# Scaling for Humongous amounts of data with MongoDB


## Alvin Richards

**Technical Director**
**alvin@10gen.com**
**@jonnyeight**
**alvinonmongodb.com**


10gen | The MongoDB Company

mongoDB

# Getting from here to there...

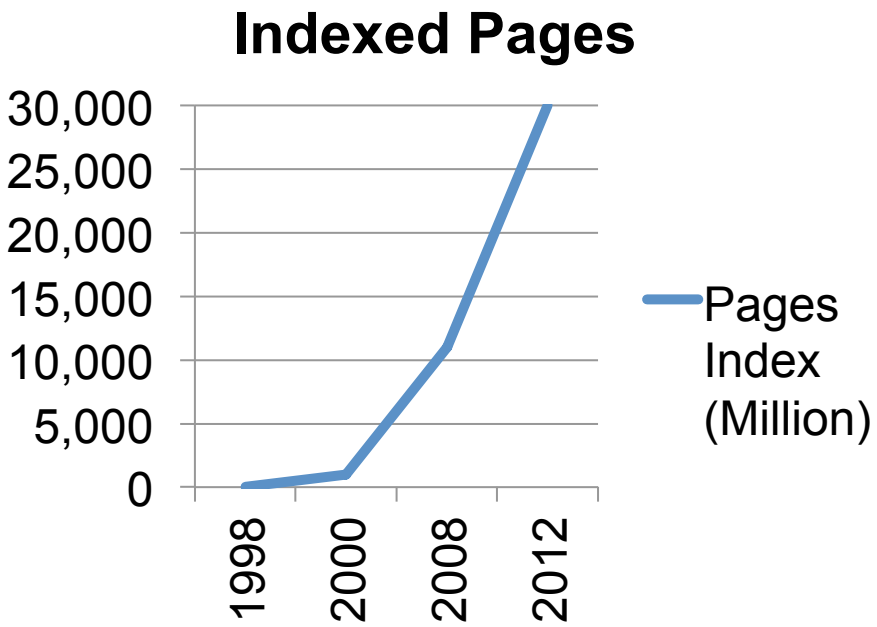


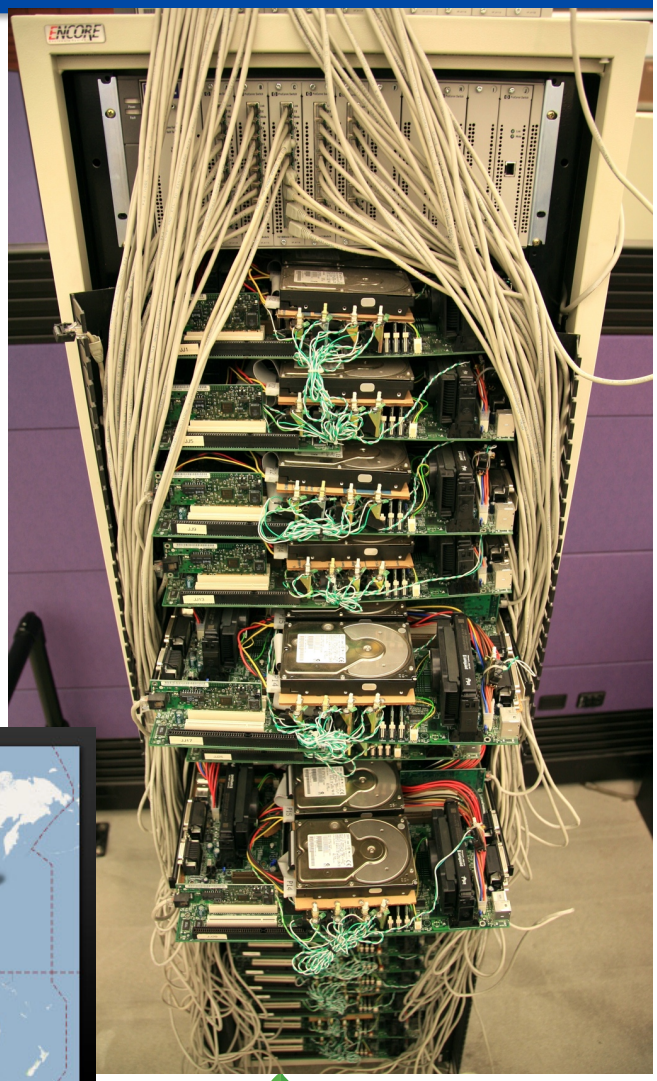
http://bit.ly/OT71M4

# ...probably using one of these

# Why NoSQL at all?

# Growth? Scaling? Cost? Flexibility?





http://bit.ly/VDkDN2
http://bit.ly/108jTHN
http://bit.ly/Wt3fl7
http://bit.ly/Qmg8YD

10gen | The MongoDB Company



mongoDB

# Need a Database that...

- Build a database for scaleout
  - Run on clusters of 100s of commodity machines
- … that enables agile development
- … and is usable for a broad variety of applications

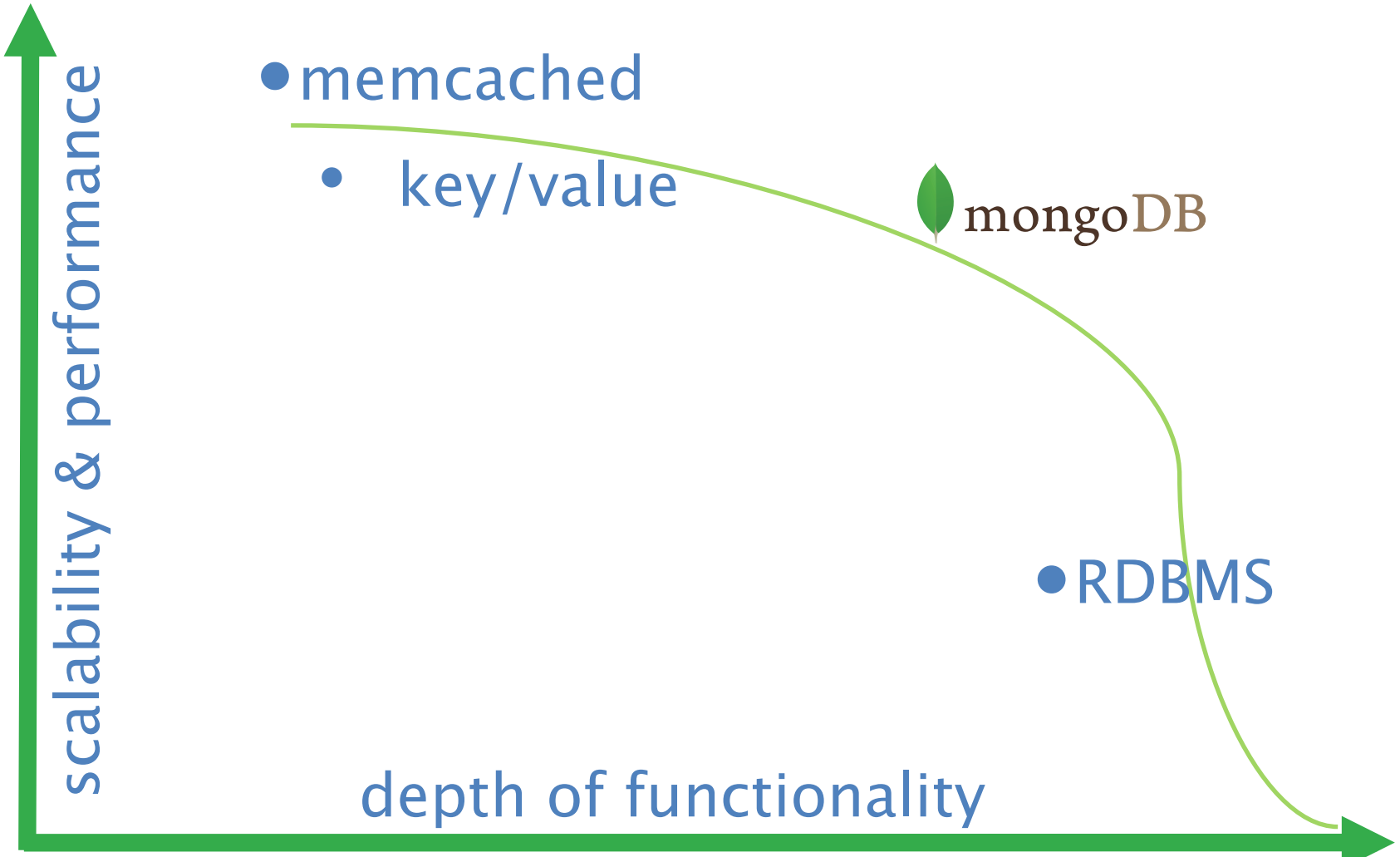
# Is Scaleout Mission Impossible?

- Partitioning of Data
  - Hashes (Dynamo) vs Ranges (Big Table)
  - Physical vs Logical segments

- Consistency
  - Eventually
    - Multi Master updates, resolve conflicts later
  - Immediately
    - Single Master updates, always consistent

# NoSQL and MongoDB

# Tradeoff: Scale vs Functionality



# What MongoDB solves

**Agility**

- Applications store complex data that is easier to model as **documents**
- **Schemaless** DB enables faster development cycles

**Flexibility**

- Relaxed transactional semantics enable **easy scale out**
- **Auto Sharding** for scale down and scale up

**Cost**

- Cost effective operationalize abundant data (clickstreams, logs, tweets, ...)

# How does MongoDB shape up?

- Build a database for scaleout
  - Run on clusters of 100s of commodity machines
- … that enables agile development
- … and is usable for a broad variety of applications

# Data Distribution across nodes - Sharding

Purpose:

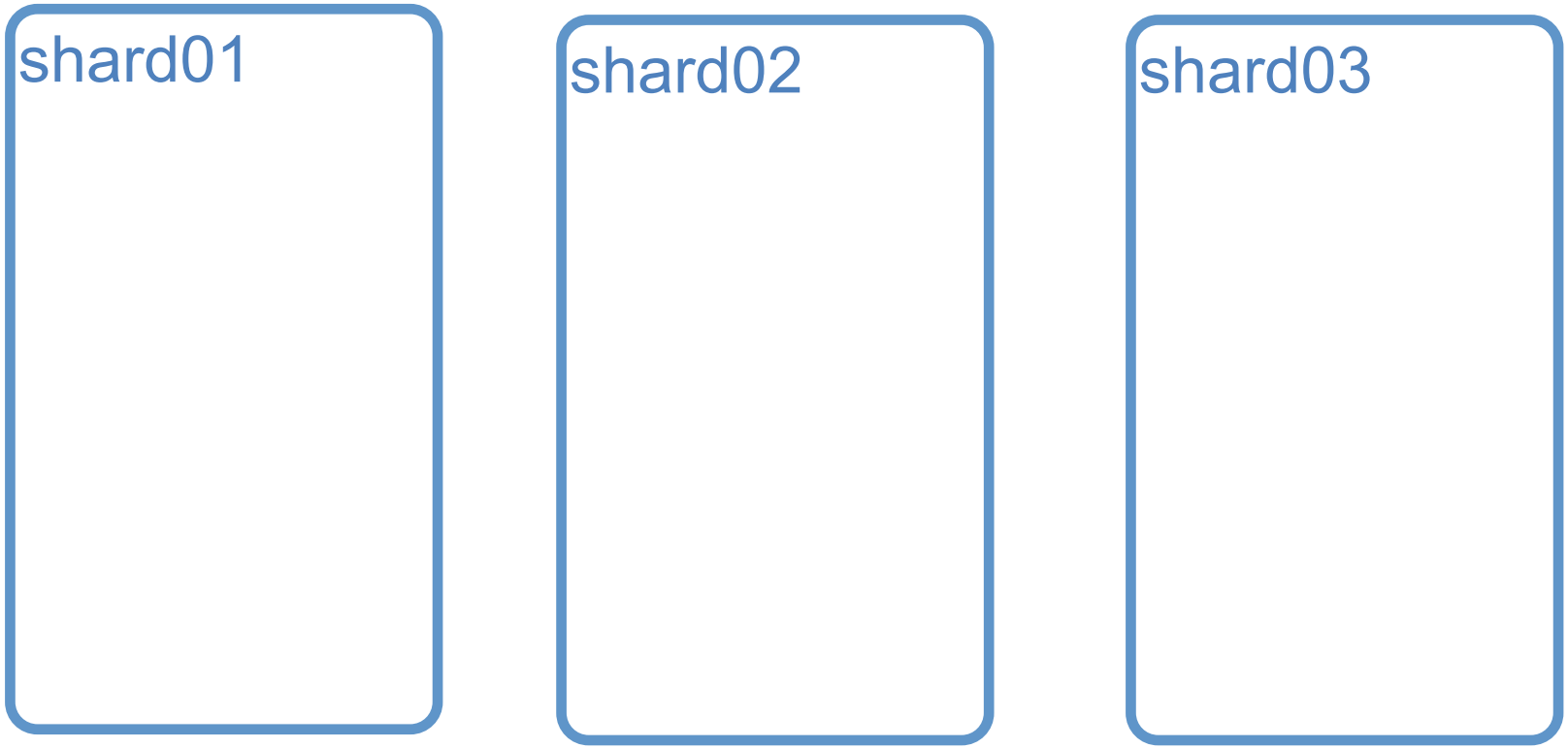
- Aggregate system resources horizontally
- Scaling writes
- Scaling consistent reads

