Everything You Need to Know About Sharding

Dylan Tong
dylan.tong@mongodb.com
Senior Solutions Architect

Agenda

Overview

- What is sharding?
- Why and what should I use sharding for?

Building your First Sharded Cluster

- What do I need to know to succeed with sharding?

Q&A

What is Sharding?

Sharding is a means of partitioning data across servers to enable:

Geo-Locality

to support geographically distributed deployments to support optimal UX for customers across vast geographies.

Scale

needed by modern applications to support massive work loads and data volume.

Hardware Optimizations

on Performance vs. Cost

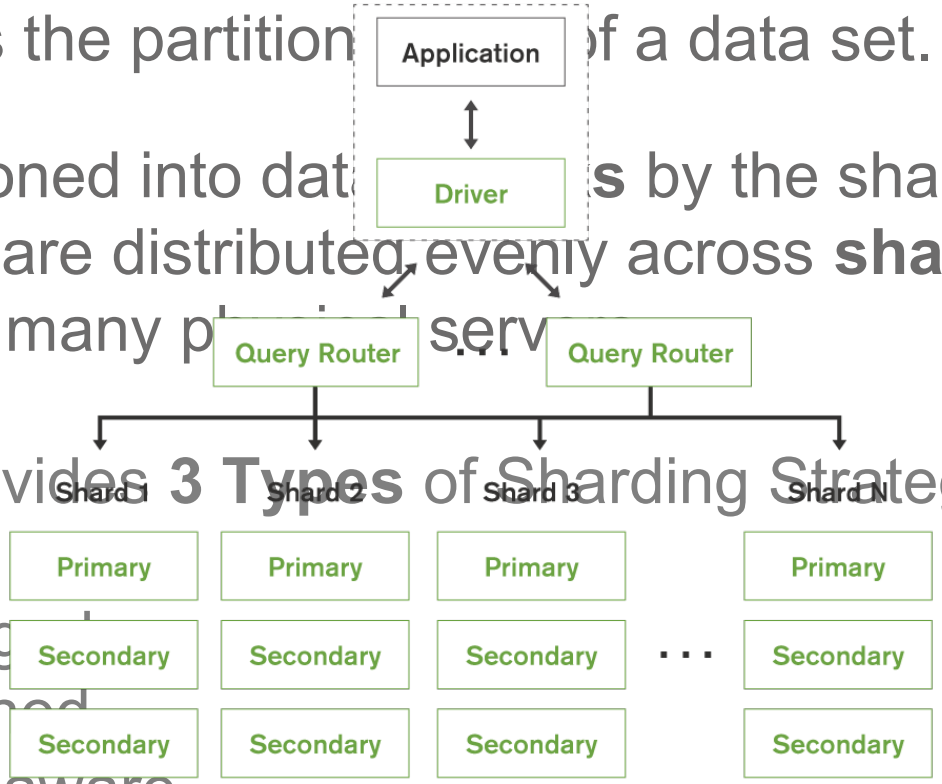
Lower Recovery Times

to make “Recovery Time Objectives” (RTO) feasible.

What is Sharding?

Sharding involves a **shard key** *defined* by a data modeler that describes the partition of a data set.

Data is partitioned into data **s** by the shard key, and these chunks are distributed evenly across **shards** that reside across many p serv

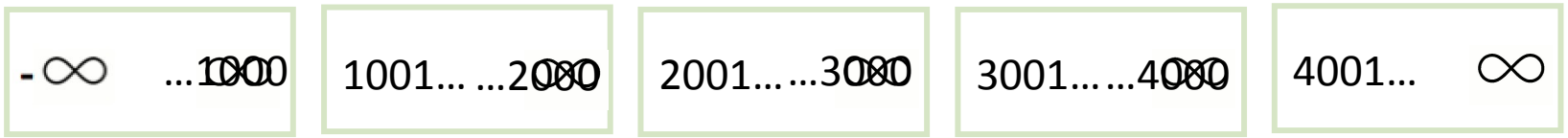


MongoDB provides **3 Types** of Sharding Strategies:

- Rang
- Hash
- Tag-aware

Range Sharding

Shard Key: **{deviceId}**



Composite Keys Supported: **{deviceId, timestamp}**

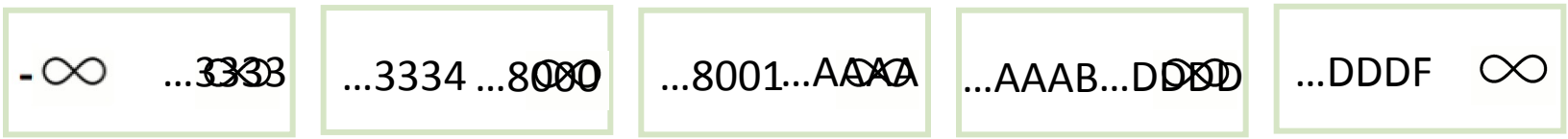
Hash Sharding

Hash Sharding is a subset of Range Sharding.

MongoDB apples a MD5 hash on the key when a hash shard key is used:

Hash Shard Key(deviceId) = MD5(deviceId)

Ensures data is distributed randomly within the range of MD5 values



Tag-aware Sharding

Tag-aware sharding allows subset of shards to be tagged, and assigned to a sub-range of the shard-key.

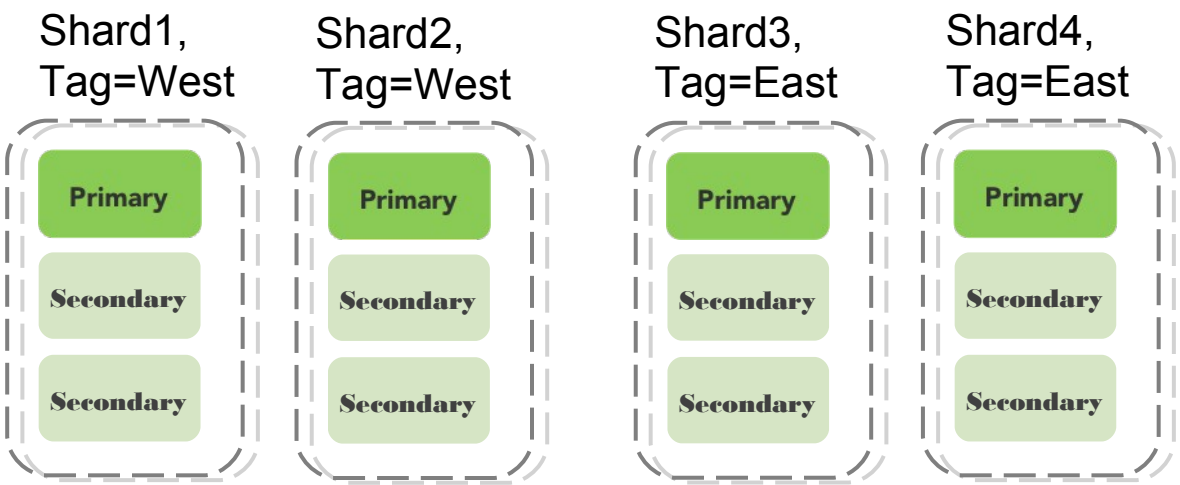
Example: Sharding User Data belong to users from 100 “regions”

