good shard key.
However, it does not provide good read targeting as the hashed nature of the shard key and the data distribution means that a broadcast/scatter-gather query will be required to find and return the document(s) for a query.

In a similar fashion, it has poor read locality as related documents are unlikely to be located on the same shard.

Finally and significantly it can have poor index locality, this means that to service the query the entire index must be loaded in memory as the random access nature of the shard key means it is not possible to know or predict which portion of the index file will be needed. This means that the entire index needs to be loaded in memory to effective service queries using as hashed sharded key.

Reasons when you should shard

You have a **reasonable and growing data size** ~200GB to 0.5TB

A **resource is maxed out** (and you know why)

Vertical scaling is not possible or cost effective

Your schema and your code have **already been optimized**

Sharding is not something that is needed for all applications and thankfully there are some clear reasons or flags as to when you should consider sharding.

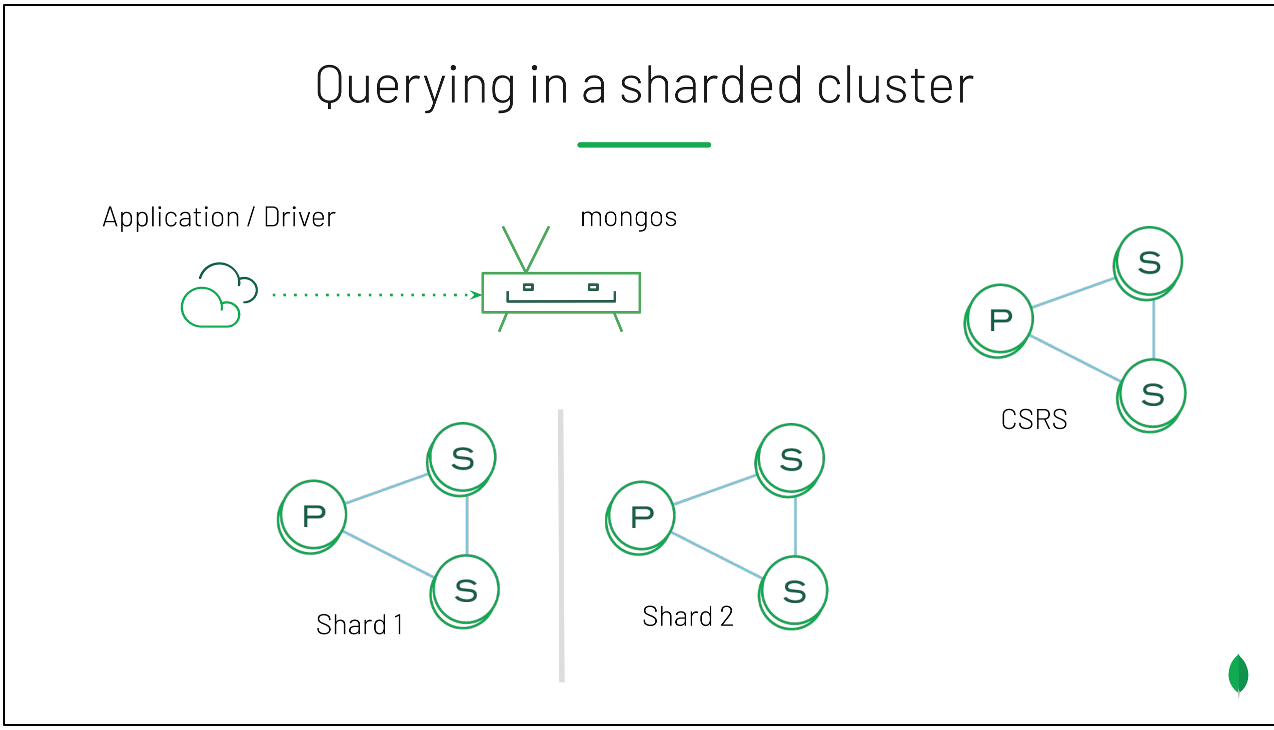
If you already have a reasonable size of data which is growing and you predict it will continue to grow. After your data is larger than 200GB and smaller than 0.5TB is probably a good point to consider sharding but it is application and scenario specific.

If you have maxed out your hardware and it is not possible or cost effective to buy a bigger machine then it’s time to consider sharding.

If you have optimized your schema and your code then it is also time to consider sharding.

