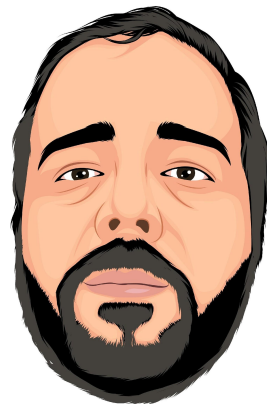




PERCONA
LIVEONLINE
MAY 12 - 13th
2021

MongoDB: To Shard or Not To Shard?

Hello!

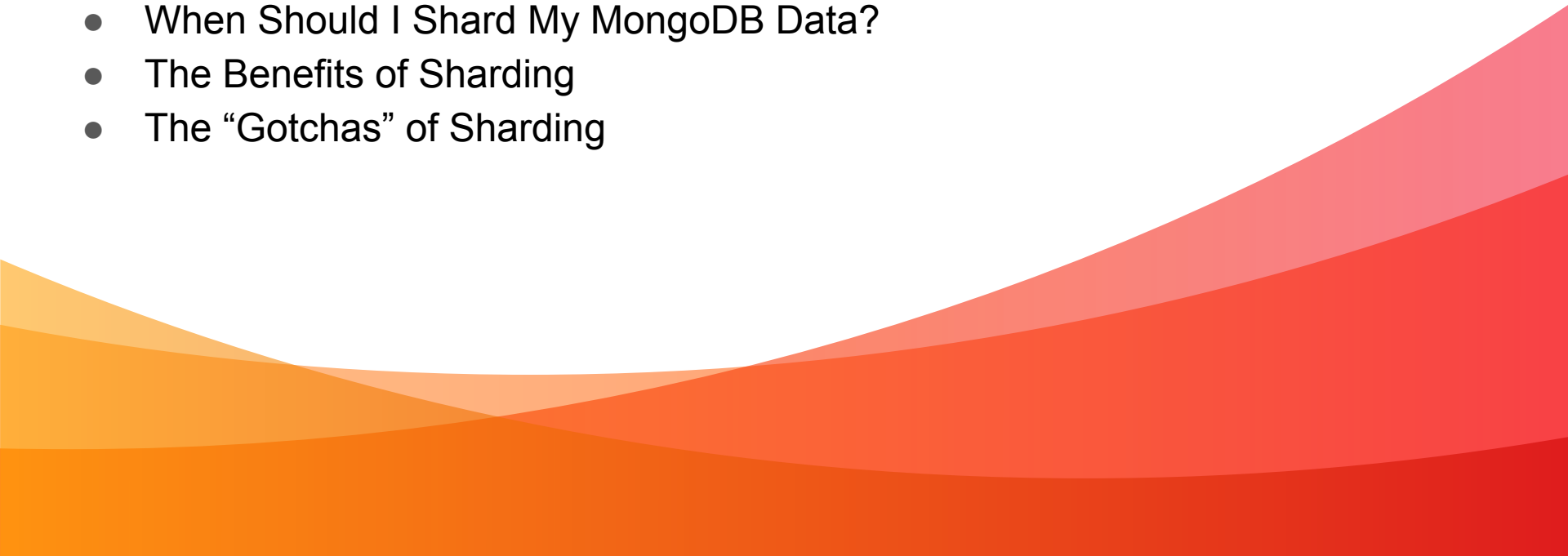


I am Mike Grayson

I am a MongoDB Database Engineer at Percona. I have been working with MongoDB since 2014. I am married with 4 kids and living near Rochester, NY, USA

You can find me at @mikegray831 on Twitter and LinkedIn at <https://www.linkedin.com/in/mikegrayson/>

Agenda

- What is Sharding in MongoDB?
 - What Three Different Types of Sharding in MongoDB?
 - When Should I Shard My MongoDB Data?
 - The Benefits of Sharding
 - The “Gotchas” of Sharding
- 

What is Sharding in MongoDB?

The background of the slide features abstract, wavy shapes in shades of orange and red, creating a modern and dynamic aesthetic.