Collection: Users, **Shard Key:** {uId, regionCode}

Tag	Start	End
West	MinKey, MinKey	MaxKey,50
East	MinKey, 50	MaxKey, MaxKey

Assign Regions 1-50 to the West

Assign Regions 51-100 to the East



Applying Sharding

Usage	Required Strategy
Scale	Range or Hash
Geo-Locality	Tag-aware
Hardware Optimization	Tag-aware
Lower Recovery Times	Range or Hash

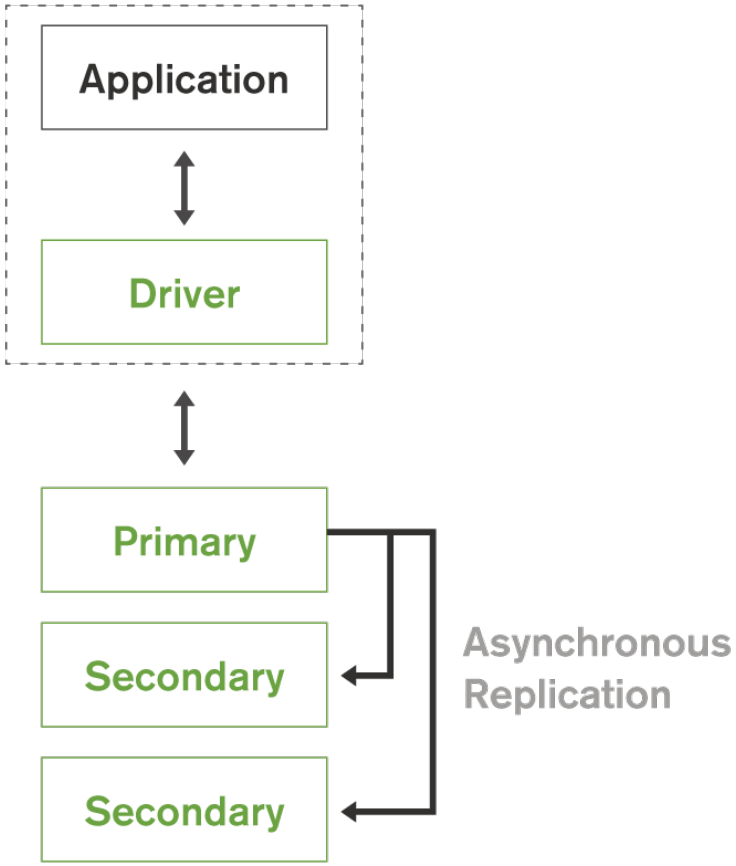
Sharding for Scale

Performance Scale: Throughput and Latency

Data Scale: Cardinality, Data Volume

Typical Small Deployment

Replica Set



Highly Available

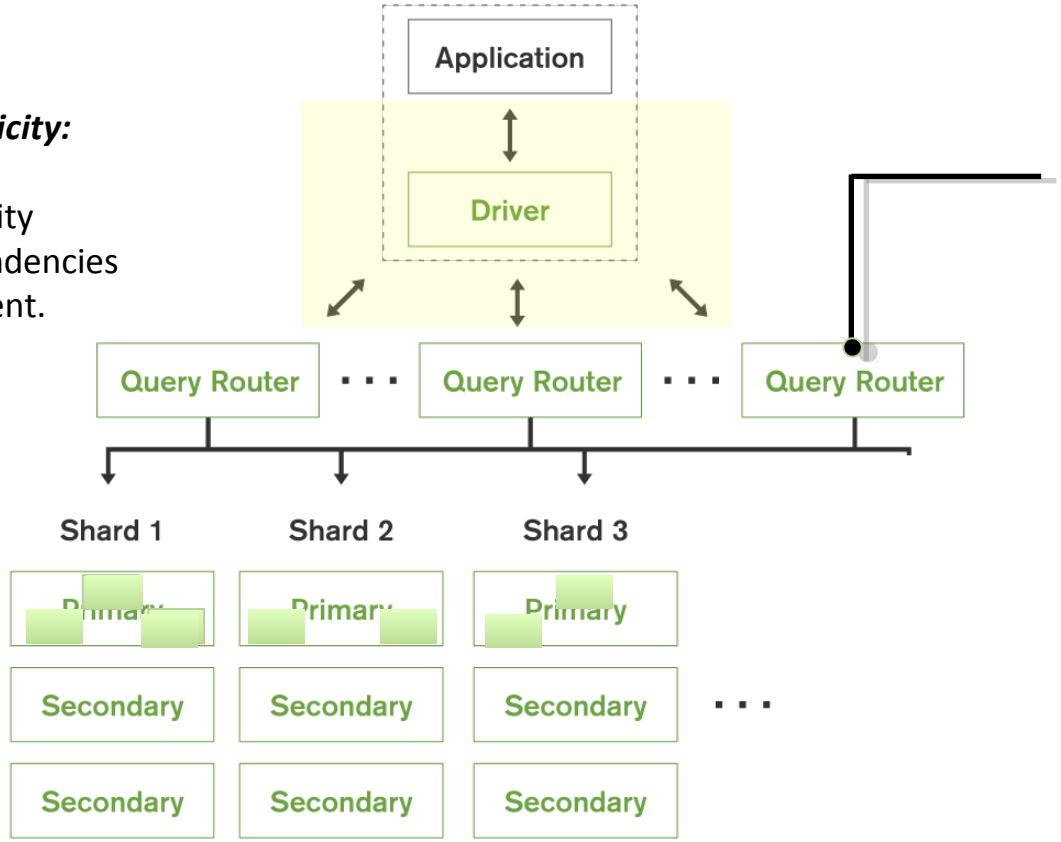
but not Scalable

Writes	Reads
Limited by capacity of the Primary's host	**When Immediate Consistency Matters**: Limited by capacity of the Primary's host
	When Eventual Consistency is Acceptable: Limited by capacity of available replicaSet members

Sharded Architecture

Decoupling for Development and Operational Simplicity:

Ops can add capacity without app dependencies and Dev involvement.