Goals:

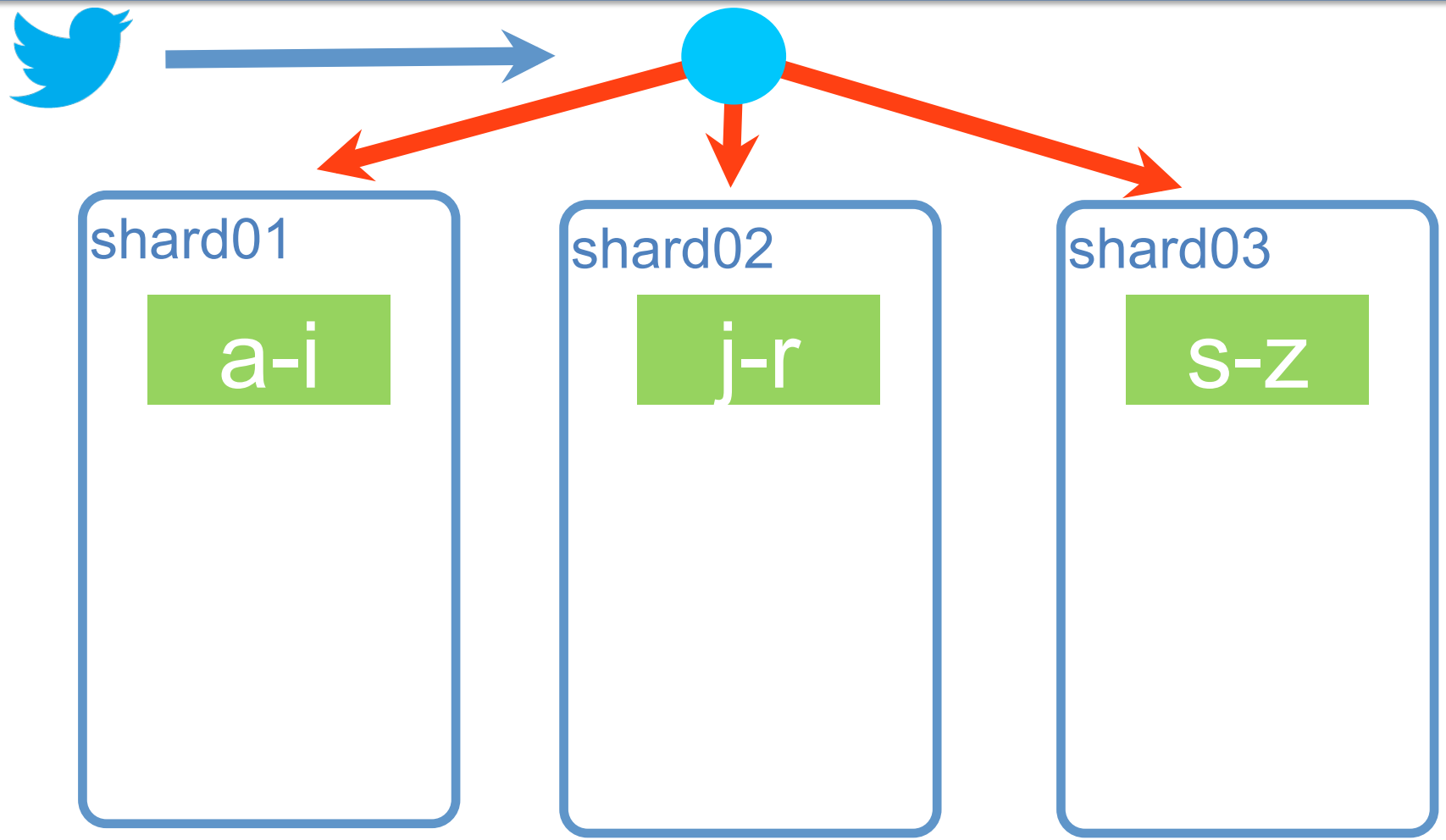
- Data location transparent to your code
- Data distribution is automatic
- No code changes required

# Sharding - Range distribution

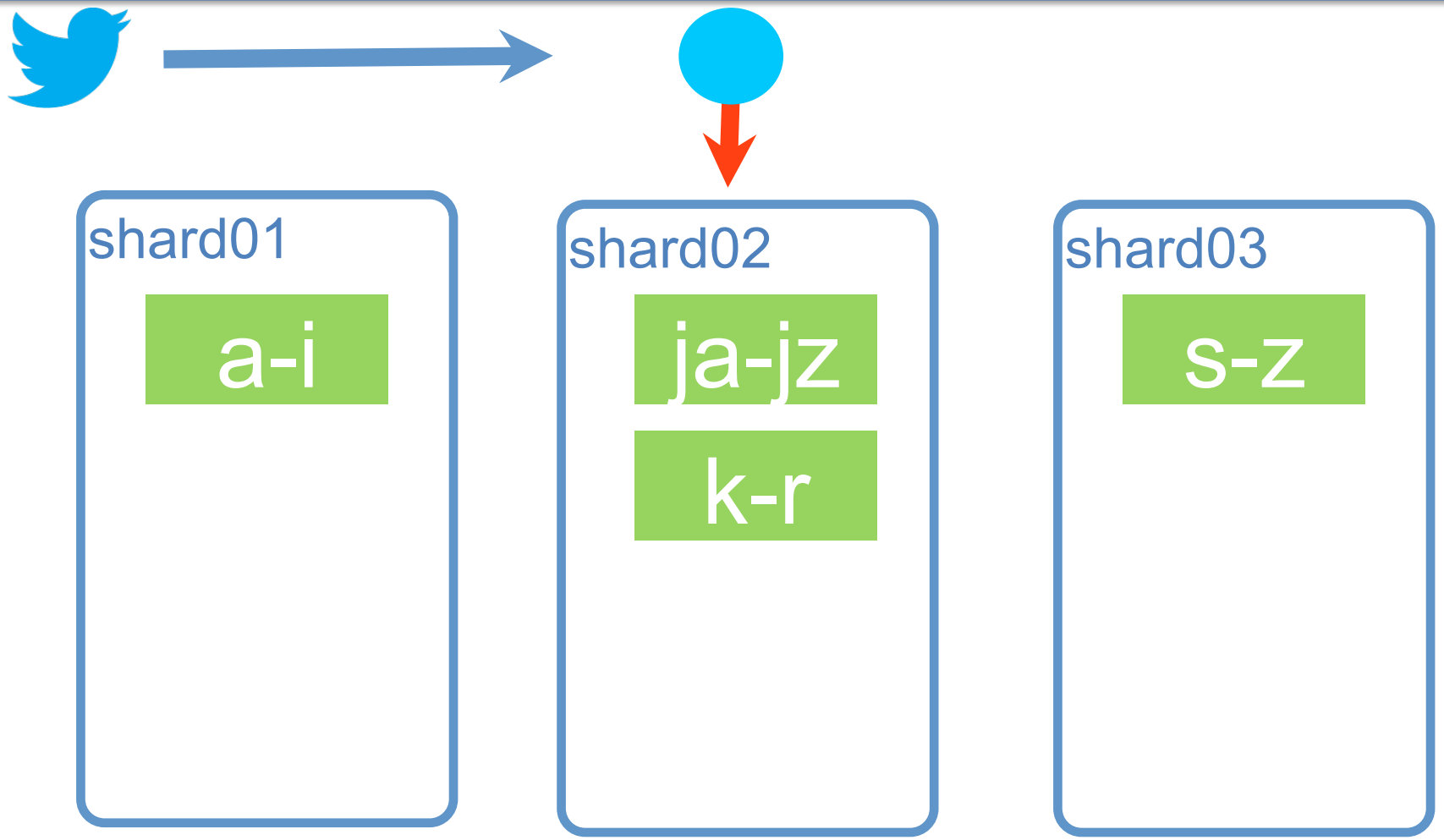
```
sh.shardCollection("test.tweets", {_id: 1} , false)
```