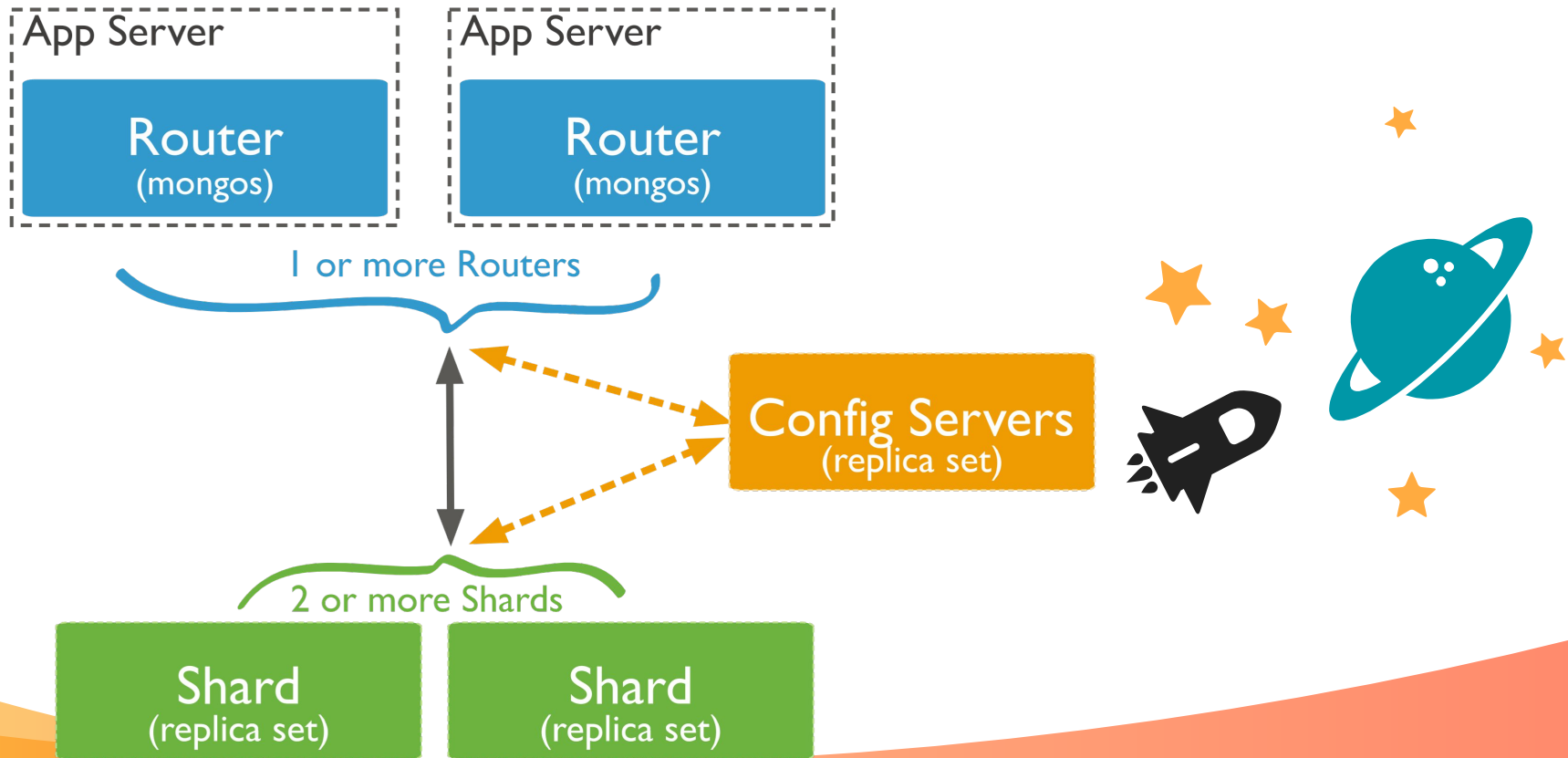


Sharding is a method for distributing data across multiple machines. MongoDB uses sharding to support deployments with very large data sets and high throughput operations.

<https://docs.mongodb.com/manual/sharding/#sharding>

Sharding Architecture Terminology

- Shards – a MongoDB Replica Set that contains a subset of the sharded data.
- Mongos/query router – an interface between your application and the sharded cluster. Allows application to see sharded cluster as if it is one MongoDB instance.
- Config servers – stores metadata and configuration settings for your sharded cluster.