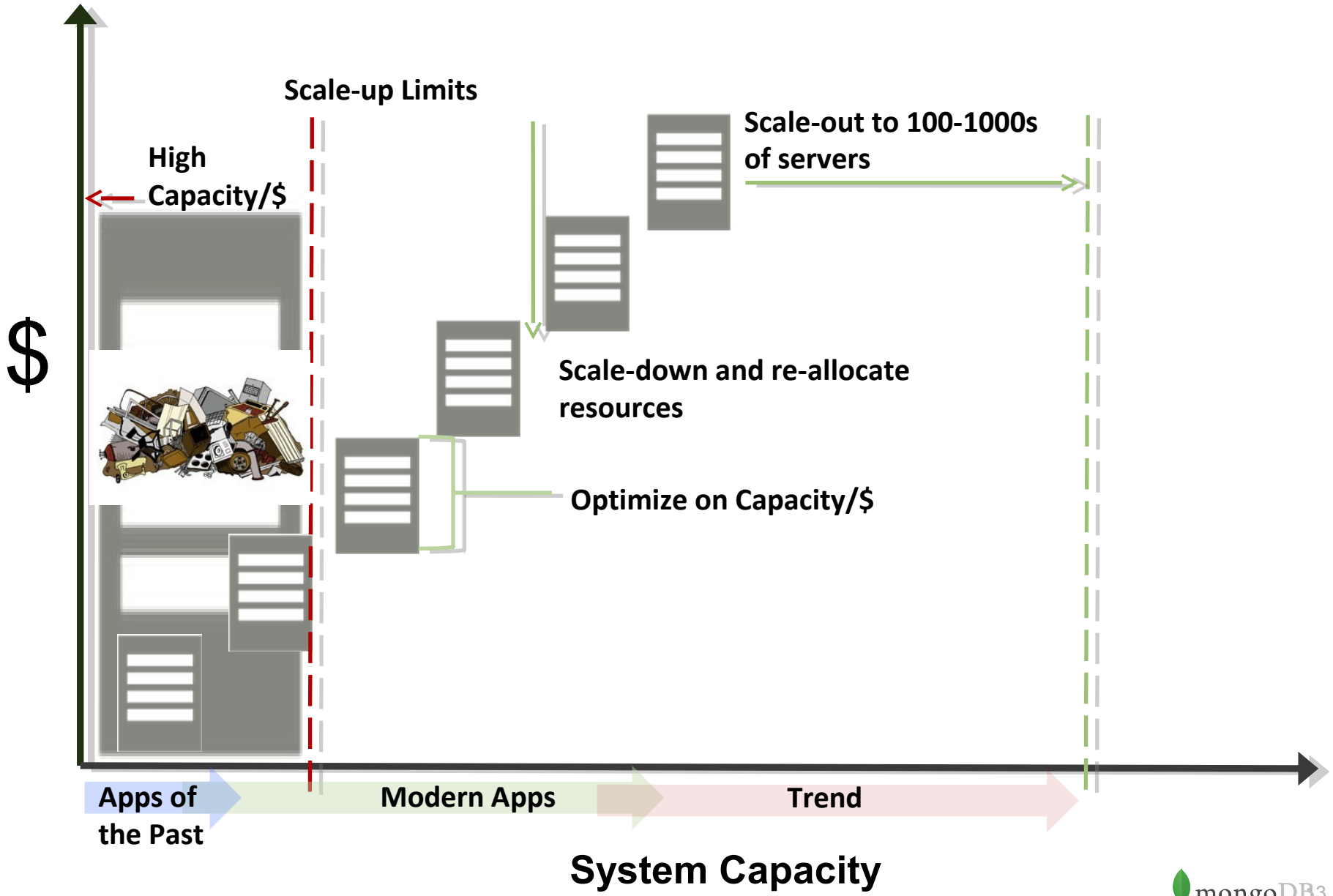


Query Routing: database operations are transparently routed across the cluster through a routing proxy process (software).

Auto-balancing: data is partitioned based on a shard key, and automatically balanced across shards by MongoDB

Horizontal Scalability: load is distributed and resources are pooled across commodity servers.

Value of Scale-out Architecture



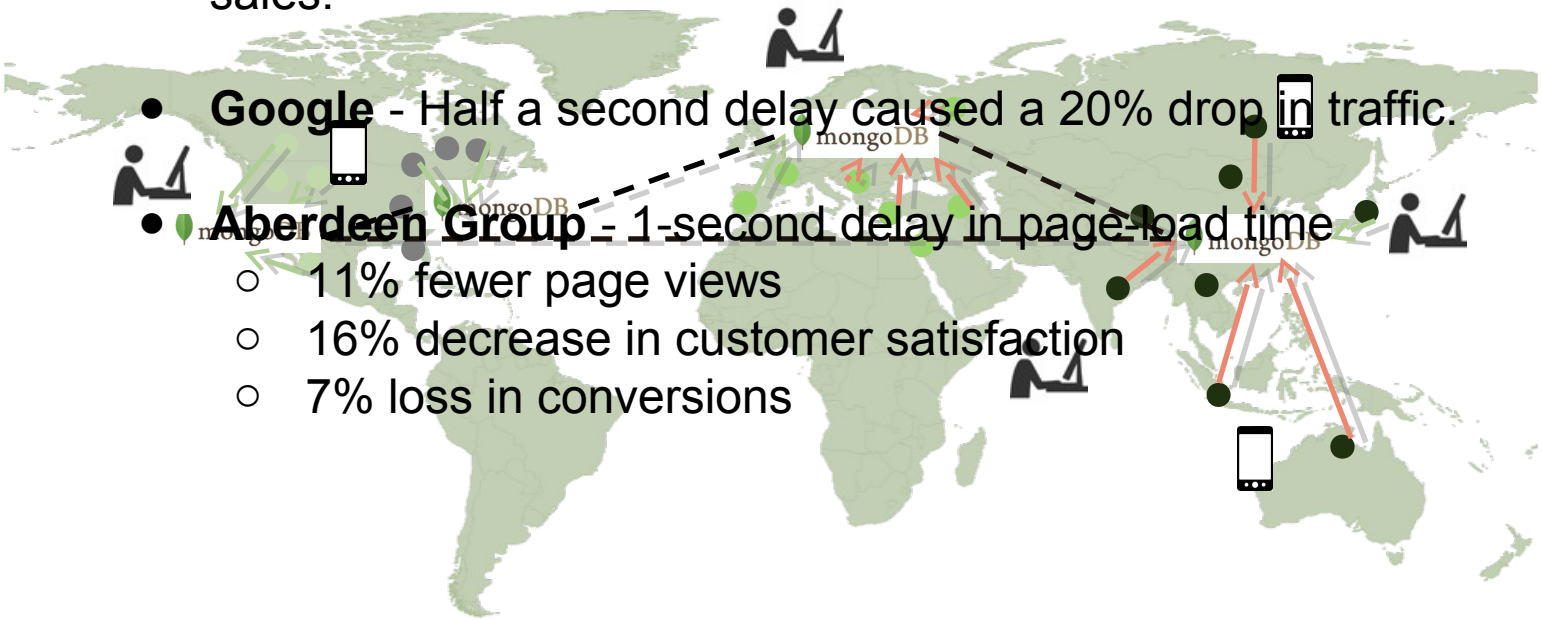
mongoDB3

Sharding for Geo-Locality

Adobe Cloud Services among other popular consumer and Enterprise services use sharding to run servers across multiple data centers across geographies.