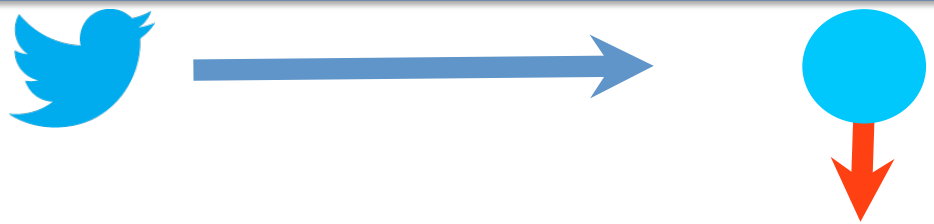
# Sharding - Range distribution



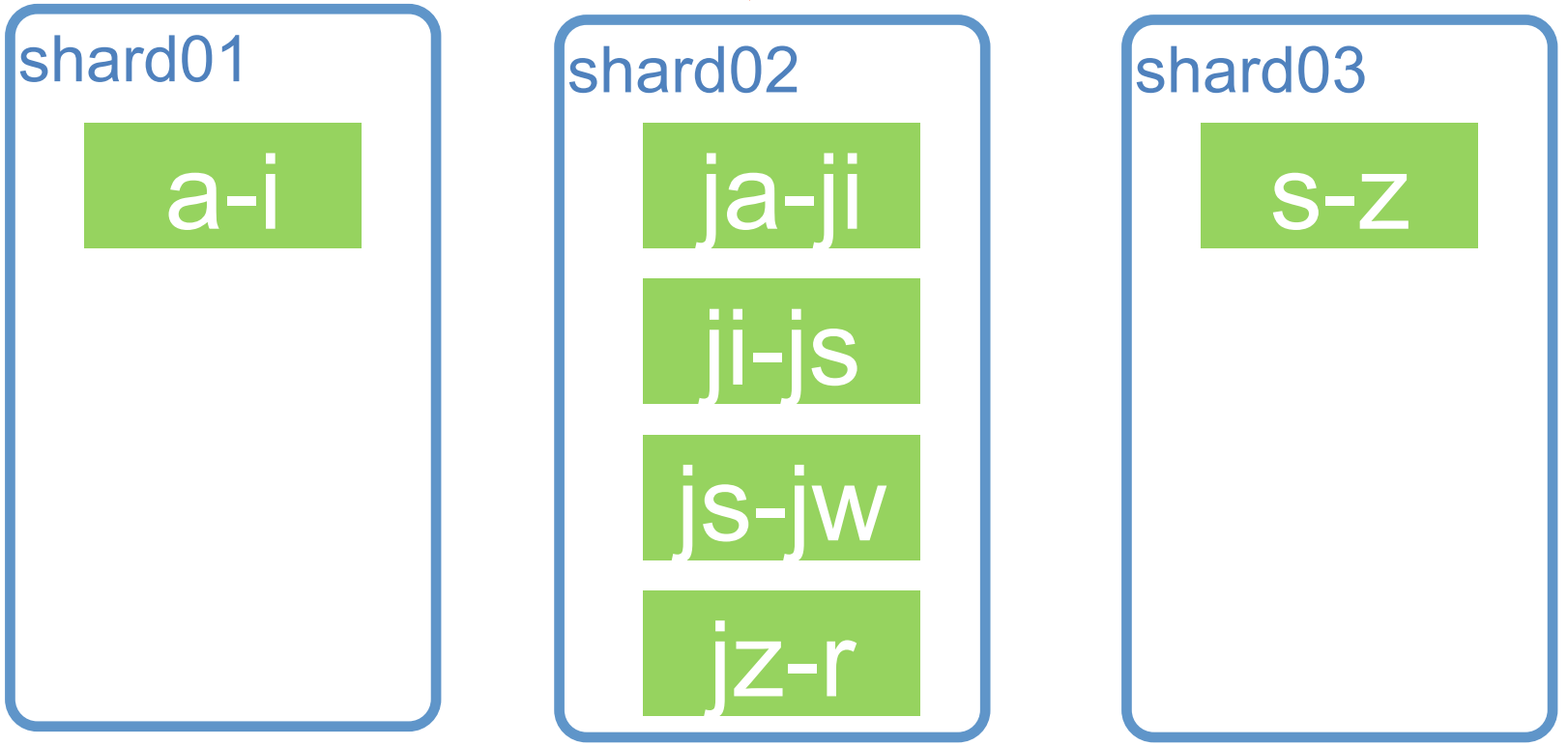
# Sharding - Splits



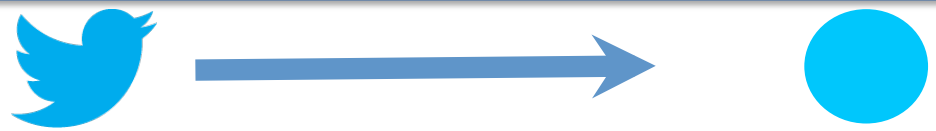
# Sharding - Splits

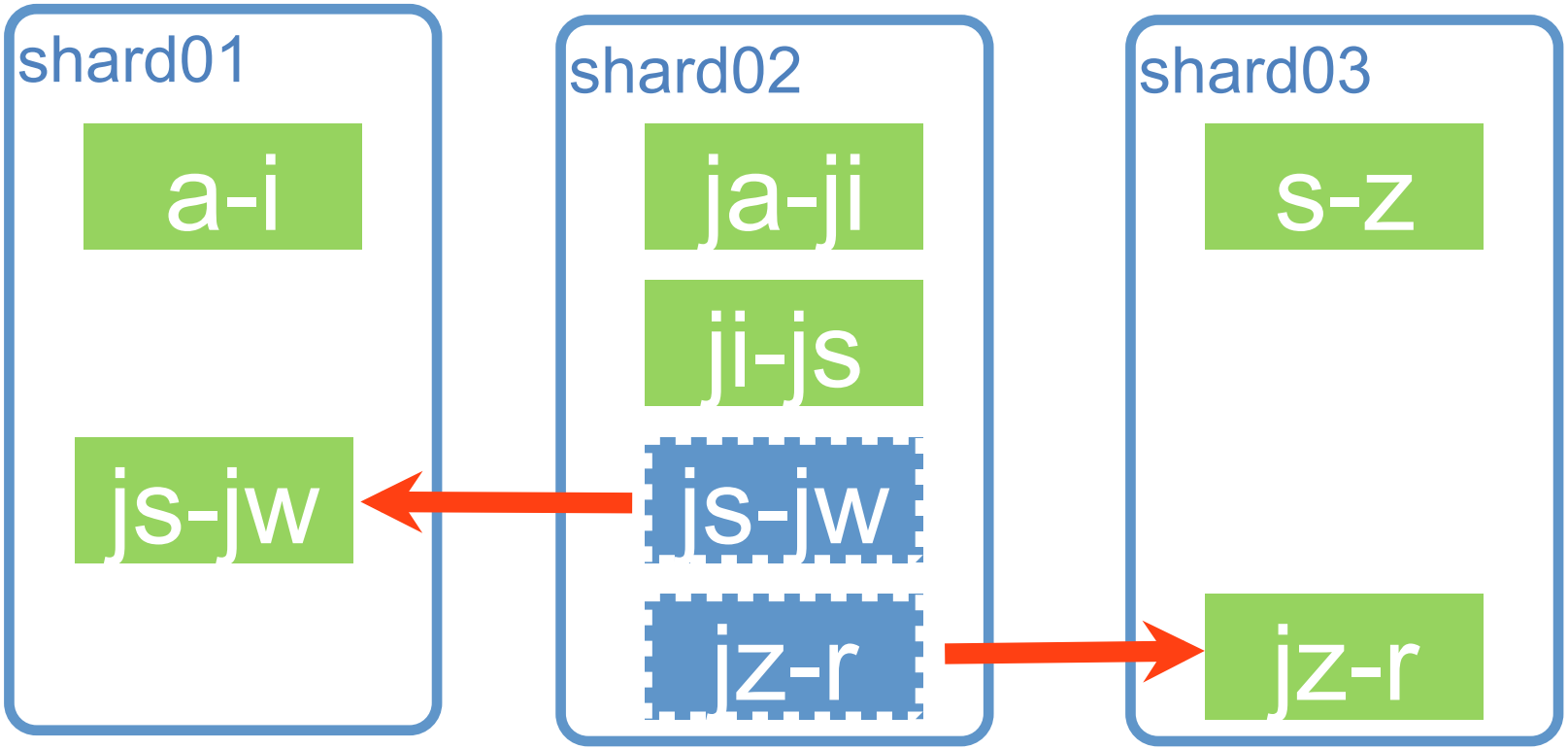




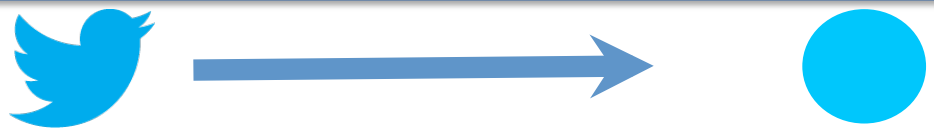


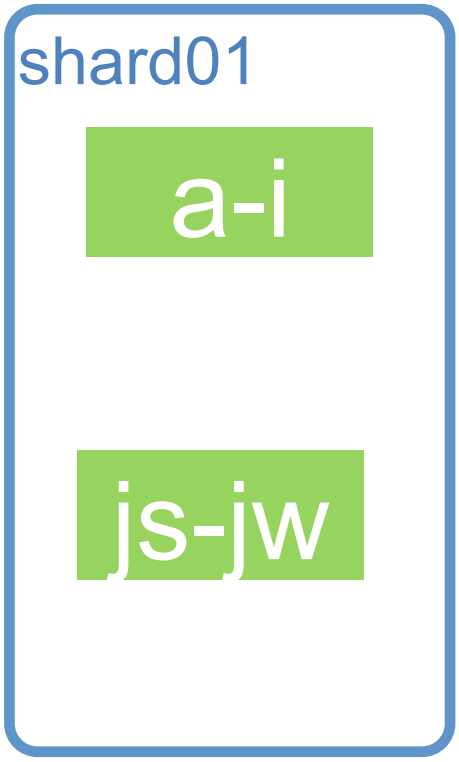
# Sharding - Auto Balancing

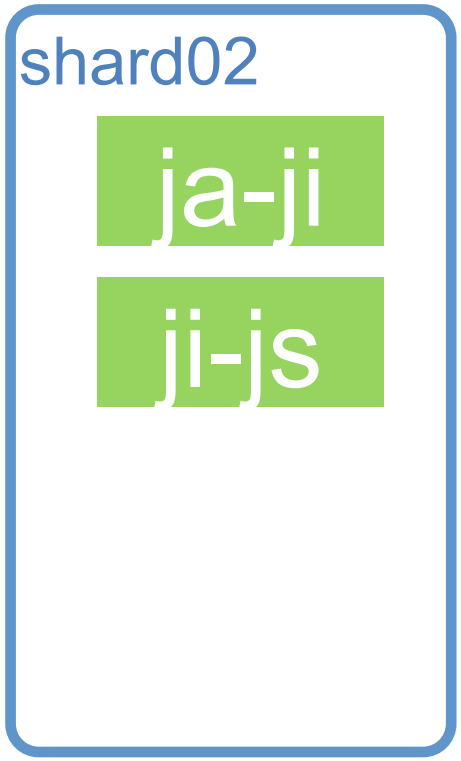


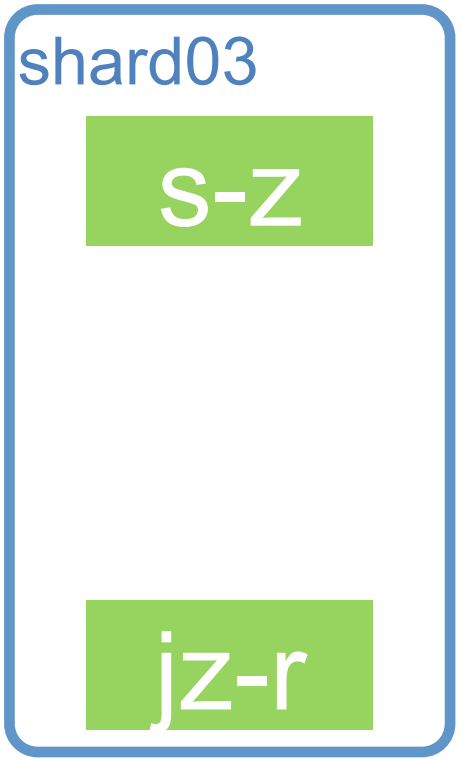


# Sharding - Goal Equilibrium

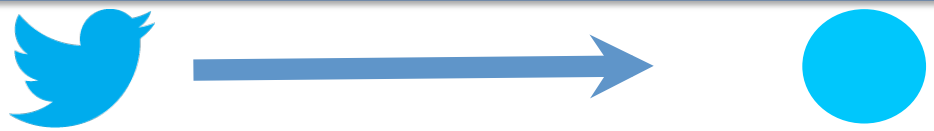