Sharding Concepts

- Shard Key – field or multiple fields in a document that are used to distribute the collections documents across shards.
- Chunks – term for each piece of data partitioned in a sharded collection.
- Shard Balancer – process that runs in the background to migrate chunks across shards to evenly balance data distribution in the cluster.

The Three Different Types of Sharding in MongoDB

The background of the slide features abstract, wavy shapes in shades of orange and red. On the left side, there are overlapping orange shapes. On the right side, there are overlapping red and pink shapes. These shapes create a modern, flowing aesthetic that frames the central text.

MongoDB Sharding Types

Hashed

Uses a hashed index of your shard key.

Ranged

Uses continuous ranges to partition data based on your shard key.

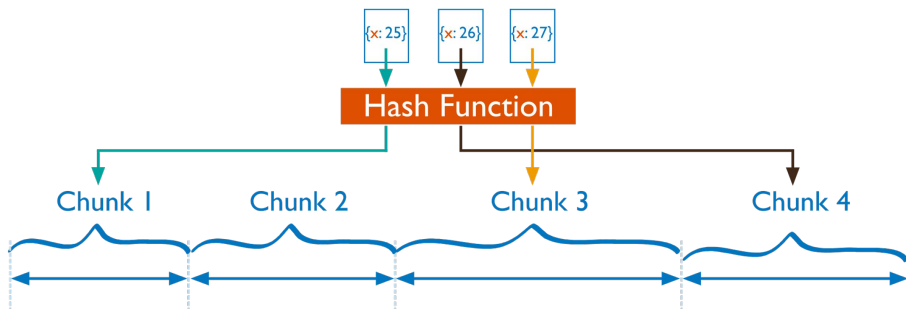
Zoned

Supports location awareness, based on the shard key and zones.

Hashed vs Ranged Sharding

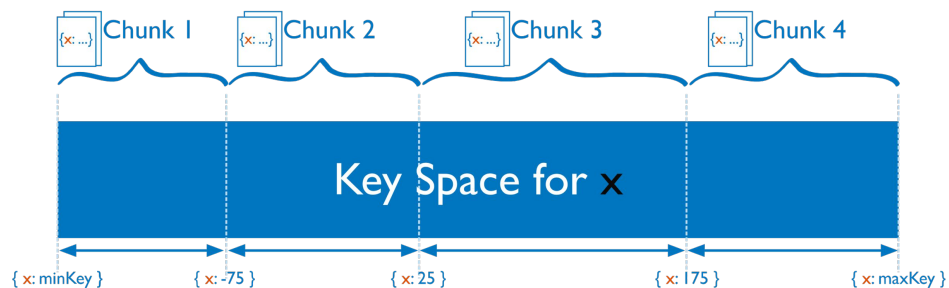
Hash Based

- Reduces Targeted Operations
- Increases likelihood of Broadcast Operations
- More evenly distributes writes