Network latency from West to East is ~80ms

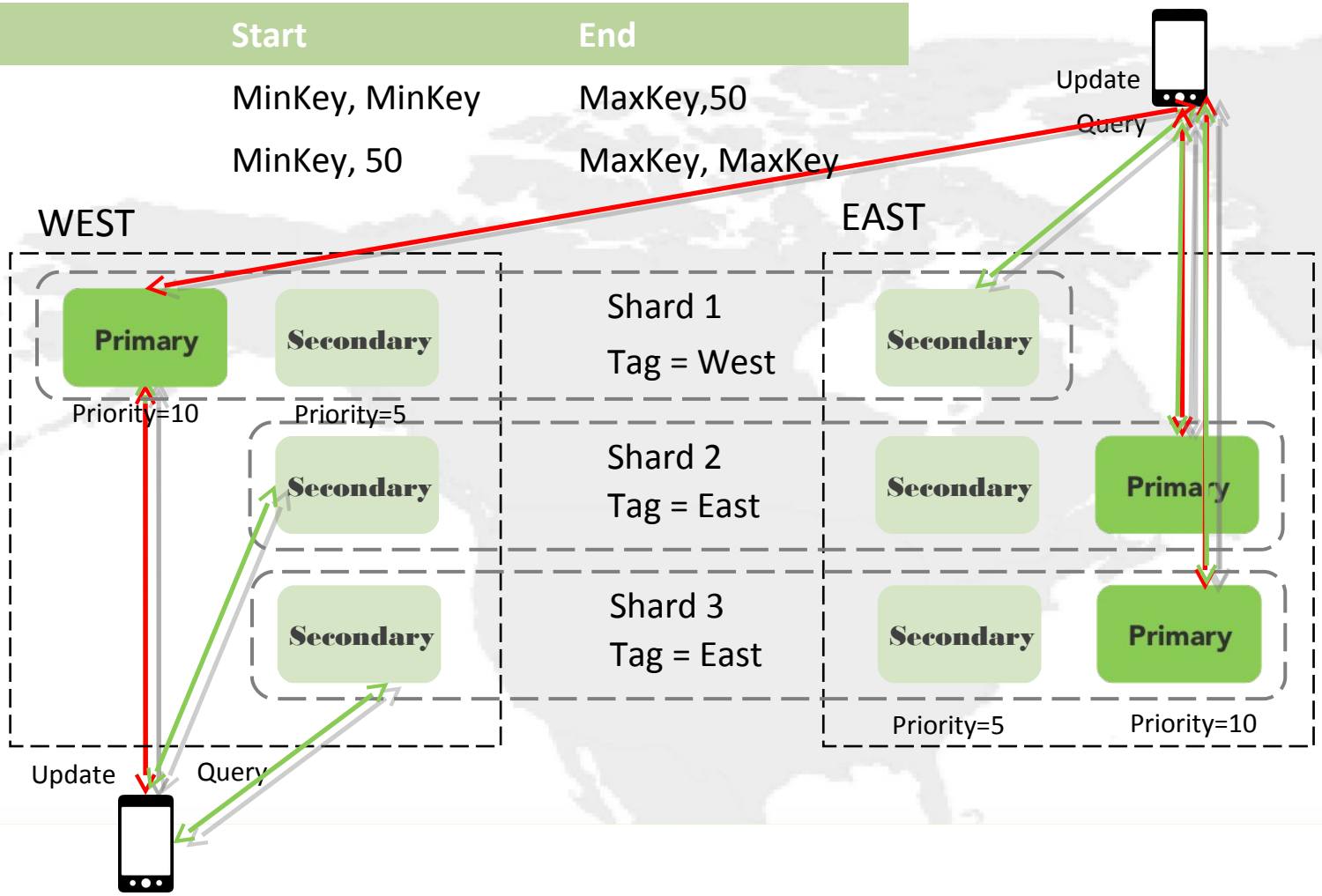
- **Amazon** - Every 1/10 second delay resulted in 1% loss of sales.
- **Google** - Half a second delay caused a 20% drop in traffic.
- **Aberdeen Group** - 1-second delay in page-load time
  - 11% fewer page views
  - 16% decrease in customer satisfaction
  - 7% loss in conversions



Multi-Active DCs via Tag-aware Sharding

Collection: Users, **Shard Key:** {uId, regionCode}

Tag	Start	End
West	MinKey, MinKey	MaxKey,50
East	MinKey, 50	MaxKey, MaxKey



Local Reads (Eg. Read Preference = *Nearest*)

Optimizing Latency and Cost

Magnitudes of Difference in Speed

Event	Latency	Normalized to 1 s
RAM access	120 ns	6 min
SSD access	150 µs	6 days
HDD access	10 ms	12 months

Magnitudes of Difference in Cost

Storage Type	Avg. Cost ($/GB)	Cost at 100TB ($)
RAM	5.50	550K
SSD	0.50-1.00	50K to 100K
HDD	0.03	3K

Optimizing Latency and Cost

Use Case: Sensor data collected from millions of devices. Data used for real-time decision automation, real-time monitoring and historical reporting.

Data Type	Description	Latency SLA	Data Volume
Meta Data	Fast look-ups to drive real-time decisions	95th Percentile < 1ms	< 1 TB
Last 90 days of Metrics	95+% of data reported and monitored	95th Percentile < 30ms	< 10 TB
Historic	Used for historic reporting. Access infrequently	95th Percentile < 2s	> 100TB

Hardware Optimizations

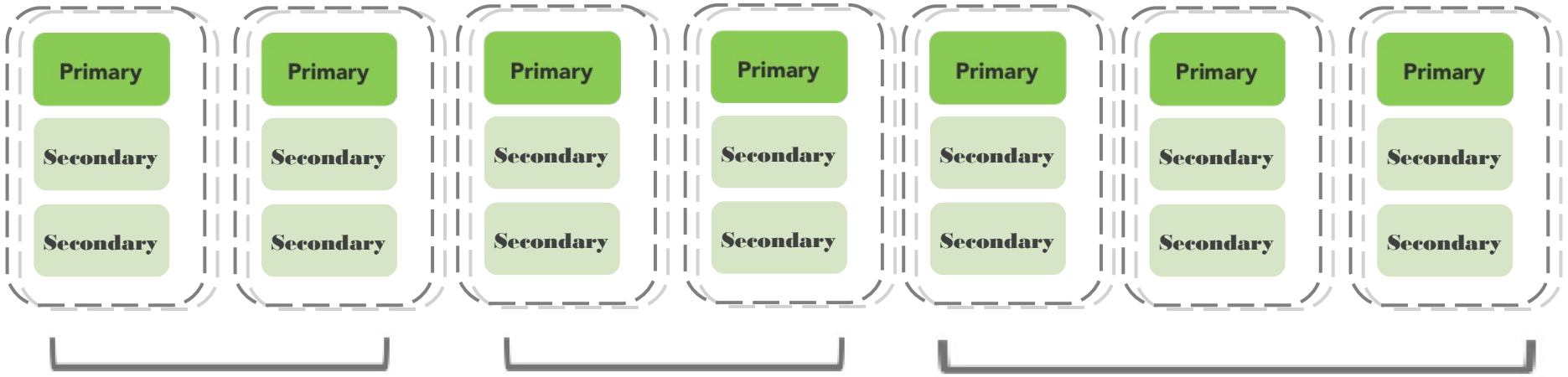
Collections	Tag	Start	End
Meta			
Metrics	Flash	MinKey,MinKey	MaxKey, 90 days ago
Metrics	Archive	MinKey,>90 days ago	MaxKey, MaxKey