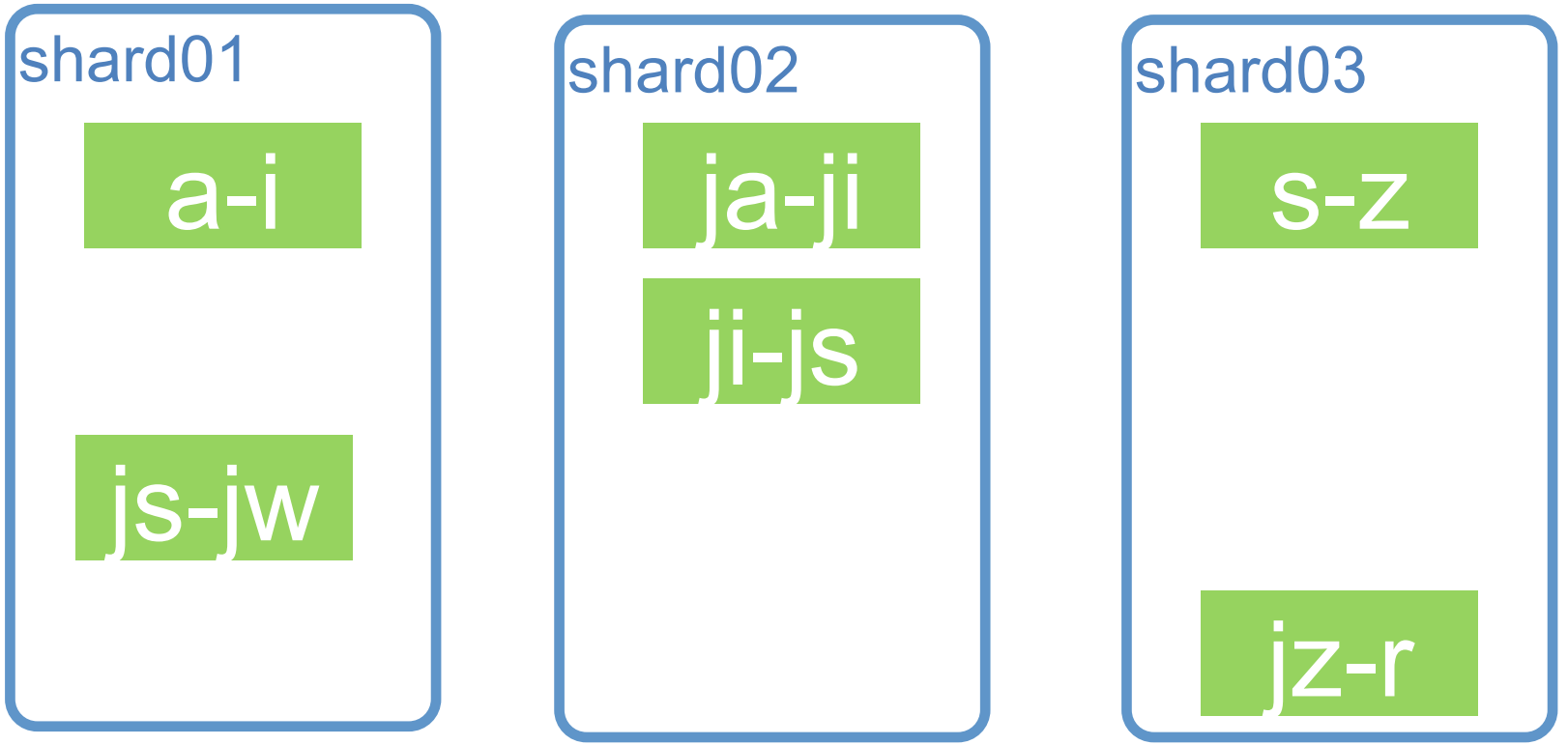



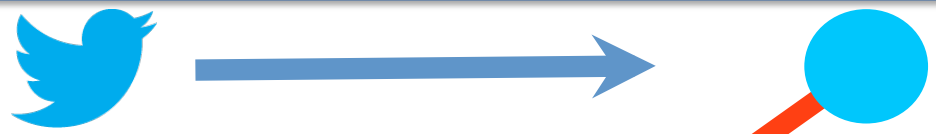
# Sharding - Find by Key



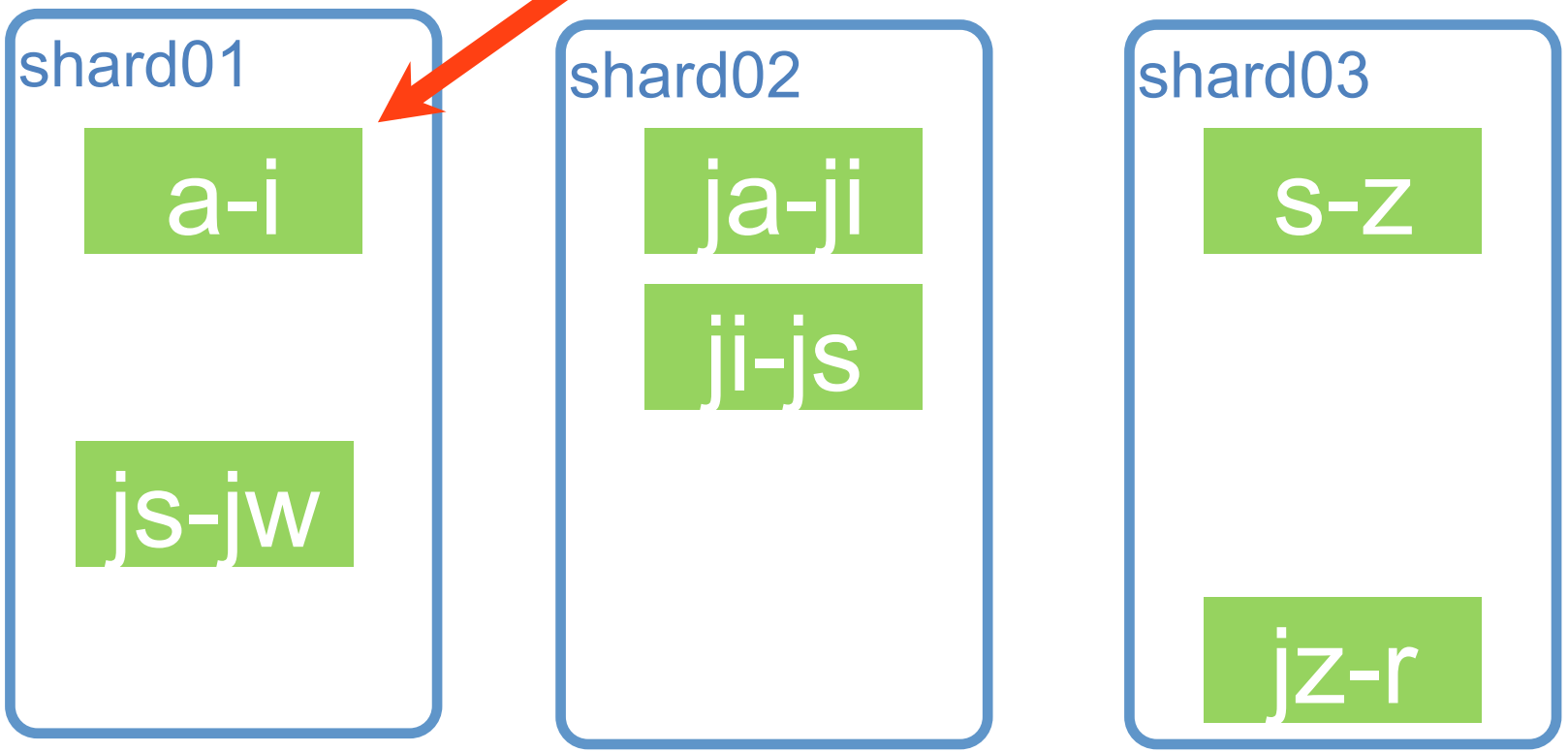
```
find({_id: "alvin"})
```



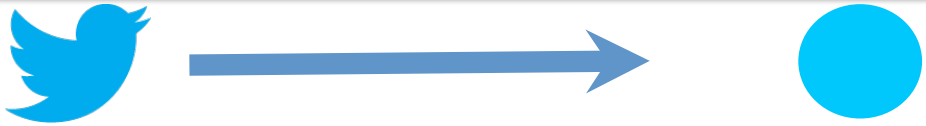
# Sharding - Find by Key



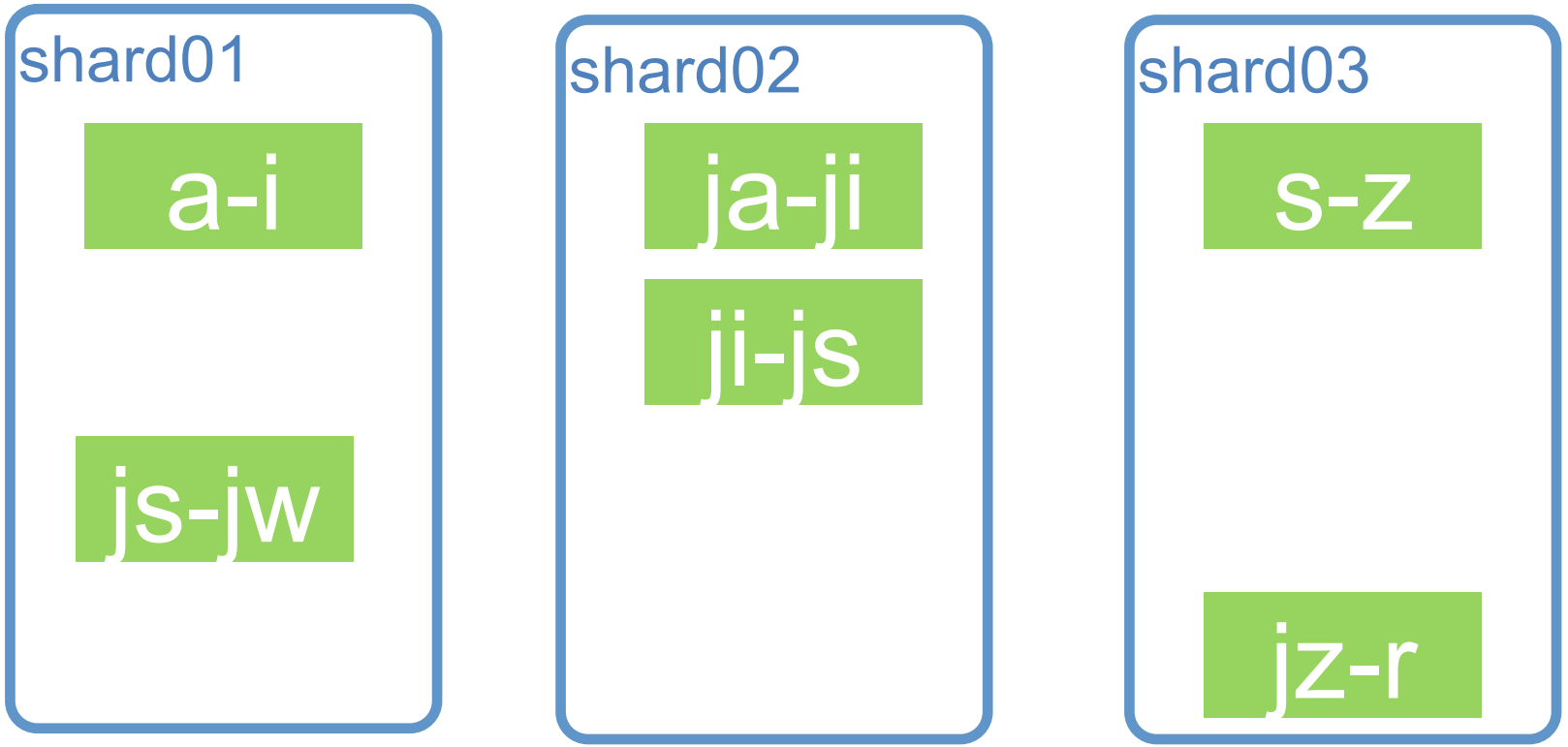
```
find({_id: "alvin"})
```



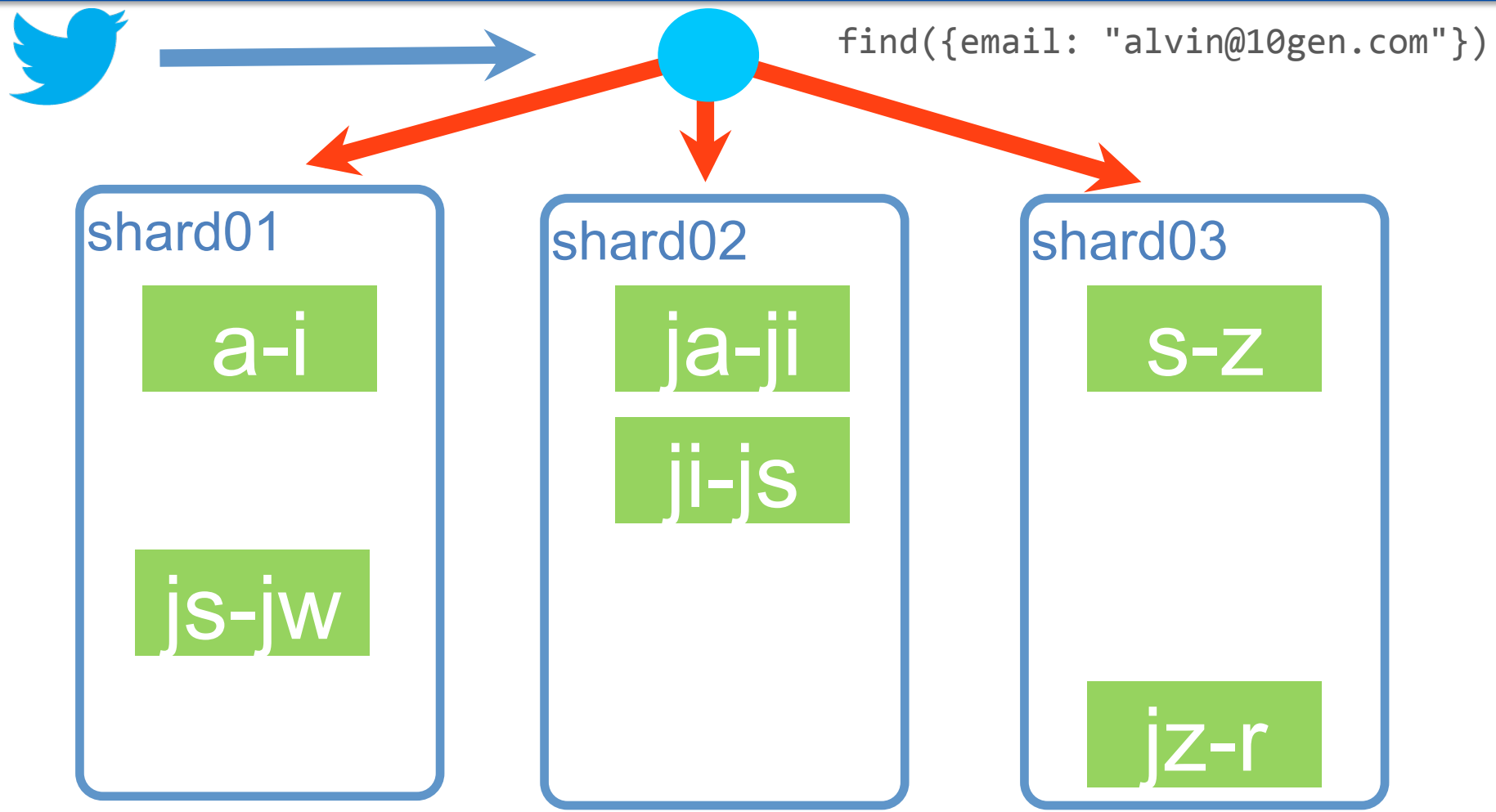
# Sharding - Find by Attribute



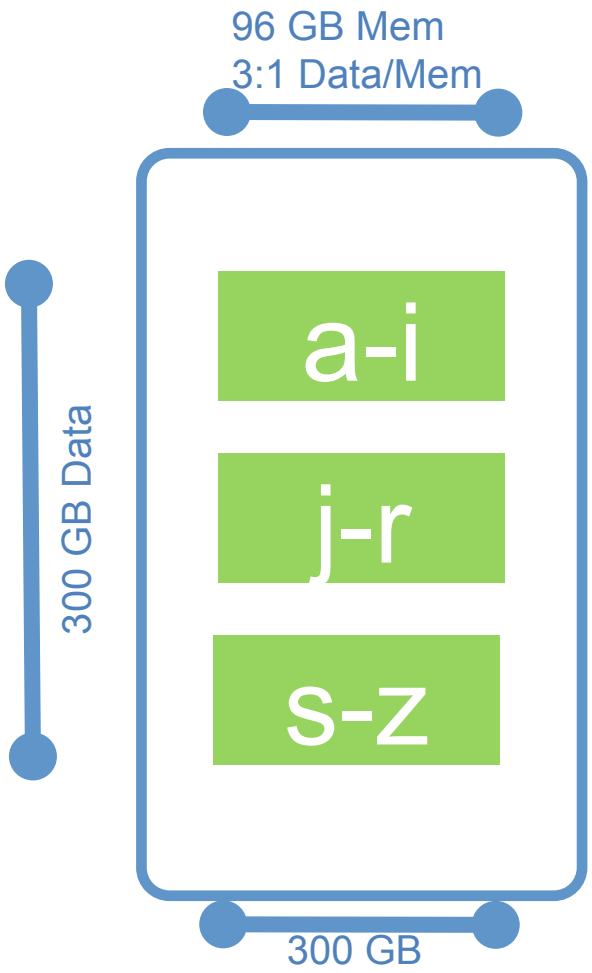
```
find({email: "alvin@10gen.com"})
```