Typically, you will have several of these reasons occurring as the ideal time to undertake sharding your data.

Querying in a Sharded Cluster



When querying from a sharded cluster, the first stage is the application / driver makes the request to the mongos routing process.

Let's look at how the mongos handles a read request from the application.

Querying in a sharded cluster for reads

If query contains shard key (or at least first field of key)

Lookup where data with that key is stored

Send query to those replica sets to run as normal

When results come back stream to client.

Else

Send query to all servers.

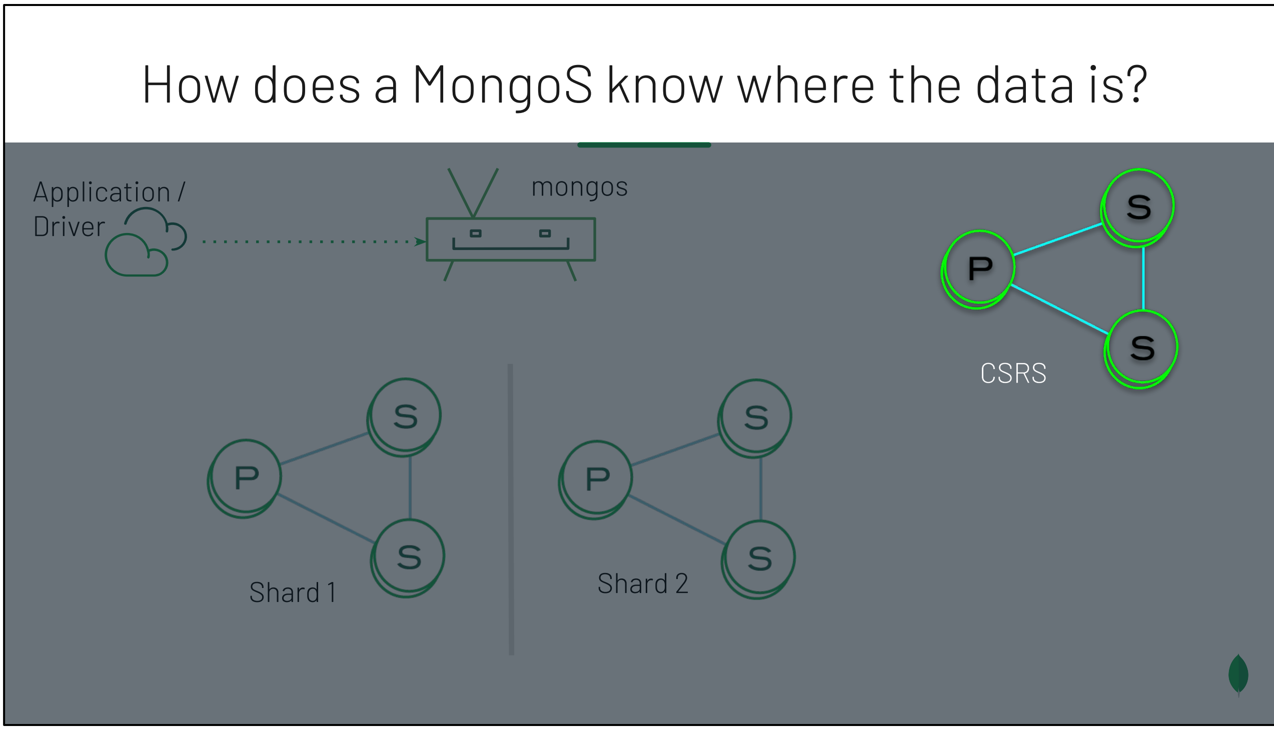
When results come back stream to client.

Let's look at how a read happens in a sharded cluster from the viewpoint of the mongos.

If the query has the shard key or at least the first field of key then this will be used. A lookup with this from the mongos cache of the routing table will happen to determine which shard holds the chunk with the range containing the key. The query is then passed to the replica set (shard) as normal. The results are then sent back to the mongos which streams them to the client.

If the query doesn't contain any details that can be used to route then the query must be sent to all servers. Once the results are back to the mongos it again also streams them back to the client.

Metadata and routing



Looking again at our cluster, it’s easy to see we use shards to host our data but how does a mongos route a query from an application to the right shard?

Let’s look at how the data about data or metadata is stored and how this helps the mongos route the query correctly.