Collection	Shard Key
Meta	DeviceId
Metrics	DeviceId, Timestamp

High Memory Ratio, Fast Cores HDD

Medium Memory Ratio, High Compute SSDs

Low Memory Ratio, Medium Compute HHD



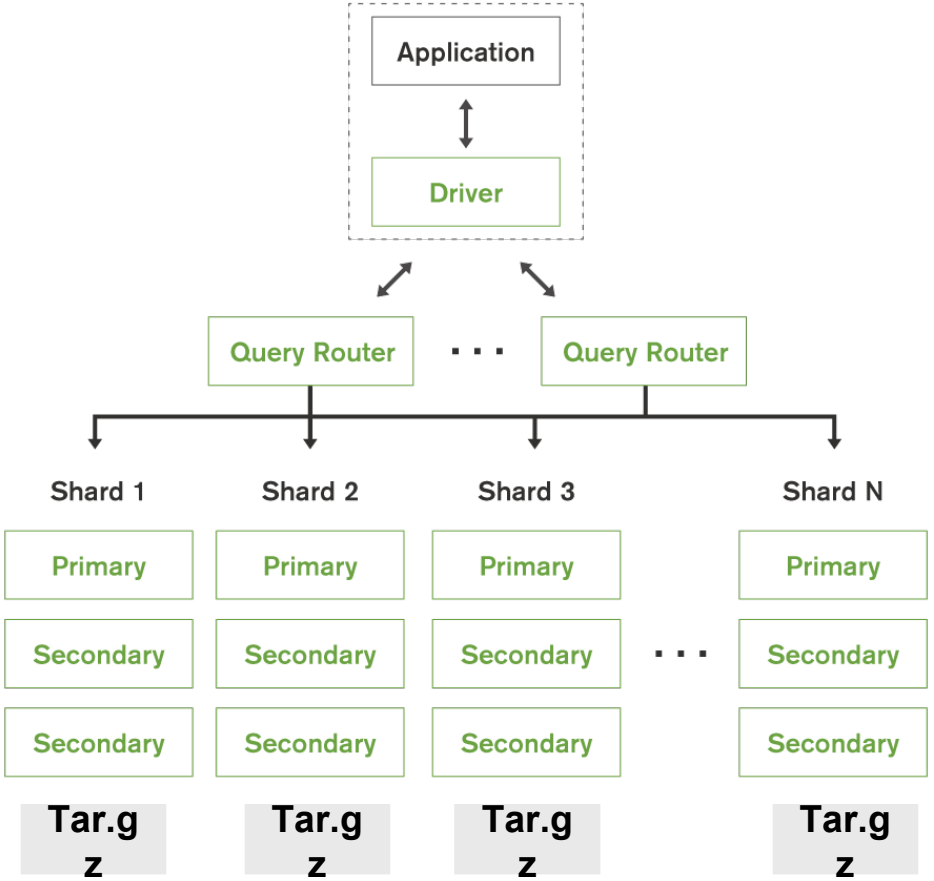
Tag: Cache

Tag: Flash

Tag: Archive

Restoration Times

Scenario: Application bug causes logical corruption of the data, and the database needs to be rolled back to a previous PIT. What's RTO does your business require in this event?



Total DB Snapshot Size = 100TB

N = 10

10X10TB Snapshots generated and/or transferred in parallel

N = 100

100X1TB Snapshots generated and/or transferred in parallel.

Potentially 10X faster restoration time.

Building Your First Sharded Cluster

Life-cycle of Sharding

Product Definition: Starts with an idea to build something big!

Predictive Maintenance Platform: a cloud platform for building predictive maintenance applications and services—such as a service that monitors various vehicle components by collecting data from sensors, and automatically prescribes actions to take.

- Allow tenant to register, ingest and modify data collected by sensors
- Define and apply workflows and business rules
- Publish and subscribe to notifications
- Data access API for things like reporting

Life-cycle of Sharding

Data Modeling: Do I need to Shard?

Throughput: data from millions of sensors updated in real-time

Latency: The value of certain attributes need to be access with 95th percentile < 10ms to support real-time decisions and automation.

Volume: 1TB of data collected per day. Retained for 5 years.

Life-cycle of Sharding

Data Modeling: Select a Good Shard Key

Critical Step

- Sharding is only effective as the shard key.
- Shard Key attributes are immutable.
- Re-sharding is non-trivial. Requires re-partitioning data.

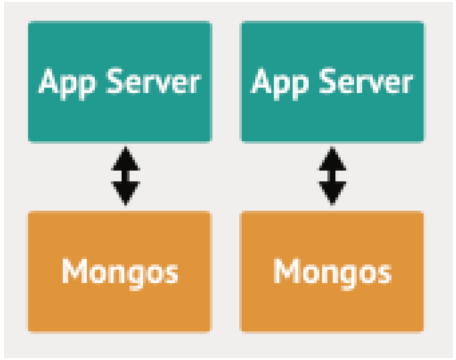
Good Shard Key

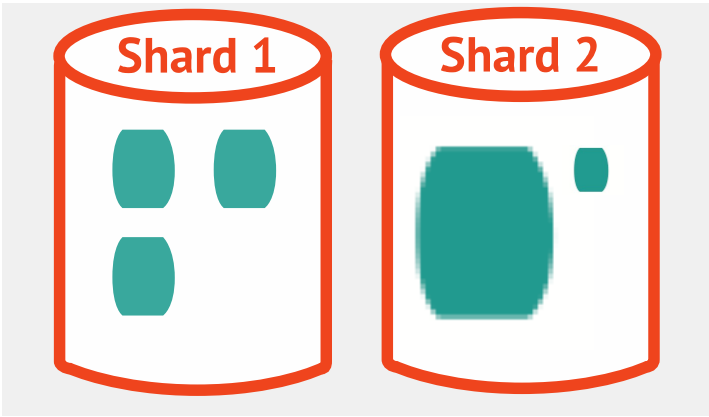
Cardinality
Write Distribution
Query Isolation