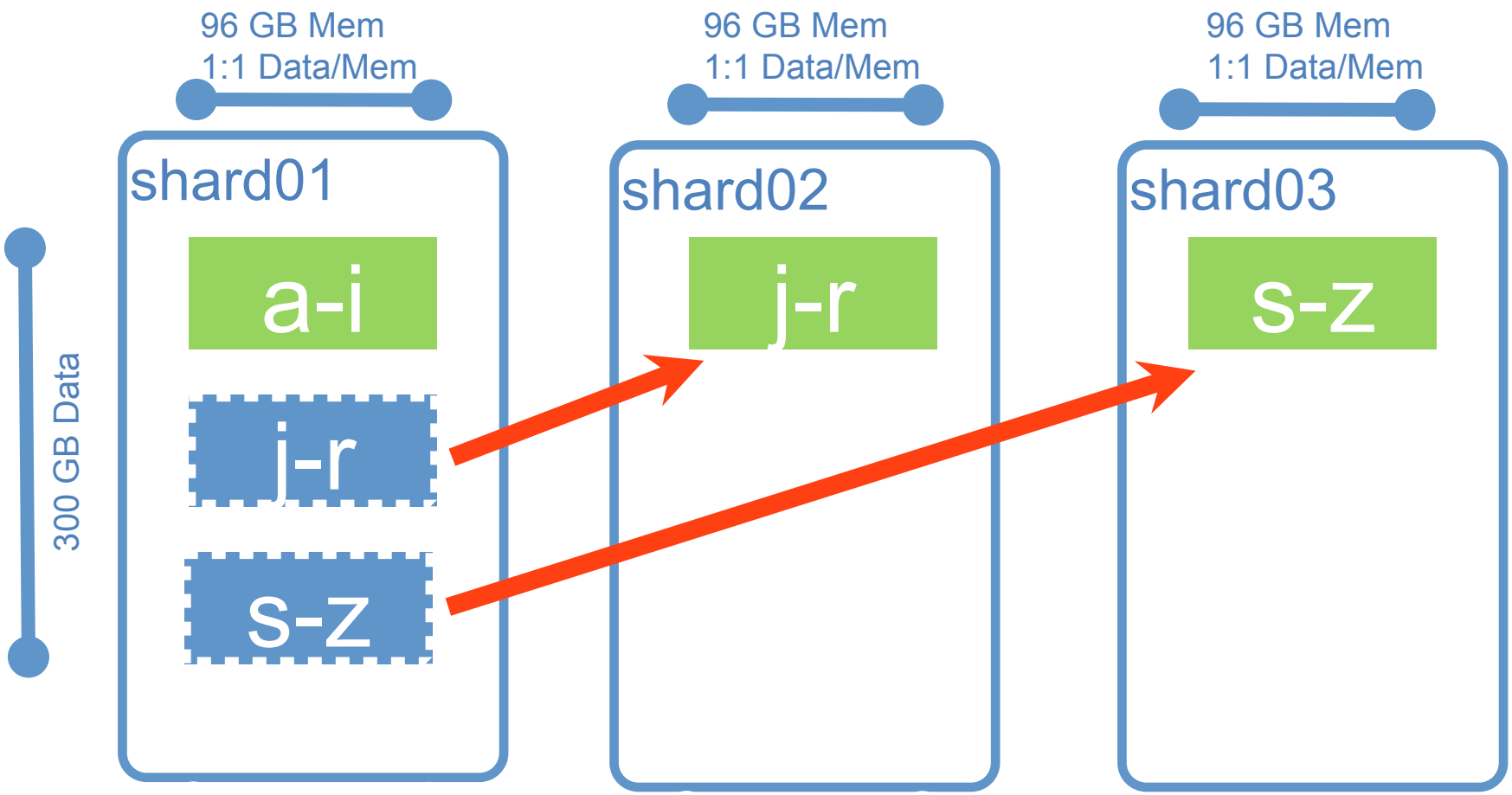
# Sharding - Find by Attribute



# Sharding - Caching



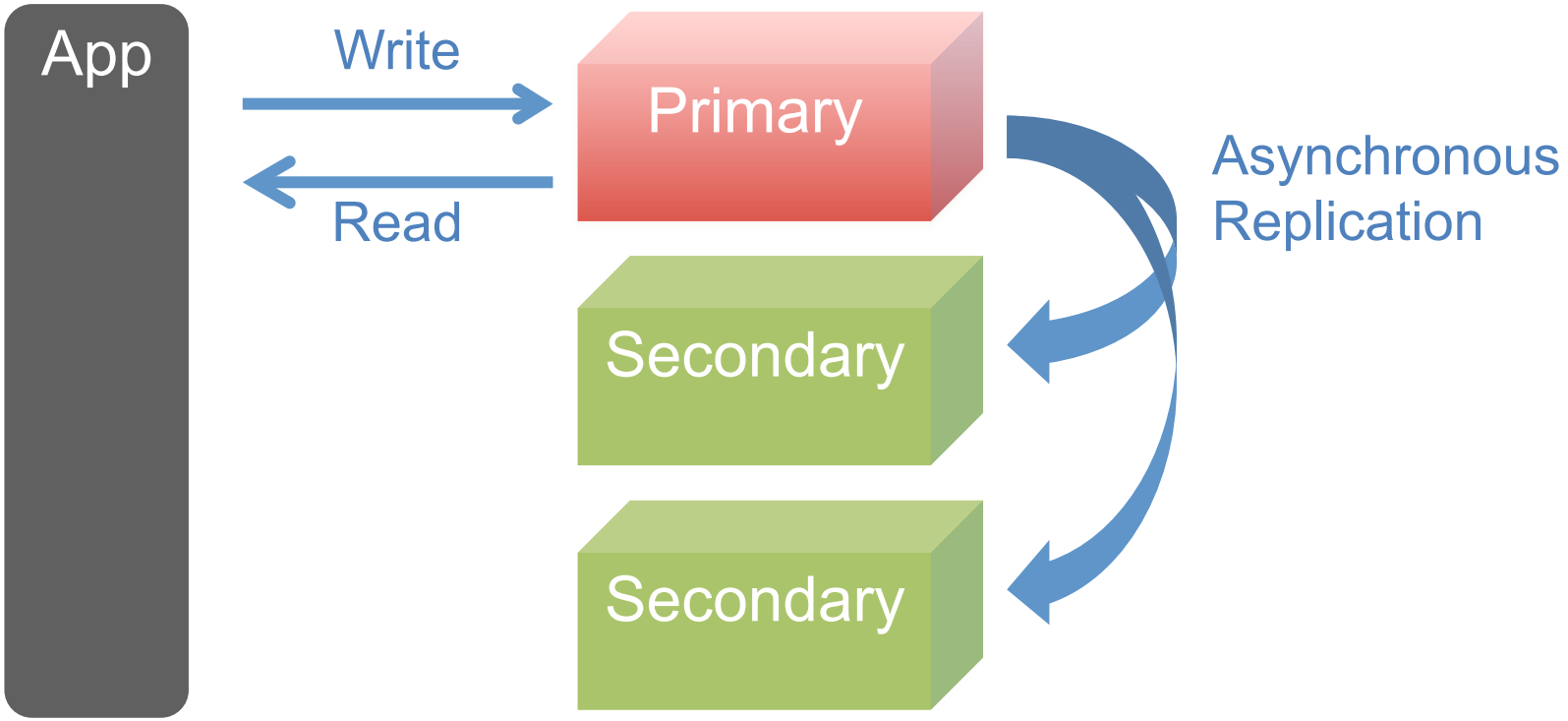
# Aggregate Horizontal Resources



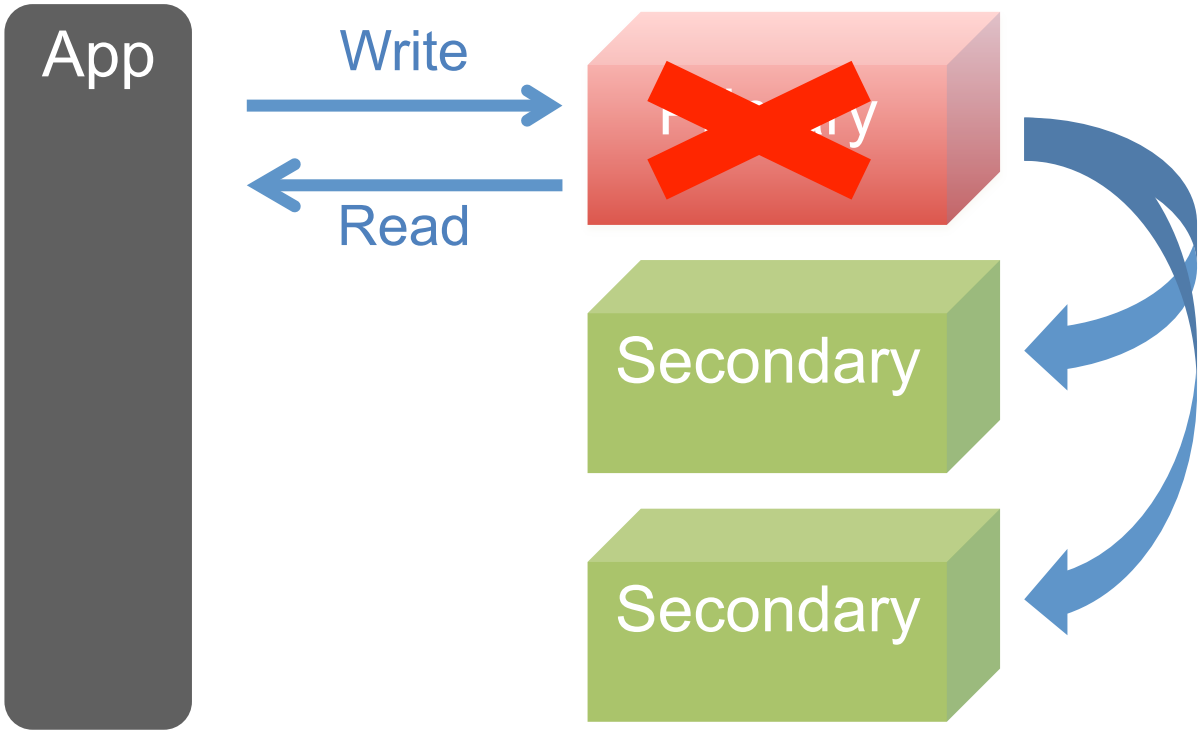
# Sharding

- Partitions data across many nodes
  - Scales Read & Writes
- What happens if a node fails?
  - Data in that partition is lost
- Must have copies of partition across
  - Nodes
  - Data Centers
  - Geographic regions