Let’s first look at the Config Servers which we haven’t spoken about before, it is here that the metadata is stored for the cluster.

Config Server Replica Set (CSRS)

Routing table: list of chunks on every Shard + the ranges that define the chunks ('config' DB).

Metadata: Authentication metadata (RBAC rules + internal authentication settings for the entire Sharded cluster ('admin' DB).

Shards read chunk metadata from the config servers using **Read Concern of Majority.**

MongoS caches the metadata from the CSRS & uses it to route.

When the metadata changes the **MongoS** cache gets updated.

The config servers or the config server replica set (CSRS) holds the routing table for the data, which lists all the chunks on each shard and the ranges that define these chunks. It also holds the authentication metadata to control access for the sharded cluster.

When metadata is read by the shards, it is done so using the read concern of majority.

The mongos router(s) take and cache a copy of the metadata from the config servers. The mongos uses this copy to route the requests from application to the appropriate shards. If and when the metadata changes then the mongos cache gets updated.

What are chunks exactly?

A **chunk** is the term used to refer to **all documents** where **the shard key is in a given range of values**.

Each chunk exists on **a single shard**.

If a shard key falls into the range then it is set to be in the chunk.

Any **read or write requests** are routed to the shard hosting that chunk.

A chunk is the term used to refer all the documents that are within a given range of values for a shard key.

Each chunk exists on a single shard.

If a shard key falls within the range for the chunk then it is said to be within that chunk.

Any read or write request will be routed to the shard hosting the chunk.

Balancing and migrations of chunks

Balancing occurs on the **primary config server.**

A **regular check for an imbalance** in terms of the number of chunks between shards.

If an **imbalance is detected a chunk migration occurs** with chunks are moved between the shards to balance the distribution.

This chunk move process is called a **chunk migration**.

Balancing is an automatic process that occurs regularly in a sharded cluster. It occurs on the primary config server. This process checks for an imbalance in the number of chunks between shards.

If an imbalance is detected then a chunk migration is triggered. This migration process seeks to balance the distribution of chunks between shards.

Chunk splitting

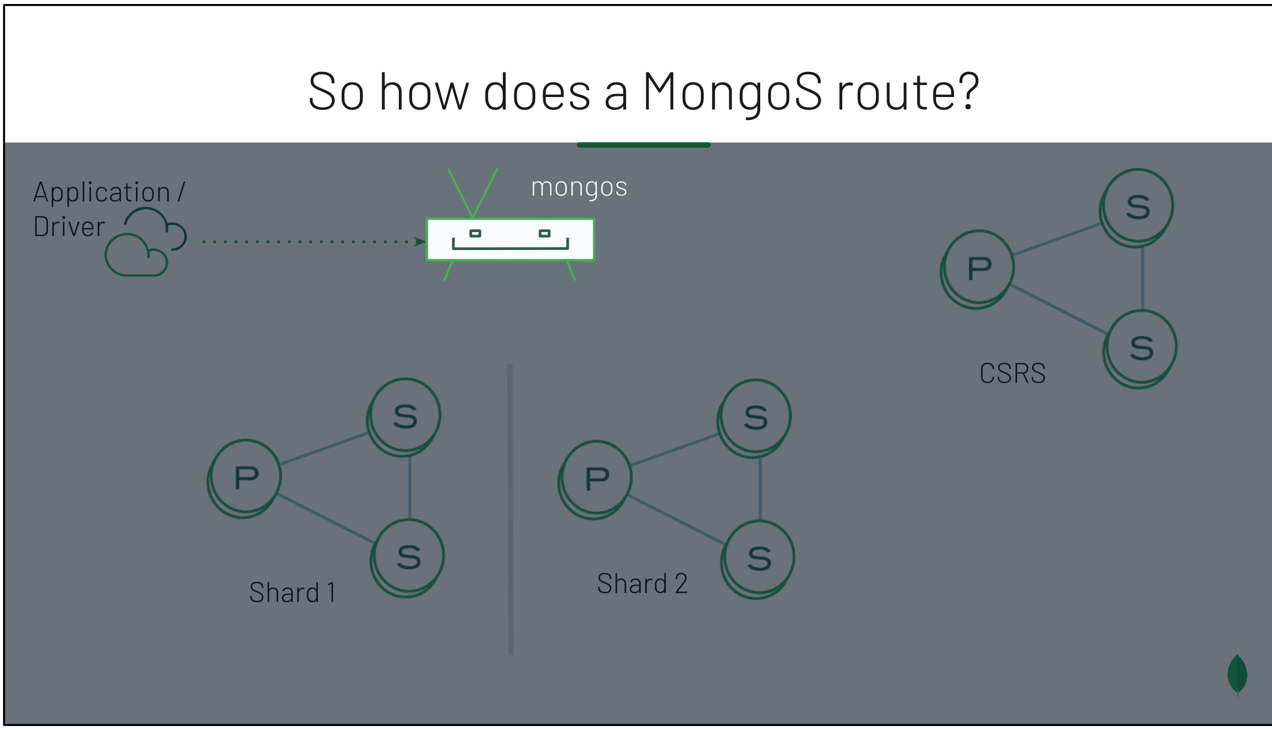
Splits occur on the primary of the shard with the chunk. It is the process that **stops chunks growing too large**.