Reliability
Index Locality

Cardinality

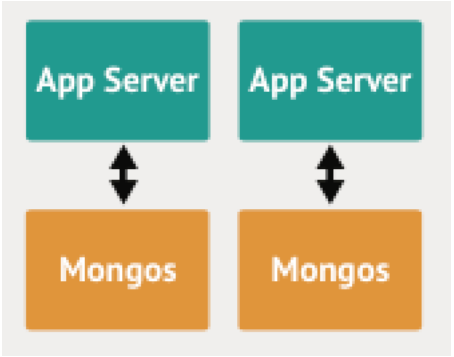
Key = Data Center

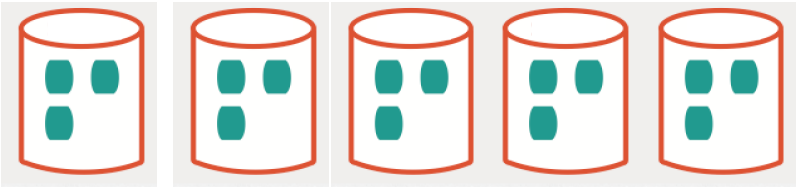




Cardinality

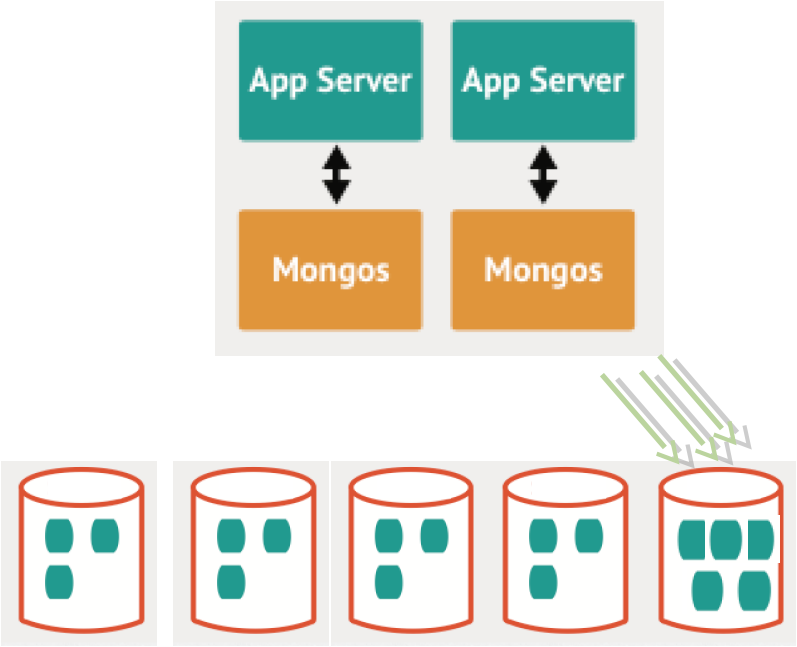
Key = Timestamp





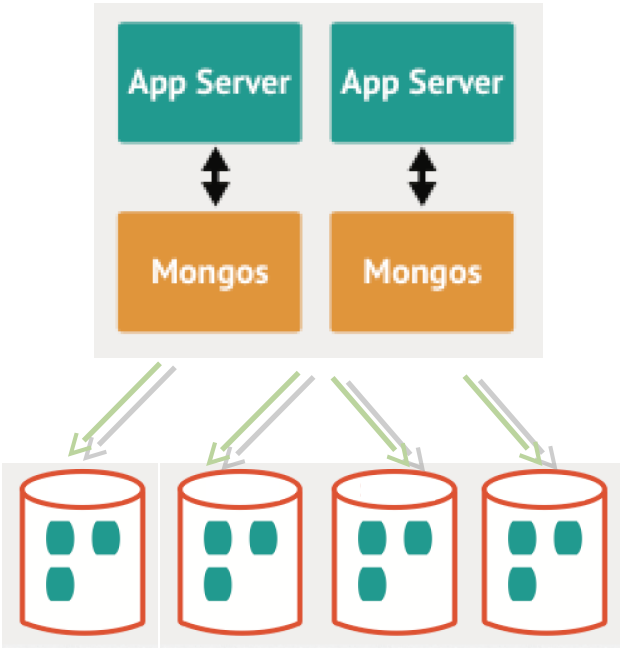
Write Distribution

Key = Timestamp



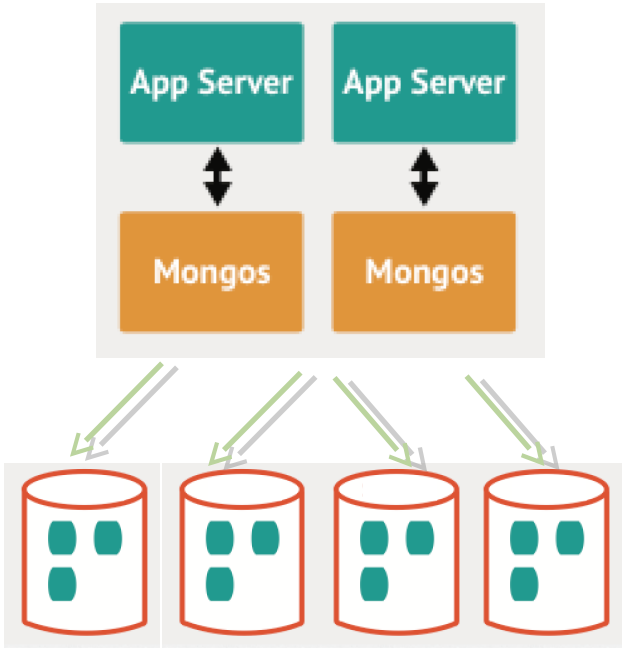
Write Distribution

Key = Hash(Timestamp)



Query Isolation

Key = Hash(Timestamp)

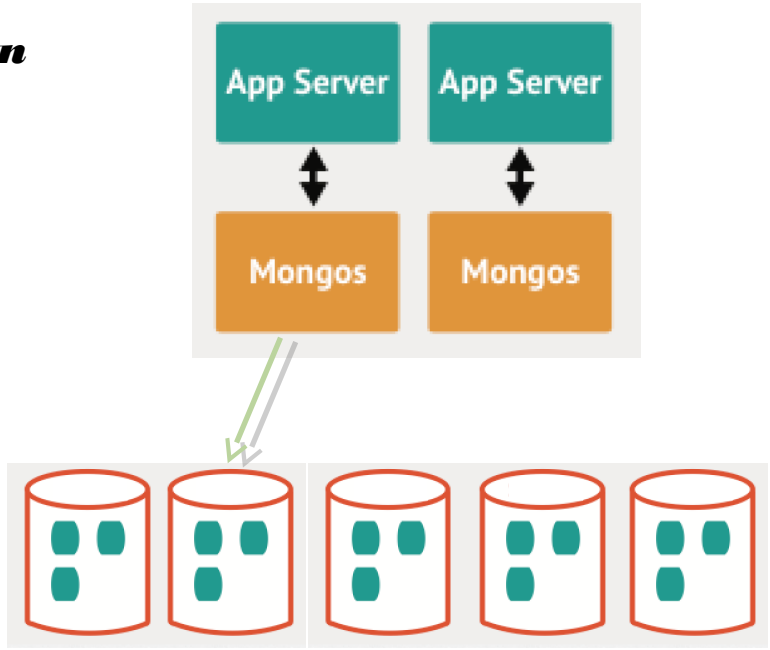


"Scatter-gather Query"