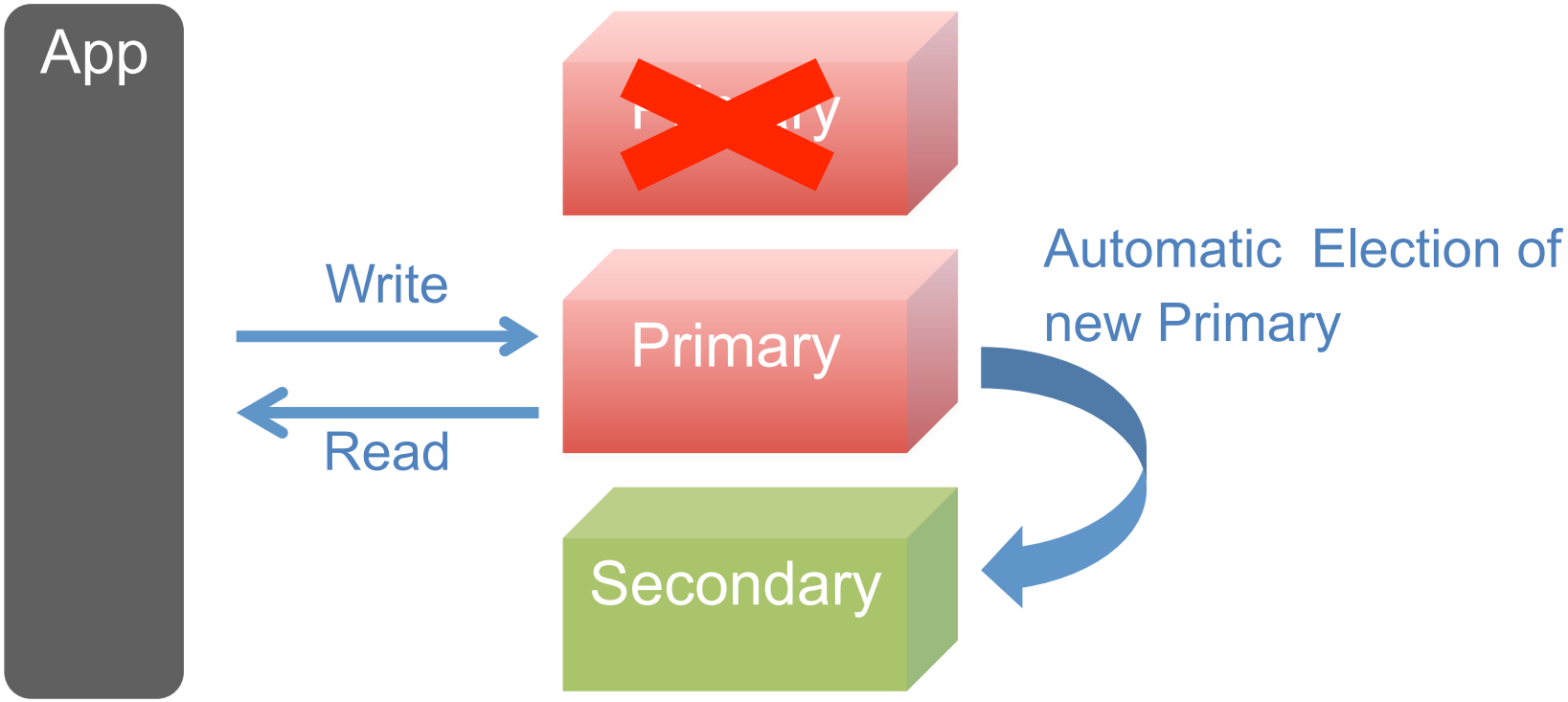
# Replica Sets



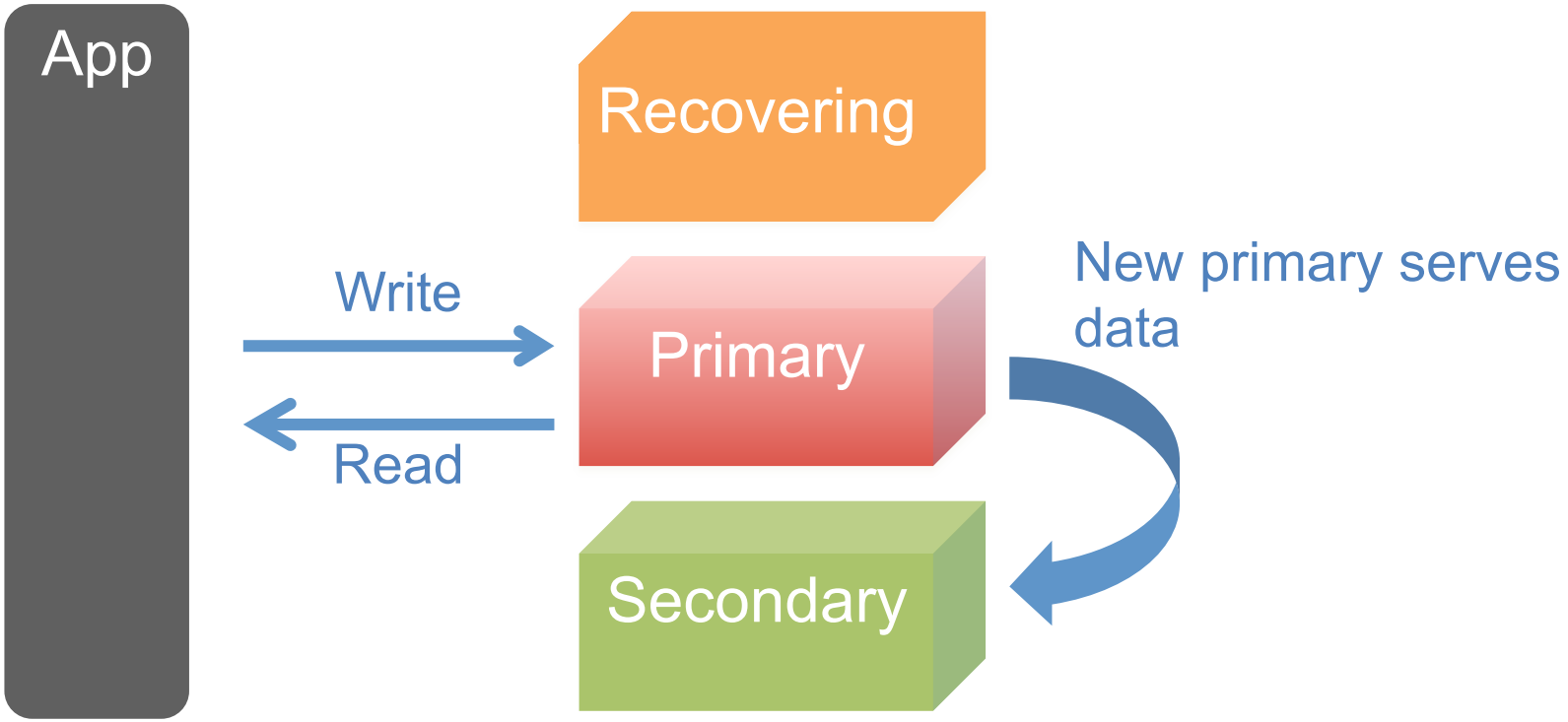
# Replica Sets



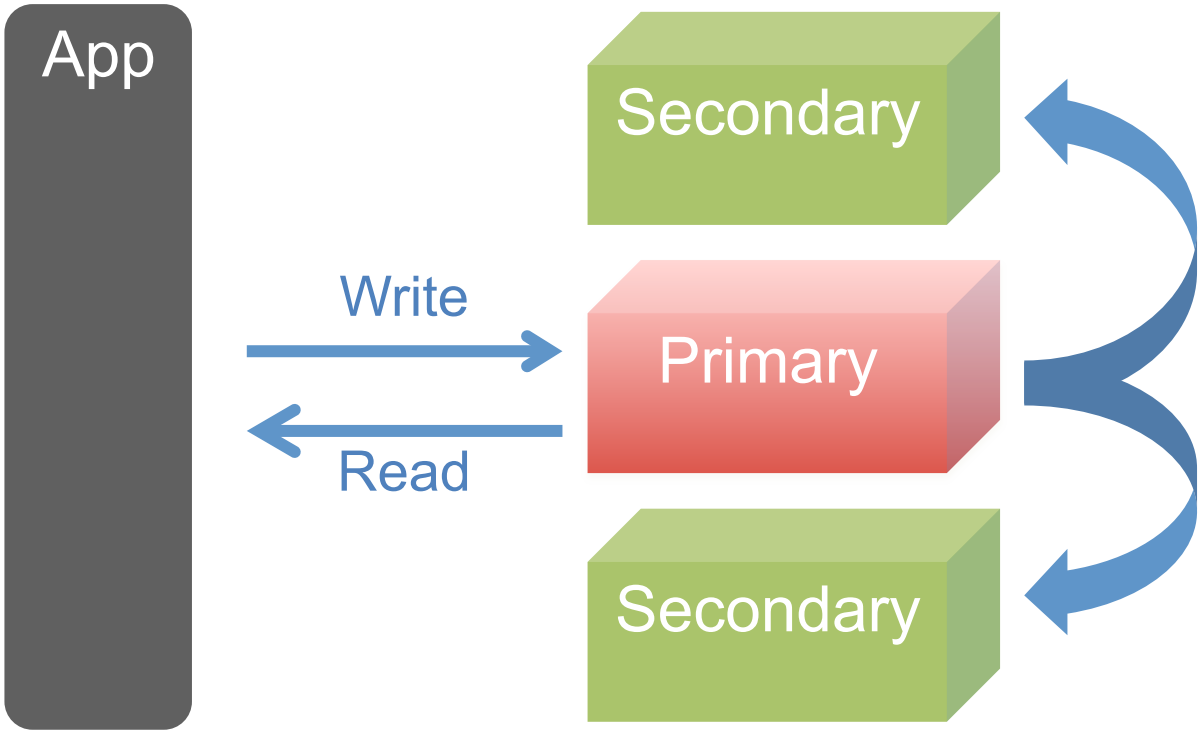
# Replica Sets



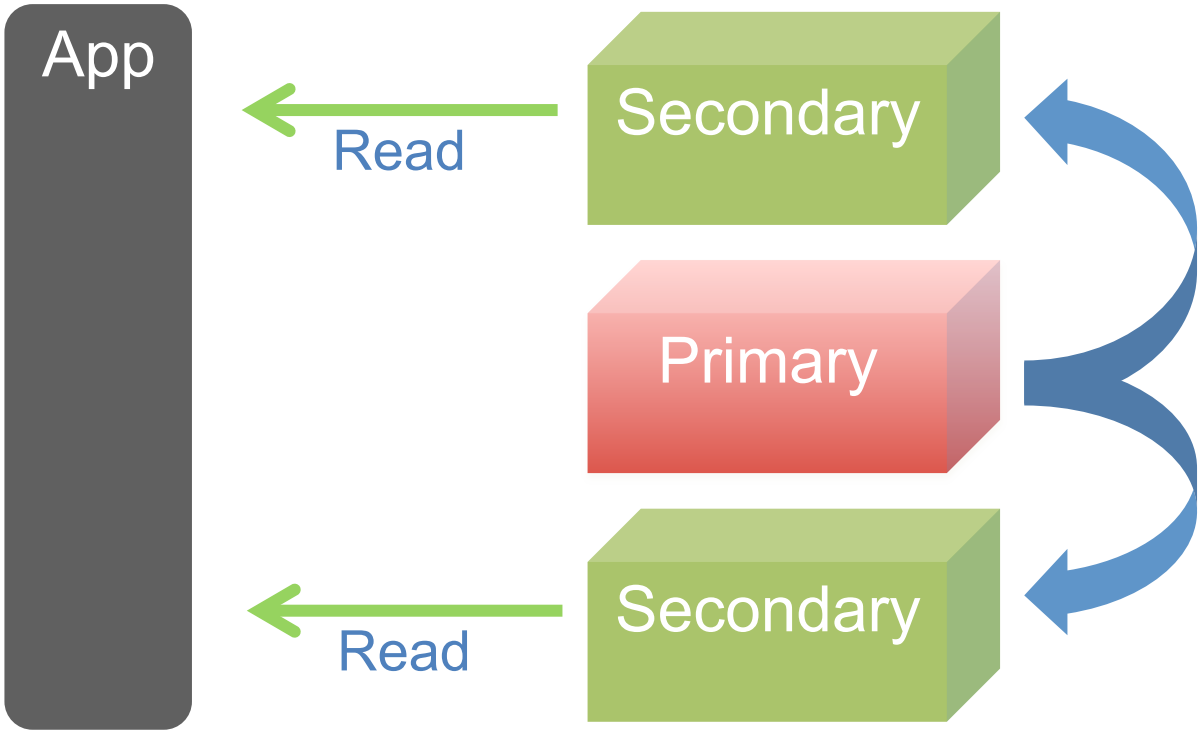
# Replica Sets



# Replica Sets



# Scale Eventually Consistent Reads



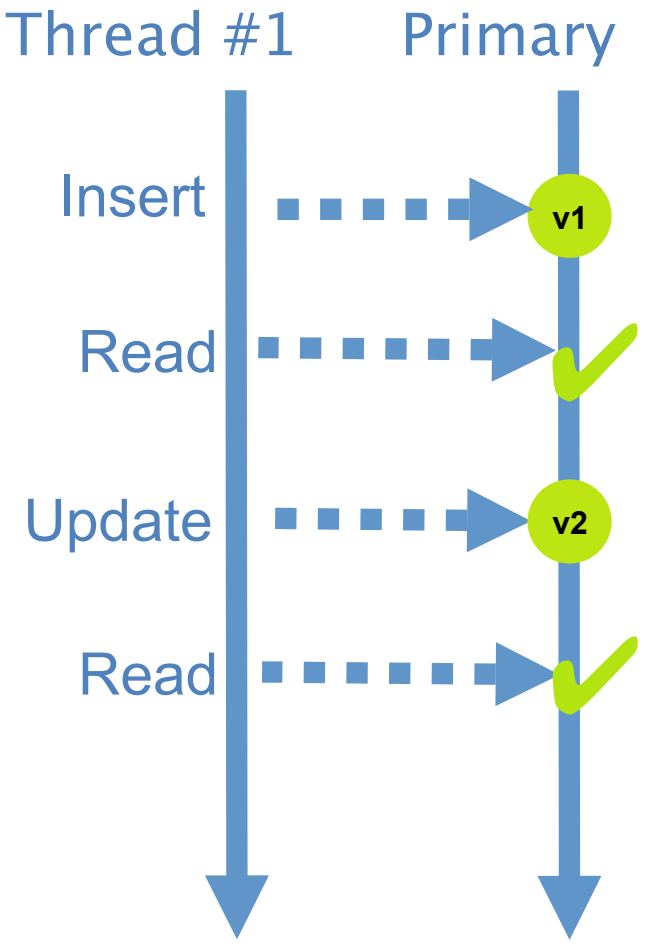
# Eventual Consistency
# Using Replicas for Read Scaling