It happens when:

- a chunk grows beyond a specified chunk size, or
- when the maximum number of documents per chunk setting is exceeded.

Chunk splitting is an automatic process within a sharded cluster. It occurs on the primary of a shard. It is designed to stop chunks from growing too large.

It is triggered when a chunk grows beyond a specified chunk size or when the maximum number of documents per chunk setting is exceeded.



We’ve covered how the metadata is managed and maintained by MongoDB.

Let’s look now at the MongoS in more depth to see how it routes and operates.

MongoS

Routes reads and writes to the relevant shard(s)

Maintains **connection pools** to all members (including the config servers)

Each request it sends has it's **version of the cache included**. If this is determined to be outdated by the shard receiving the request, an error is returned and the mongos refreshes its cache.

MongoDB driver will pick a random **mongos but from those with the lowest latency** from the seed-list of mongos processes automatically.

The mongos routes reads and writes to the relevant shard(s). It does this by establishing a cursor on all targeted shards, then merging all the results. If there sorting required then the primary of the shard(s) sorts not the mongos which only gets th end results.

The MongoS also maintains connection pools to all members (including the config servers).

Each request it sends has it's version of the cache included. If this is determined to be outdated by the shard receiving the request, an error is returned and the mongos refreshes its cache.

MongoDB driver will pick a random mongos but from those with the lowest latency from the seed-list of mongos processes automatically.

Quiz

Quiz

Fill in the blank for each of these questions from what we have just learnt.

- A. A chunk refers to the _________ where the shard key is in a given range of values
- B. Balancing occurs on the _________
- C. Chunk splitting occurs on the _________
- D. The purpose of a mongos is to _________ read and write operations

The first is:

A. A chunk refers to the _________ where the shard key is in a given range of values

Quiz

- A. A chunk refers to the **documents** where the shard key is in a given range of values
- B. Balancing occurs on the __________
- C. Chunk splitting occurs on the __________
- D. The purpose of a mongos is to __________ read and write operations

A chunk refers to the documents where the shard key is in a given range of values.

Quiz

- A. A chunk refers to the **documents** where the shard key is in a given range of values
- B. Balancing occurs on the _________
- C. Chunk splitting occurs on the _________
- D. The purpose of a mongos is to _________ read and write operations

The second is:

A. Balancing occurs on the _________

Quiz

- A. A chunk refers to the **documents** where the shard key is in a given range of values
- B. Balancing occurs on the **primary config server**
- C. Chunk splitting occurs on the _________
- D. The purpose of a mongos is to _________ read and write operations

Balancing occurs on the primary config server, this can be confused with the next question on chunk splitting.

Quiz

- A. A chunk refers to the **documents** where the shard key is in a given range of values
- B. Balancing occurs on the **primary config server**
- C. Chunk splitting occurs on the _________
- D. The purpose of a mongos is to _________ read and write operations

The third is:

A. Chunk splitting occurs on the _________

Quiz

- A. A chunk refers to the **documents** where the shard key is in a given range of values
- B. Balancing occurs on the **primary config server**
- C. Chunk splitting occurs on the **primary of the shard with the chunk**
- D. The purpose of a mongos is to __________ read and write operations

Chunk splitting occurs on the primary of the shard with the chunk. As part of this process the primary will update the primary config server with the changes needed for the metadata but this process is managed by the primary of the shard.