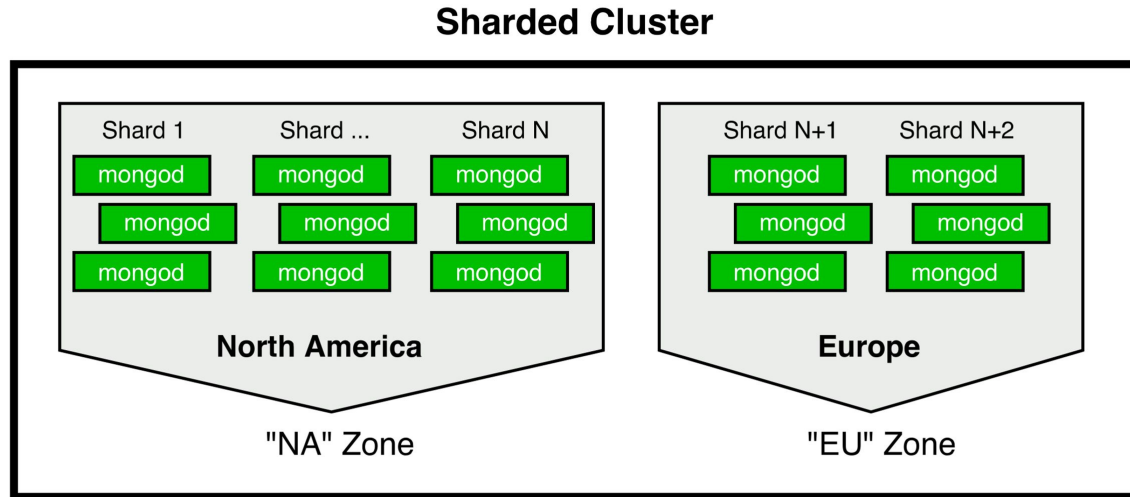
Range Based

- Increases Targeted Operations
- Reduces likelihood of Broadcast Operations
- Writes can be pinned to one shard with a monotonically increasing shard key

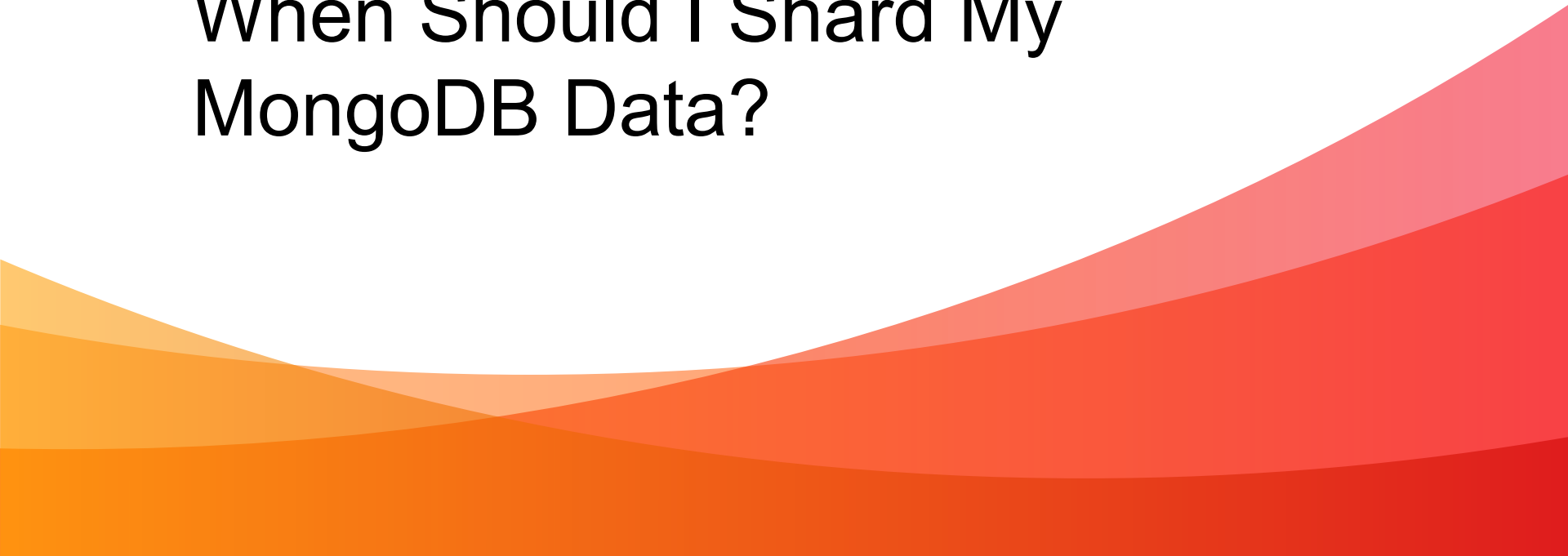


Zoned Sharding

- Can use range or hashed sharding.
- Partitions based off shard-key and Zones which can be mapped to a locality.



When Should I Shard My MongoDB Data?