- Read Preferences
  - PRIMARY, PRIMARY PREFERRED
  - SECONDARY, SECONDARY PREFERRED
  - NEAREST

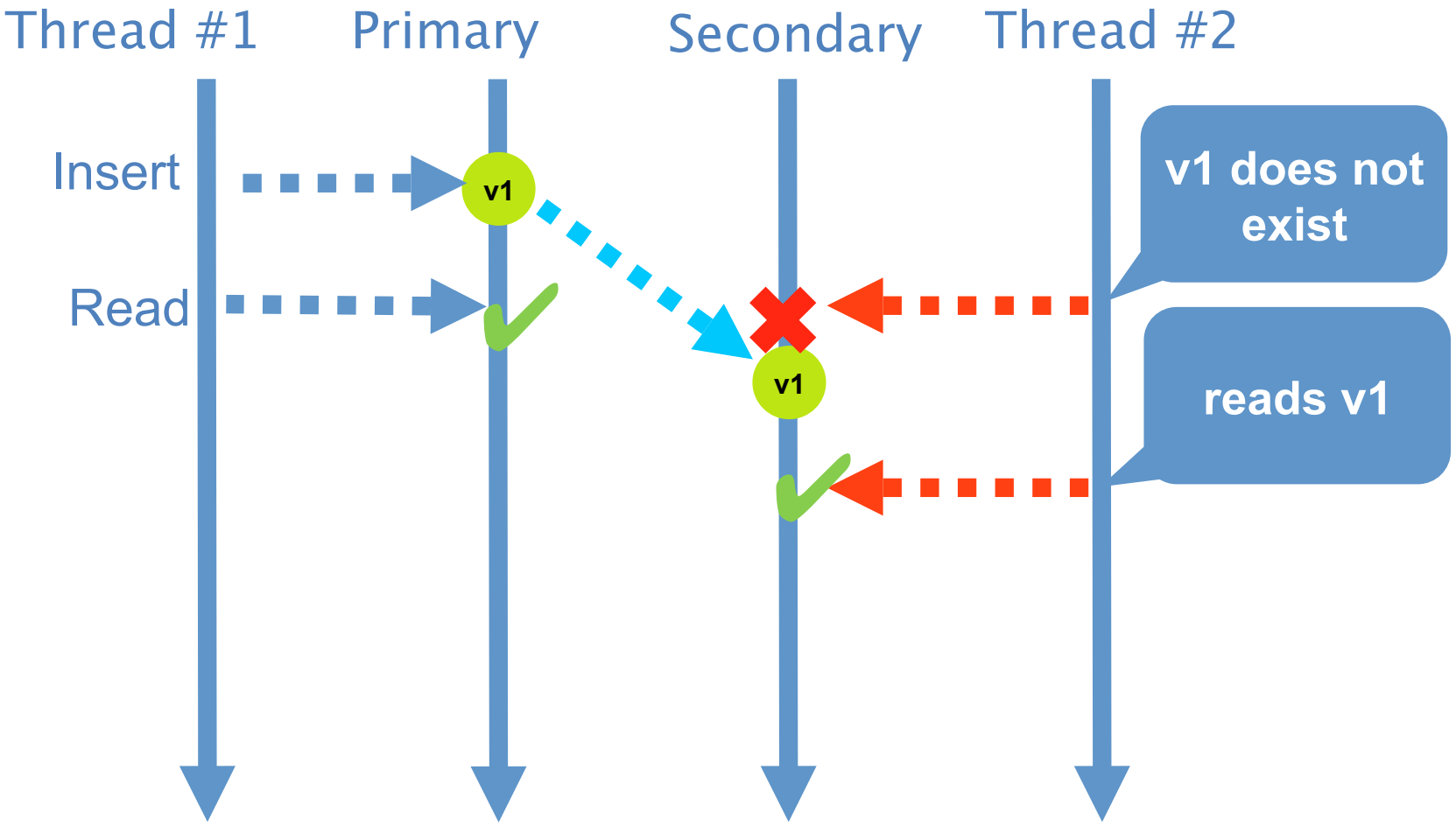
Java example

```
ReadPreference pref = ReadPreference.primaryPreferred();
DBCursor cur = new DBCursor(collection, query,
                            null, pref);
```

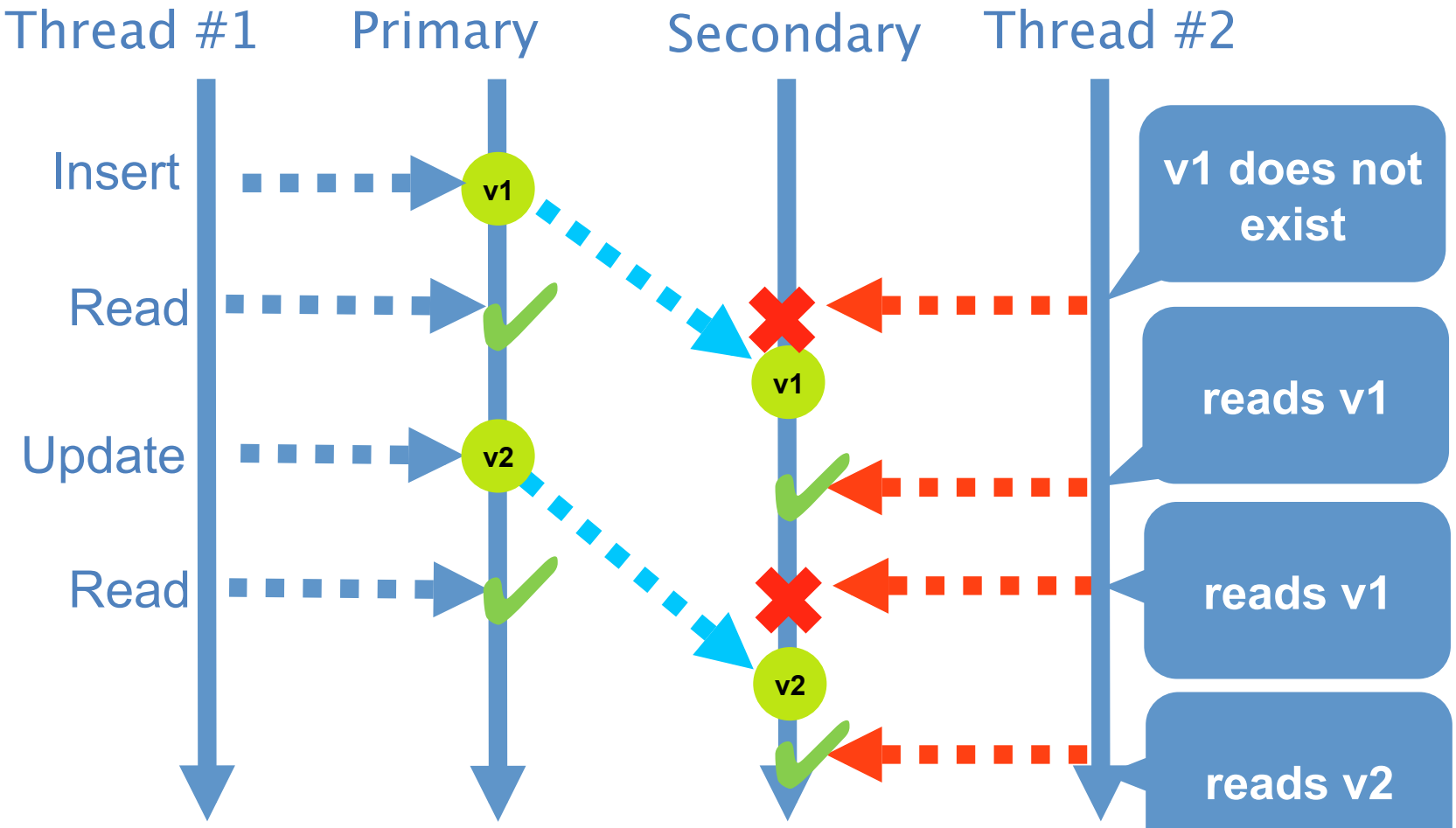
# Immediate Consistency



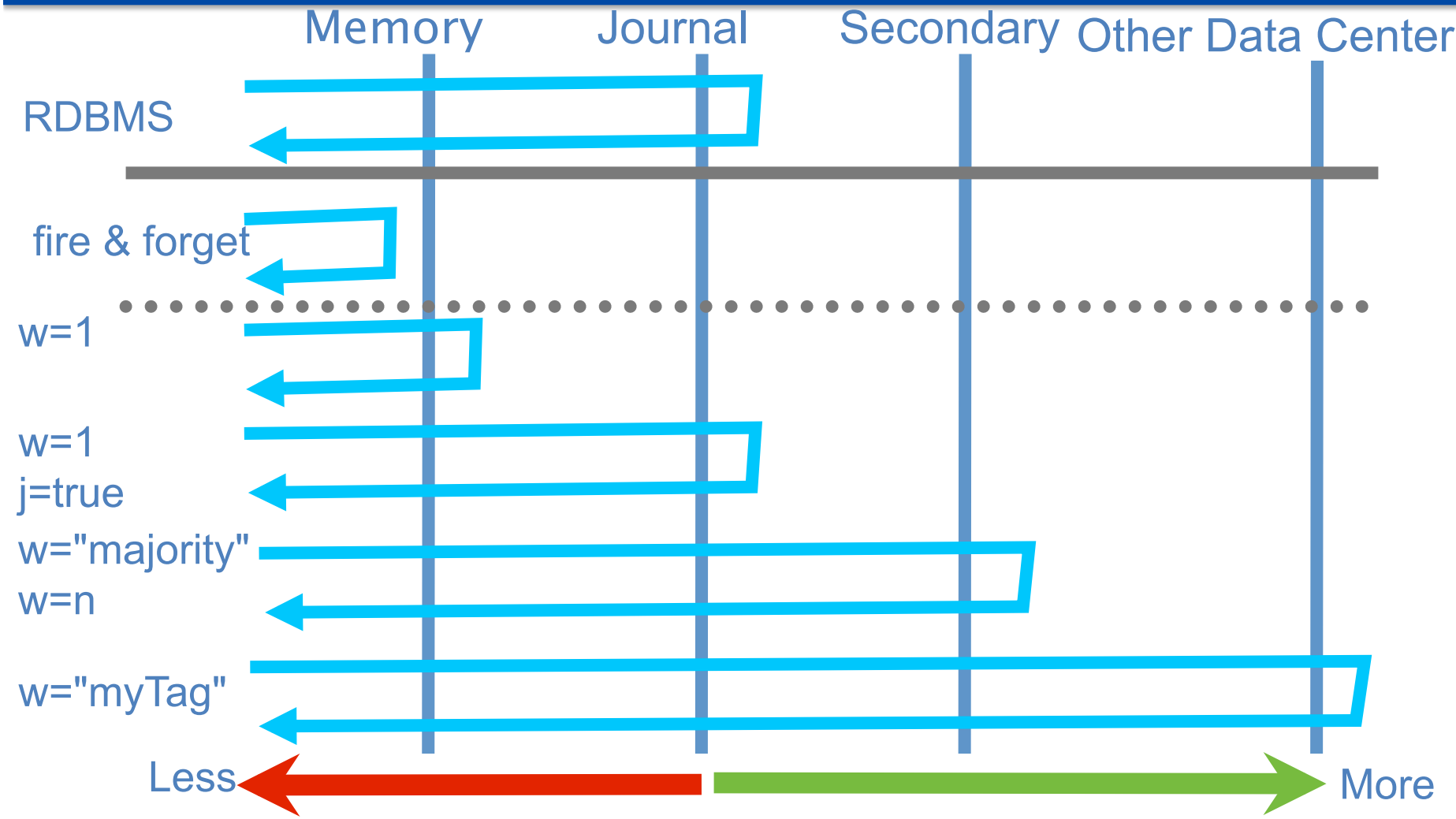
# Eventual Consistency



# Eventual Consistency



# Tunable Data Durability



# Other MongoDB features

- Capped Collections
  - Limit data by size, acts as a circular buffer / FIFO
  - Use cases: Audit, history, logs
- Time To Live (TTL) collections
  - Expire data based on timestamp
  - Use cases: Archiving, purging, sessions
- Text Search
  - Search by word, phrase, stemming, stop words
  - Use cases: Consistent text search

# How does MongoDB shape up?

✓ Build a database for scaleout
  – Run on clusters of 100s of commodity machines