Query Isolation

Key = Hash(DeviceId)

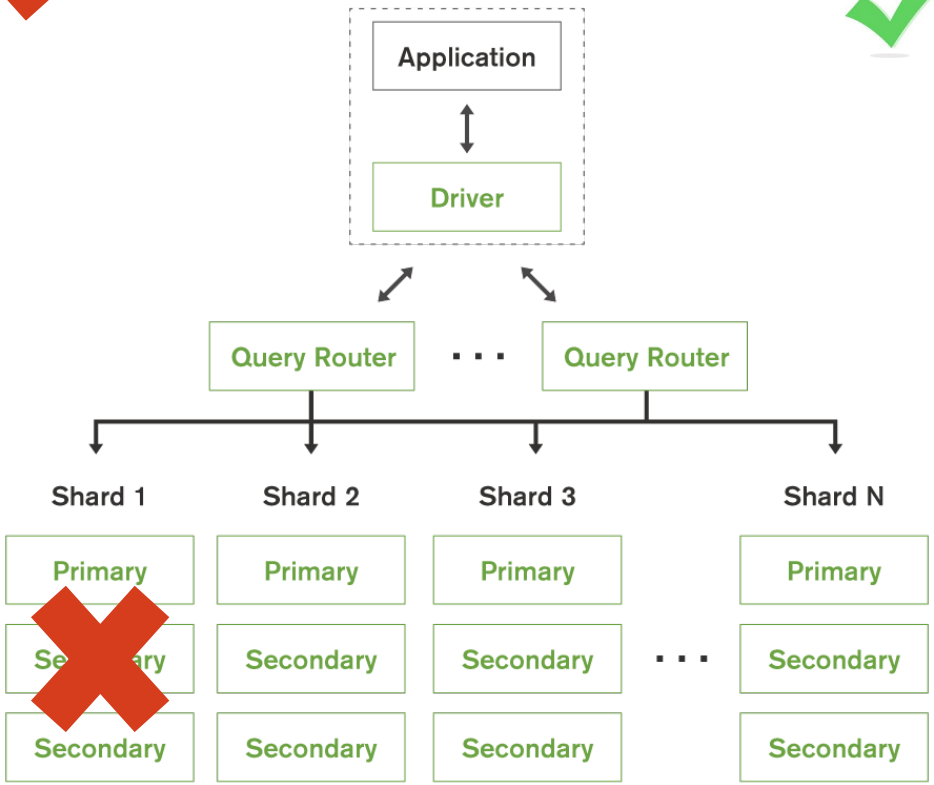
***Assumes bulk of queries on collection are in context of a single deviceId**



Reliability

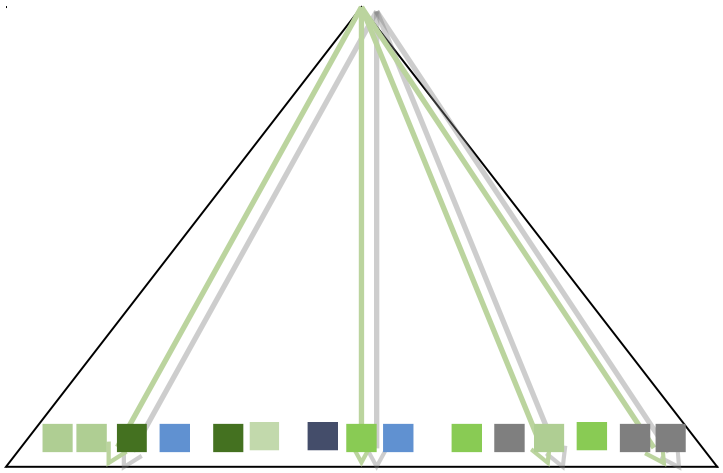
Key = Hash(Timestamp) ✗

Key = Hash(DeviceId ✓

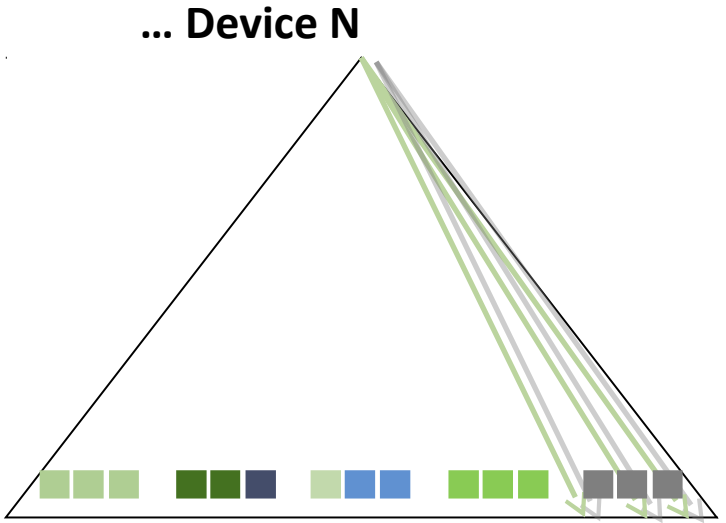


Index Locality

Key = Hash(DeviceId)
Random Access Index



Key = DeviceId, Timestamp
Right Balance Index



Working Set

Right balanced index may only need to be partially in RAM to be effective.

Good Shard Key

Key = DeviceId, Time

- ✓ Cardinality
- ✓ Write Distribution
- ✓ Query Isolation
- ✓ Reliability
- ✓ Index Locality

Life-cycle of Sharding

Performance Testing: Avoid Pitfalls

Best Practices:

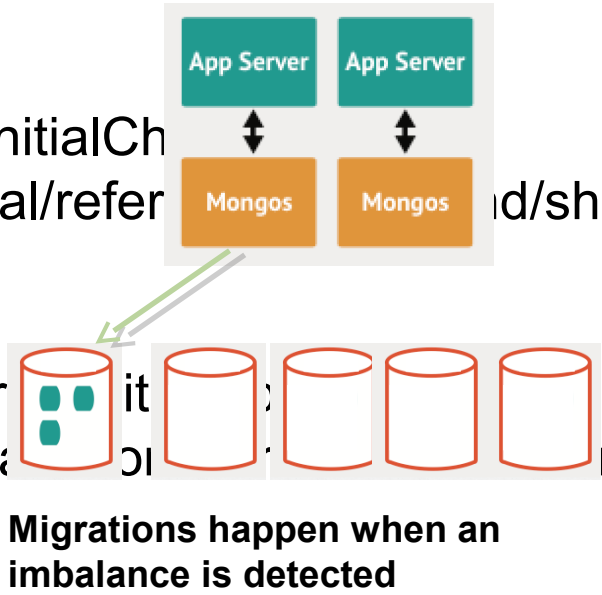
Sharding results in massive performance degradation!

•Pre-split:

1. Hash Shard Key: specify numInitialCh… http://docs.mongodb.org/manual/refer…d/shardCollection/

2. Custom Shard Key: create a pr…it… http://docs.mongodb.org/manua…on-sharded-cluster/

Shards happy: created as chunks grow



Migrations happen when an imbalance is detected

- Run mongos (query router) on app server if possible.