Quiz

- A. A chunk refers to the **documents** where the shard key is in a given range of values
- B. Balancing occurs on the **primary config server**
- C. Chunk splitting occurs on the **primary of the shard with the chunk**
- D. The purpose of a mongos is to _________ read and write operations

The fourth is:

A. The purpose of a mongos is to __________ read and write operations

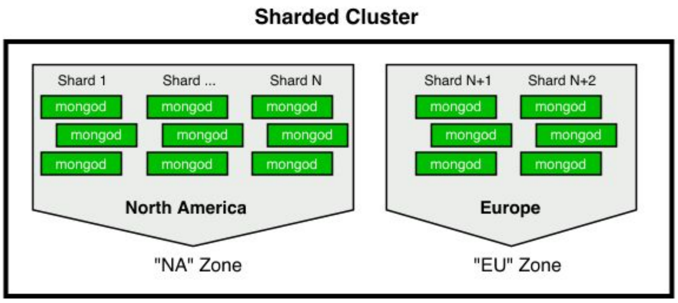
Quiz

- A. A chunk refers to the **documents** where the shard key is in a given range of values
- B. Balancing occurs on the **primary config server**
- C. Chunk splitting occurs on the **primary of the shard with the chunk**
- D. The purpose of a mongos is to **route** read and write operations

The purpose of a mongos is to route read and write operations. It is designed to be lightweight and be only focused on aspects related to this task.

Zoned Sharding

Sharding strategies: Range sharding



Isolate **subsets** of the data to **specific shards**

To support **data sovereignty**

To enable **low-latency reads & writes** from local applications

To allow for **tiered storage** on hardware

See: https://docs.mongodb.com/manual/core/zone-sharding/

Zoned sharding can be used to create zones of sharded data, a zone can be associated with one or more shards. A shard can be associated with any number of zones.

Zoned sharding allow for data to isolated to specific shards, looking at the diagram we can see two zones, one for North America and one for Europe.

These zones can used to support data sovereignty for example ensuring European user's data is retained within the EU (or the EU zone in this example).
Zoning in this fashion can also help applications in the different continents to target their local zone so North American users can target the NA zone to reduce the latency for their reads and their writes.
Another possibility with zoned shared is it allows the ability to use tiered storage, essentially this means HDDs and SSDs together and being able to use the shard key to send older or archival records which your application may not be using to the cheaper HDDs and keep your working set/the current documents on the more expensive SSDs.

Quiz

Quiz

Which of the following scenarios are supported by zone sharding?
More than 1 answer choice can be correct.

- A. Isolating data to a specific geographic region
- B. Directing applications to the nearest data in terms of latency
- C. Using different types of disk storage
- D. Geographical failover and disaster recovery
- E. Optimal shard chunk / data distribution

Quiz

Which of the following scenarios are supported by zone sharding?

- ✓ A. Isolating data to a specific geographic region
- ✓ B. Directing applications to the nearest data in terms of latency
- ✓ C. Using different types of disk storage
- ✗ D. Geographical failover and disaster recovery
- ✗ E. Optimal shard chunk / data distribution

CORRECT: Isolating data to a specific geographic region - This is one of the key use cases for zones in sharding.
CORRECT: Directing applications to the nearest data in terms of latency - The nearest zone can be targeted by the applications to support this.
CORRECT: Using different types of disk storage - This is another of the key use cases for zones in sharding, easily being able to use cheaper but slower disk storage along with expensive but faster disk storage allows for example archival data moved to the cheaper storage tier.
INCORRECT: Geographical failover and disaster recovery - Zone sharding and indeed sharding is not used for disaster recovery or high availability. Those aspects are handled by replication.
INCORRECT: Optimal shard chunk / data distribution - The balancer controls the chunk distribution and manages optimal data distribution within the sharded cluster. This is a separate feature and unrelated to zone sharding.

Quiz

Which of the following scenarios are supported by zone sharding?

- ✓ A. **Isolating data to a specific geographic region**
- ✓ B. Directing applications to the nearest data in terms of latency
- ✓ C. Using different types of disk storage
- ✗ D. Geographical failover and disaster recovery
- ✗ E. Optimal shard chunk / data distribution

This is correct. This is one of the key use cases for zones in sharding.

CORRECT: Isolating data to a specific geographic region - This is correct. This is one of the key use cases for zones in sharding.