Horizontal vs Vertical Scaling

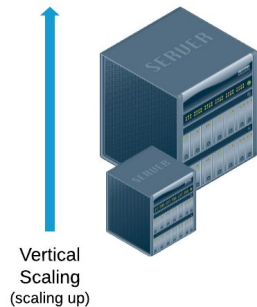
Horizontal (Scaling Out)

Many smaller nodes (less RAM, CPU, IO)

Can be harder to keep in sync

More resistant to failure, if architected well

Many smaller shards = faster backups, but must be cluster consistent



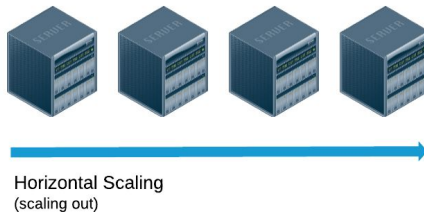
Vertical (Scaling Up)

Fewer, larger nodes (more RAM, CPU)

Less components to keep in sync

More single points of failure

Bigger Database = longer backup times



Horizontal vs Vertical Scaling Terms

- Shard Key – field or multiple fields in a document that are used to distribute the collections documents across shards.
- Chunks – term for each piece of data partitioned in a sharded collection.
- Shard Balancer – Process that runs in the background to migrate chunks across shard to evenly balance data distribution in the cluster.
- Shards – MongoDB instance/Replica Sets that hold a subset of your database.

Horizontal Scaling MongoDB

- Replica Sets – Read Scaling with Secondary Reads.
- Sharded Clusters – Read/Write Scaling through distributed requests to the cluster, achieved through Shard Balancer distributing queries that utilize the Shard Key to their respective chunks on the respective shard.

Vertical Scaling MongoDB

- Costs more, but is easier.
- Increase any or all of the following:
 - CPU
 - RAM
 - I/O
- At some point you will hit a limit (monetary or technology) on how much larger you can grow.

When Should I Pick Sharding Instead of Scaling Up?

- When you've hit a limit to how big you should grow.
- When it's cheaper to have many smaller instances as opposed to a few bigger ones.
- Data set requires, or is forecasted to require, more RAM than a server has.
- Backups are taking too long.
- When your application is greenfield and you can predict high throughput and data growth.

When Should I Pick Scaling Up Over Sharding?

- Operational complexities outweigh the benefits of sharding.
- Data growth doesn't necessitate sharding, and an appropriate purge/archive strategy can be utilized.
- Data never expected to grow larger than 1 terabyte.

How Many Shards? Existing Applications

- Understand your working set size
- Understand disk space, RAM and I/O requirements
- Rule of thumb:
 - 1-2 terabytes of data per shard is best
- Know your application
 - Schema design can lead to more/less shards

How Many Shards? New Applications

- New applications
 - Indexes are 10% of the data size on average.
 - Use tools like Faker or Mockaroo to simulate document sizes and extrapolate from there.