Design & Development | Test/QA | Pre-Production | Production

Life-cycle of Sharding

Capacity Planning: How many shards do I need?

Sizing:
•What are the total resources required for your initial deployment?
•What are the ideal hardware specs, and the # shards necessary?

Capacity Planning: create a model to scale MongoDB for a specific app.
•How do I determine when more shards need to be added?
•How much capacity do I gain from adding a shard?

How Many Servers?

Strategy	Accuracy	Level of Effort	Feasibility of Early Project Analysis
Domain Expert	High to Low: inversely related to complexity of the Application	Low	Yes
Empirical (Load Testing)	High	High	Unlikely

Domain Expert

Normally, performed by MongoDB Solution Architect:
http://bit.ly/1rkXcfN

- What is the document model? Collections, documents, indexes
- What are the major operations?
  - Throughput
  - Latency
- What is the working set? Eg. Last 90 days of orders

Domain Expert

Resource	Methodology
RAM	**Standard:** Working Set + Indexes Adjust more or less depending on latency vs. cost requirements. Very large clusters should account for connection pooling/thread overhead (1MB per active thread)
IOPs	Primarily based on throughput requirements. Writes + estimation on query page faults. Assume random IO. Account for replication, journal and log (note: sequential IO). Ideally, estimated empirically through prototype testing. Experts can use experience from similar applications as an estimate. Spot testing maybe needed.
Storage	Estimate using throughput, document and index size approximations, and retention requirements. Account for overhead like fragmentation if applicable.
CPU	Rarely the bottleneck; a lot less CPU intensive than RDBMs. Using current commodity CPU specs will suffice.
Network	Estimate using throughput and document size approximations.

Sizing by Empirical Testing

- **Sizing** can be more accurately obtained by prototyping your application, and performing load tests on selected hardware.
- **Capacity Planning** can be simultaneously accomplished through load testing.
- **Past Webinars:** http://www.mongodb.com/presentations/webinar-capacity-planning

Strategy:

1. Implement a prototype that can at least simulate major workloads
2. Select an economical server that you plan to scale-out on.
3. Saturate a single replicaSet or shard (maintaining latency SLA as needed). Address bottlenecks, optimize and repeat.
4. Add an additional shard (as well as mongos and clients as needed). Saturate and confirm roughly linear scaling.
5. Repeat step 4 until you are able to model capacity gains (throughput + latency) versus #physical servers.

Operational Scale

Business Critical Operations: How do I manage 100s to 1000s of nodes?

MongoDB Management Services (MMS): https://mms.mongodb.com

MMS Automation

- Automated cluster provisioning
- Automation of daily operational tasks like no-downtime upgrades
- Centralized configuration management

- Real-time monitoring and visualization of cluster health
- Alerting

MMS Backup

- Automated PIT snapshotting of clusters
- PITR support for sharded clusters

MMS Automation

MMS

MMS Automation

MMS Backup

Monitoring

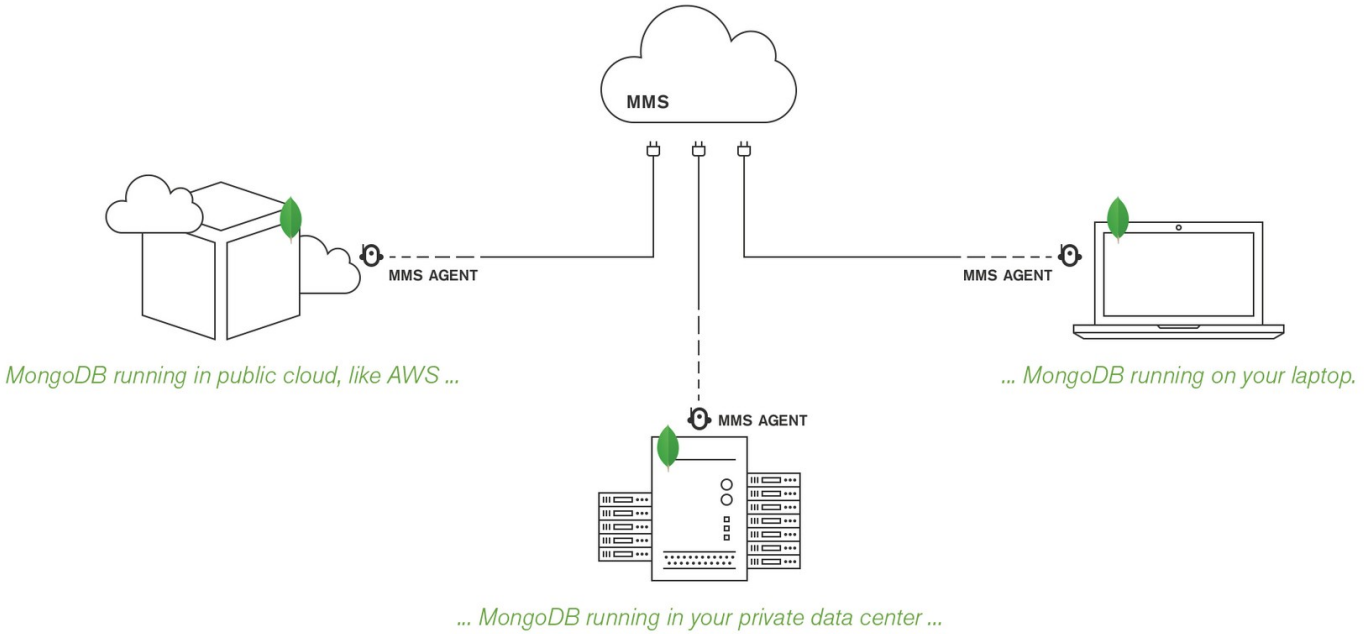
Agent

On-Prem or SaaS

Server Resources (anywhere)

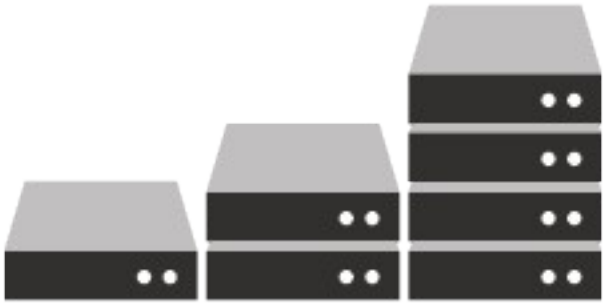
Scalable, Anywhere

Quick Demo



Get Expert Advice on Scaling. For Free.

For a limited time, if you're considering a commercial relationship with MongoDB, you can sign up for a free one hour consult about scaling with one of our MongoDB Engineers.



Sign Up: http://bit.ly/1rkXcfN


Stay tuned after the webinar and take our survey for your chance to win MongoDB swag.
Webinar Q&A
dylan.tong@mongodb.com
mongoDB


mongoDB