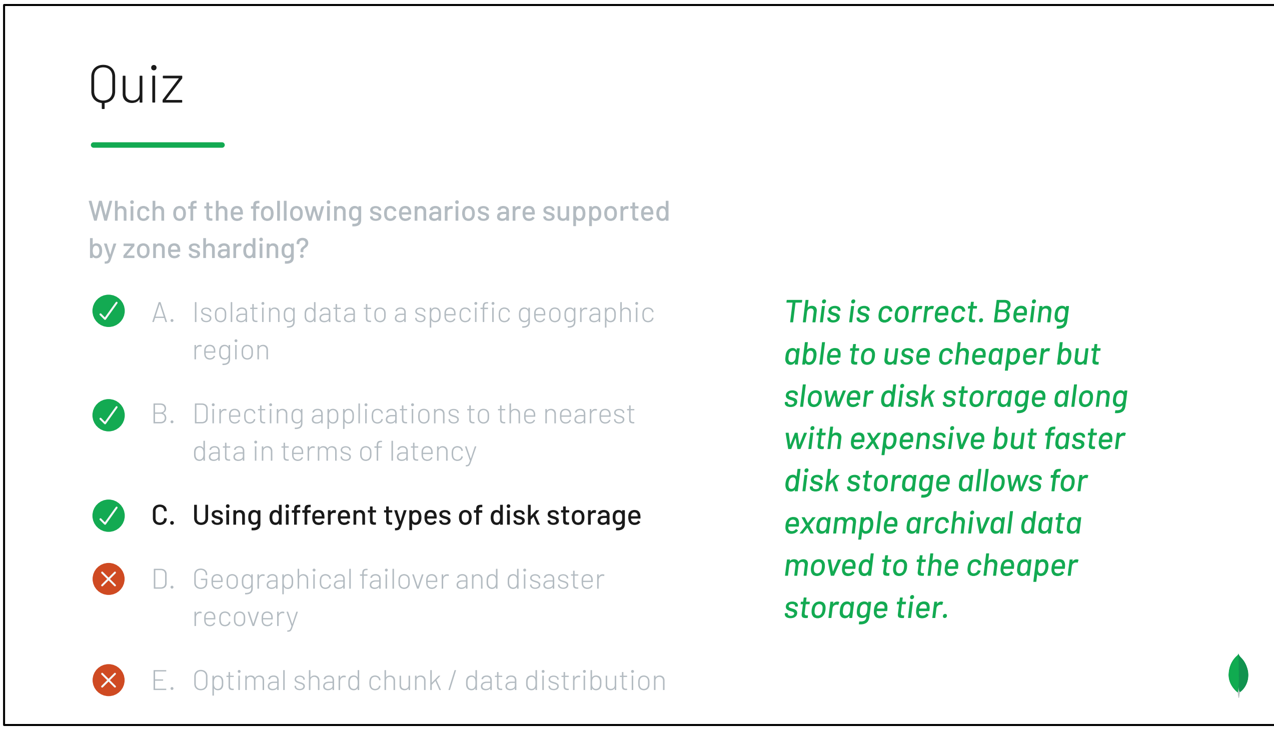
Quiz

Which of the following scenarios are supported by zone sharding?

- A. Isolating data to a specific geographic region
- B. Directing applications to the nearest data in terms of latency
- C. Using different types of disk storage
- D. Geographical failover and disaster recovery
- E. Optimal shard chunk / data distribution

This is correct. The nearest zone can be targeted by the applications to support this.

CORRECT: Directing applications to the nearest data in terms of latency - This is correct. The nearest zone can be targeted by the applications to support this.



CORRECT: Using different types of disk storage - This is correct. Being able to use cheaper but slower disk storage along with expensive but faster disk storage allows for example archival data moved to the cheaper storage tier.

Quiz

Which of the following scenarios are supported by zone sharding?

- ✓ A. Isolating data to a specific geographic region
- ✓ B. Directing applications to the nearest data in terms of latency
- ✓ C. Using different types of disk storage
- ✗ **D. Geographical failover and disaster recovery**
- ✗ E. Optimal shard chunk / data distribution

This incorrect. Zone sharding and indeed sharding is not used for disaster recovery or high availability. Those aspects are handled by replication.