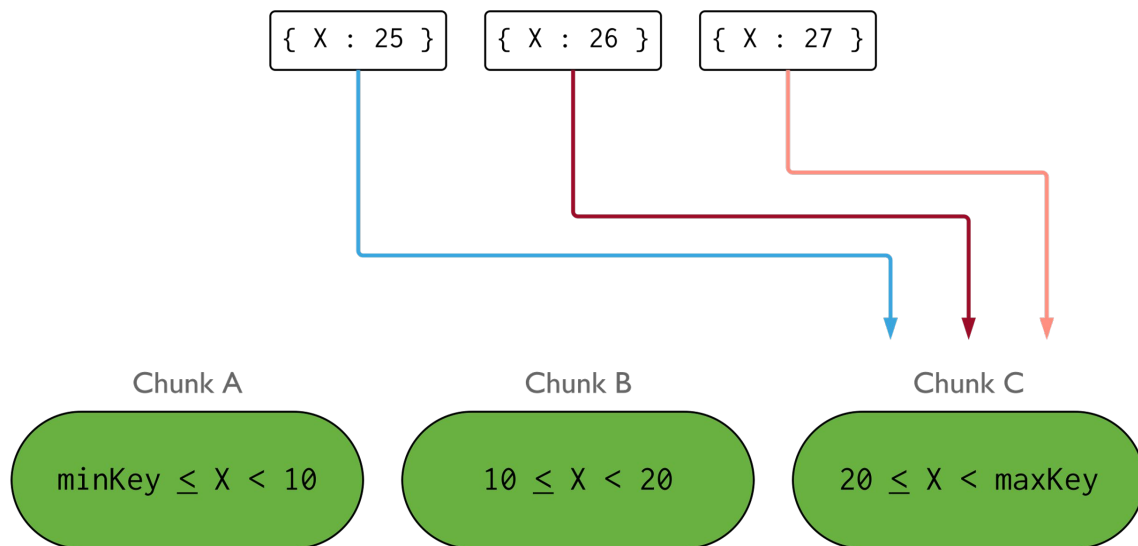


The “Gotchas” of Sharding



The “Gotchas” of Sharding

- Choosing your shard key will make or break your application.
 - Avoid Monotonically increasing shard keys



The “Gotchas” of Sharding

- MongoDB 4.4 allows you to add fields to your shard key.

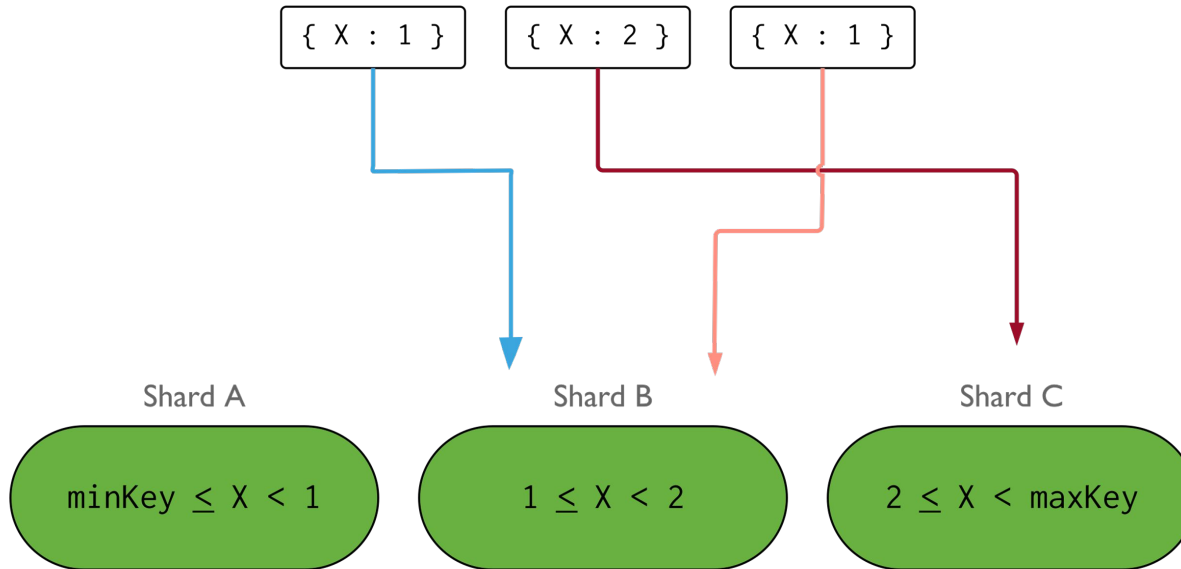
Example:

Shard key { customer_id: 1 } can become

Shard key { customer_id: 1, order_id: 1 }

The “Gotchas” of Sharding

- Avoid shard keys with low cardinality.



The “Gotchas” of Sharding

- Waiting too long to add a shard can be painful.
- Under provisioning config server replica sets.

The Benefits of Sharding

The background of the slide features abstract, wavy, overlapping shapes in shades of orange and red, creating a modern and dynamic aesthetic.

The Benefits of Sharding

- Overcome hardware limitations.
- Overcome infrastructure limitations.
- Able to partition data geographically or in hot/cold segments, easily opens up more use cases.
- Failure isolation.
- Can speed up queries by distributing the load.
- Quicker Restores from backups.

Thanks!

Any questions?

You can find me at:

- @mikegray831
- mike.grayson@percona.com
- <https://www.linkedin.com/in/mikegrayson/>

THANK YOU !



PERCONA
LIVEONLINE
MAY 12 - 13th
2021