INCORRECT: Geographical failover and disaster recovery - This is incorrect. Zone sharding and indeed sharding is not used for disaster recovery or high availability. Those aspects are handled by replication.

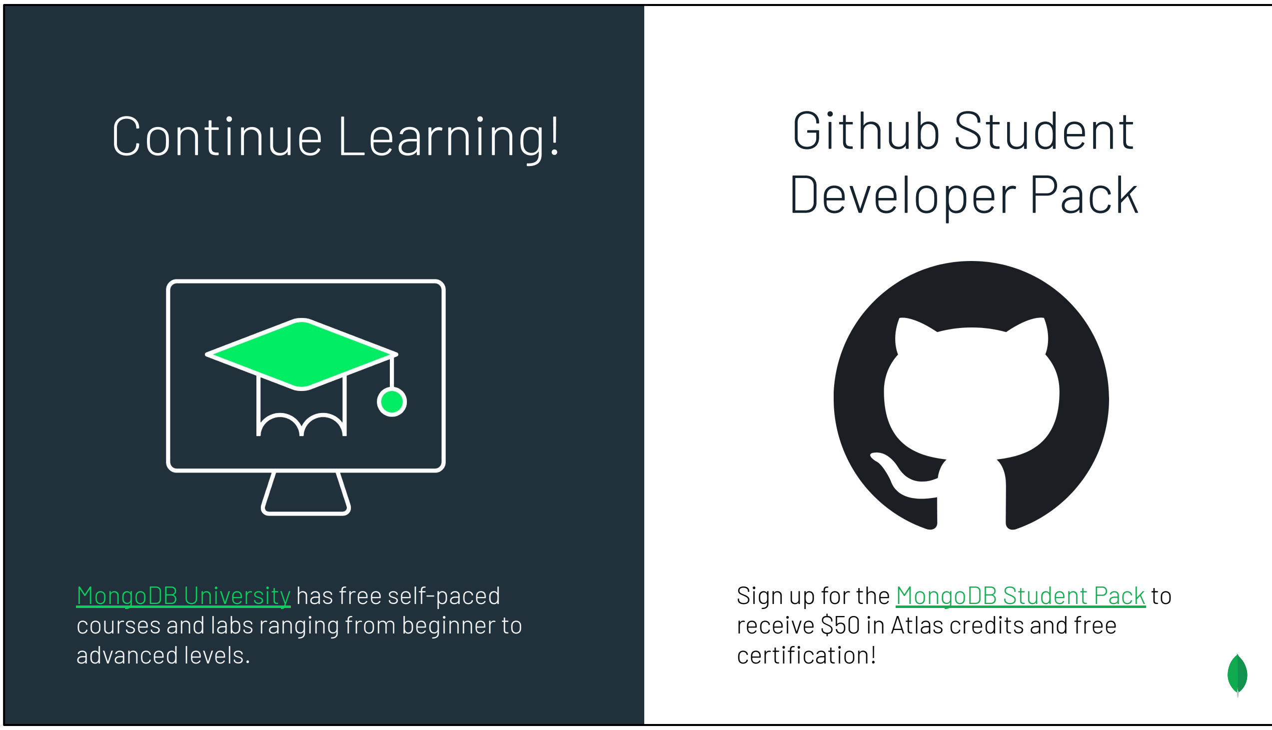
Quiz

Which of the following scenarios are supported by zone sharding?

- ✓ A. Isolating data to a specific geographic region
- ✓ B. Directing applications to the nearest data in terms of latency
- ✓ C. Using different types of disk storage
- ✗ D. Geographical failover and disaster recovery
- ✗ **E. Optimal shard chunk / data distribution**

This incorrect. The balancer controls the chunk distribution and manages optimal data distribution within the sharded cluster. This is a separate feature and unrelated to zone sharding.

INCORRECT: Optimal shard chunk / data distribution - This is incorrect. The balancer controls the chunk distribution and manages optimal data distribution within the sharded cluster. This is a separate feature and unrelated to zone sharding.



This concludes the material for this lesson. However, there are many more ways to learn about MongoDB and non-relational databases, and they are all free! Check out MongoDB's University page to find free courses that go into more depth about everything MongoDB and non-relational. For students and educators alike, MongoDB for Academia is here to offer support in many forms. Check out our educator resources and join the Educator Community. Students can receive $50 in Atlas credits and free certification through the Github Student Developer Pack.