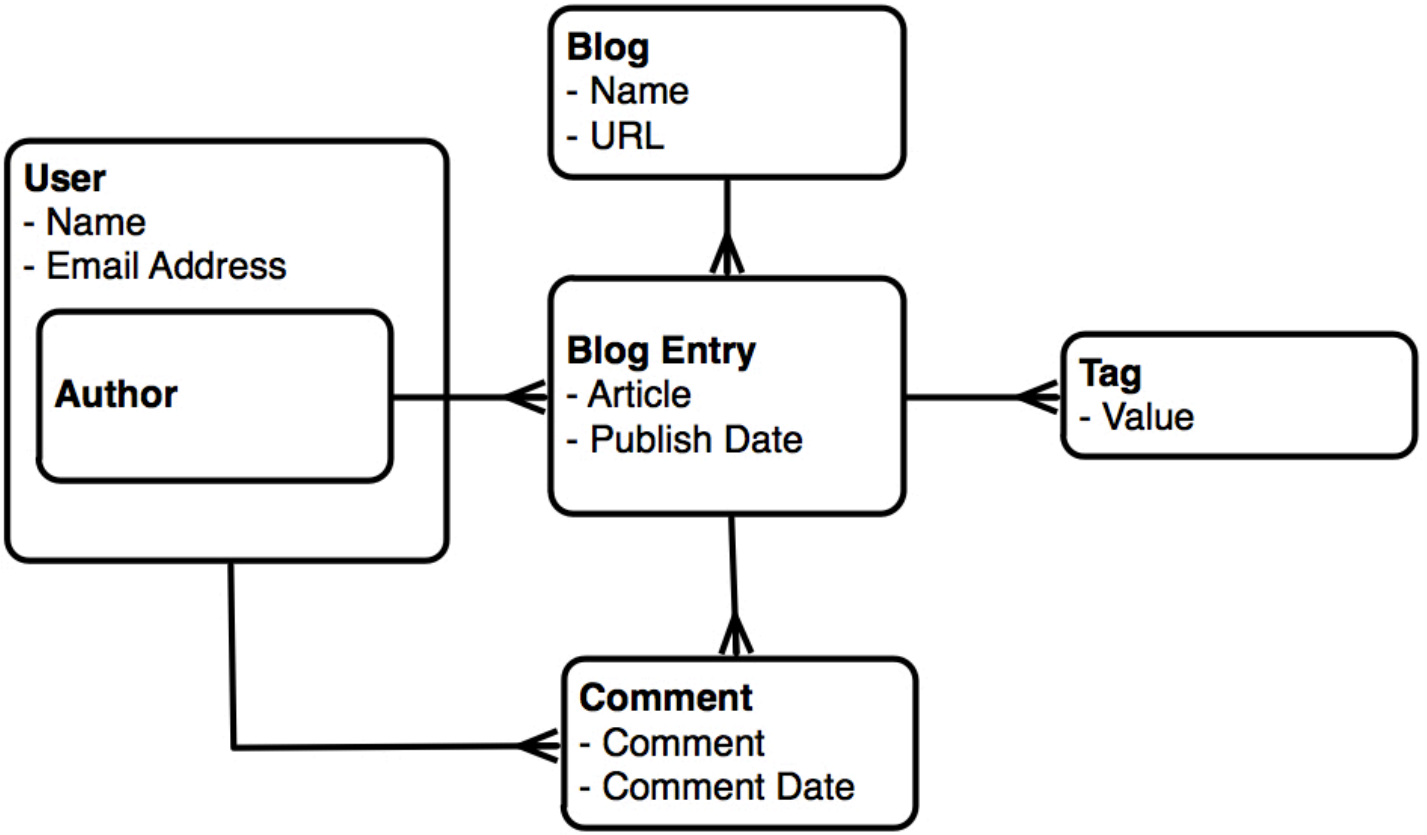
- … that enables agile development

- … and is usable for a broad variety of applications

# Data Model

- Why JSON?
  - Simple, well understood encapsulation of data
  - Maps simply to objects in your OO language
  - Linking & Embedding to describe relationships

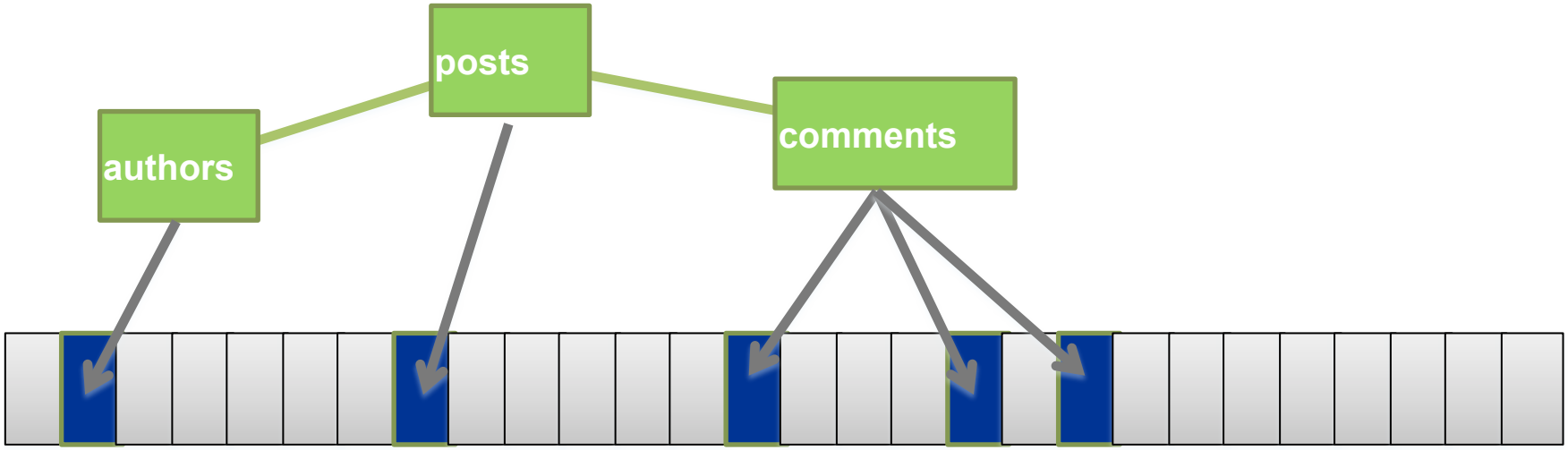
# Why Mess with the Data Model?



# Mapping Objects to RDBMS

```
select *
from posts p,
     authors a,
     comments c
where p.author_id = a.id
  and p.id = c.post_id
  and p.id = 123
```

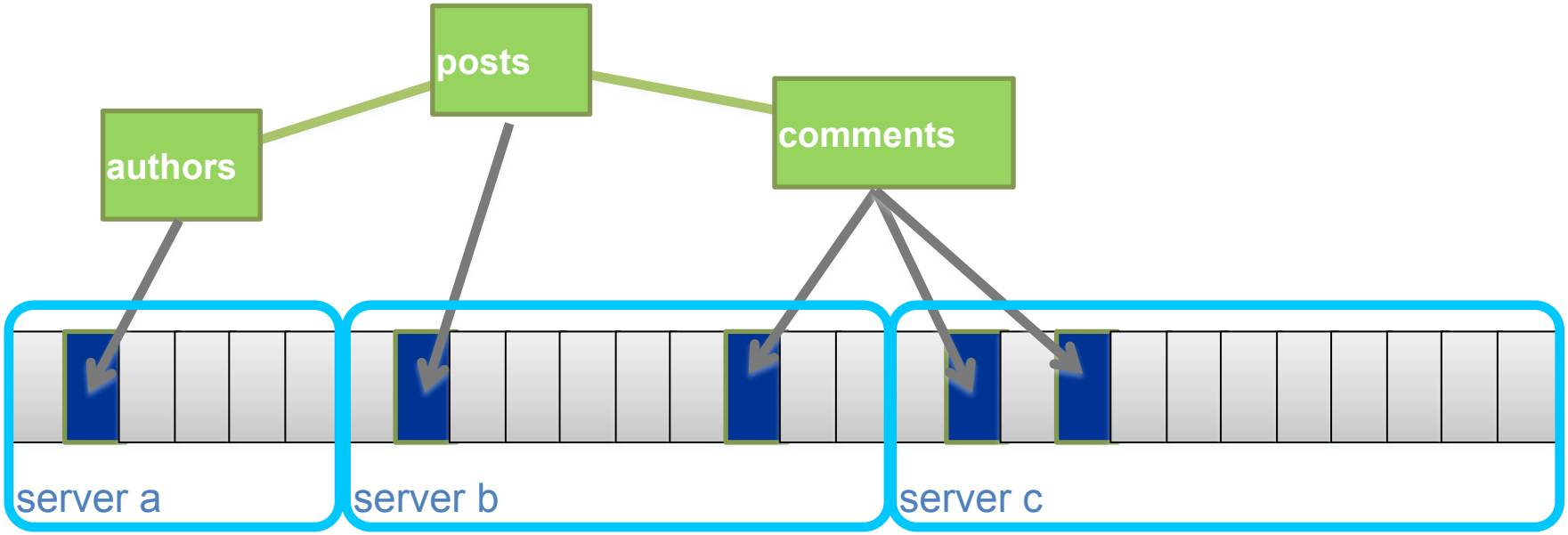
```
start transaction

insert into comments (...)

update posts
  set comment_count = comment_count + 1
  where post_id = 123

commit
```



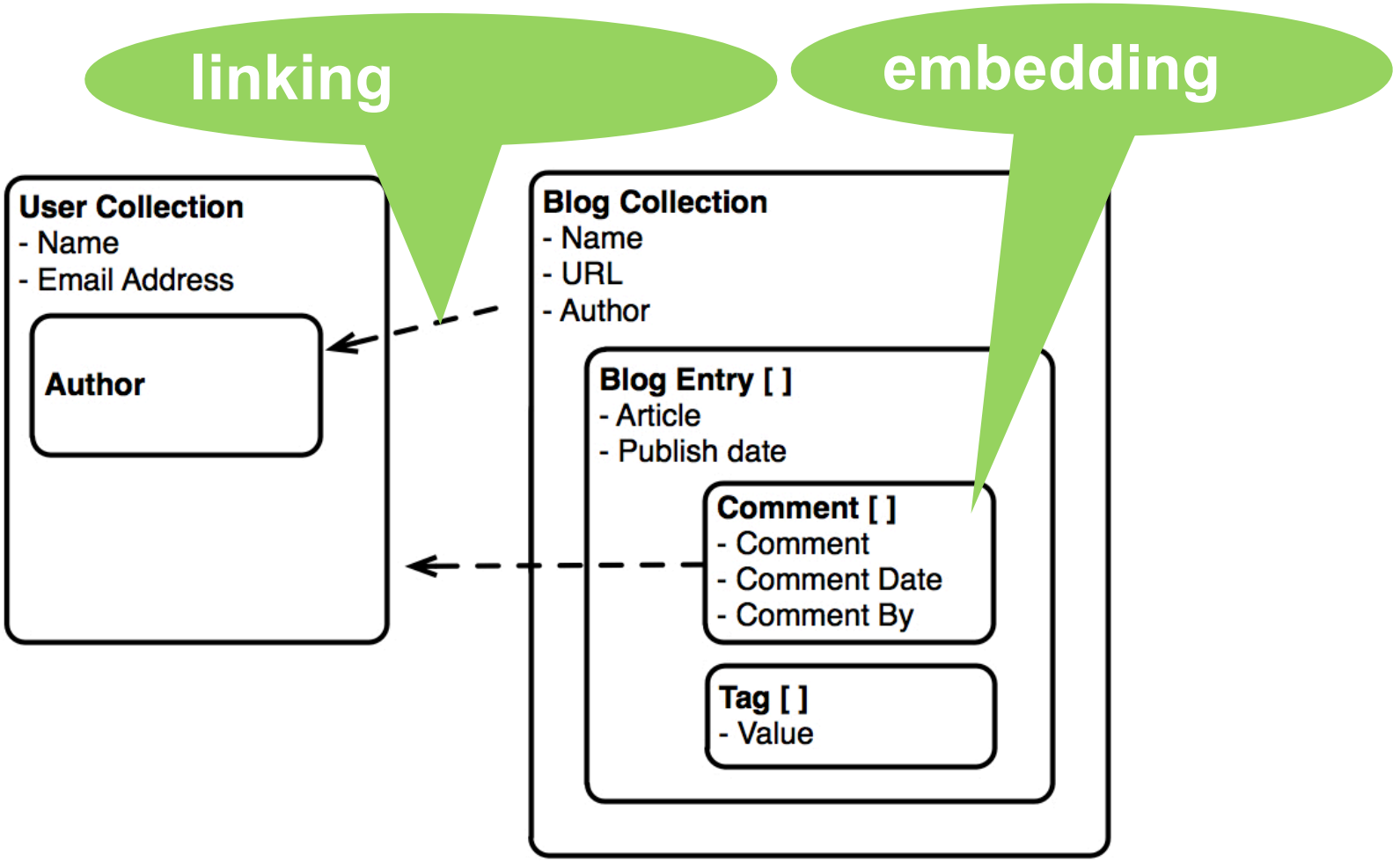
# Mapping Objects to Distributed RDBMS

```
select *
from posts p,
     authors a,
     comments c
where p.author_id = a.id
  and p.id = c.post_id
  and p.id = 123
```

```
start transaction

insert into comments (...)

update posts
  set comment_count = comment_count + 1
  where post_id = 123

commit
```



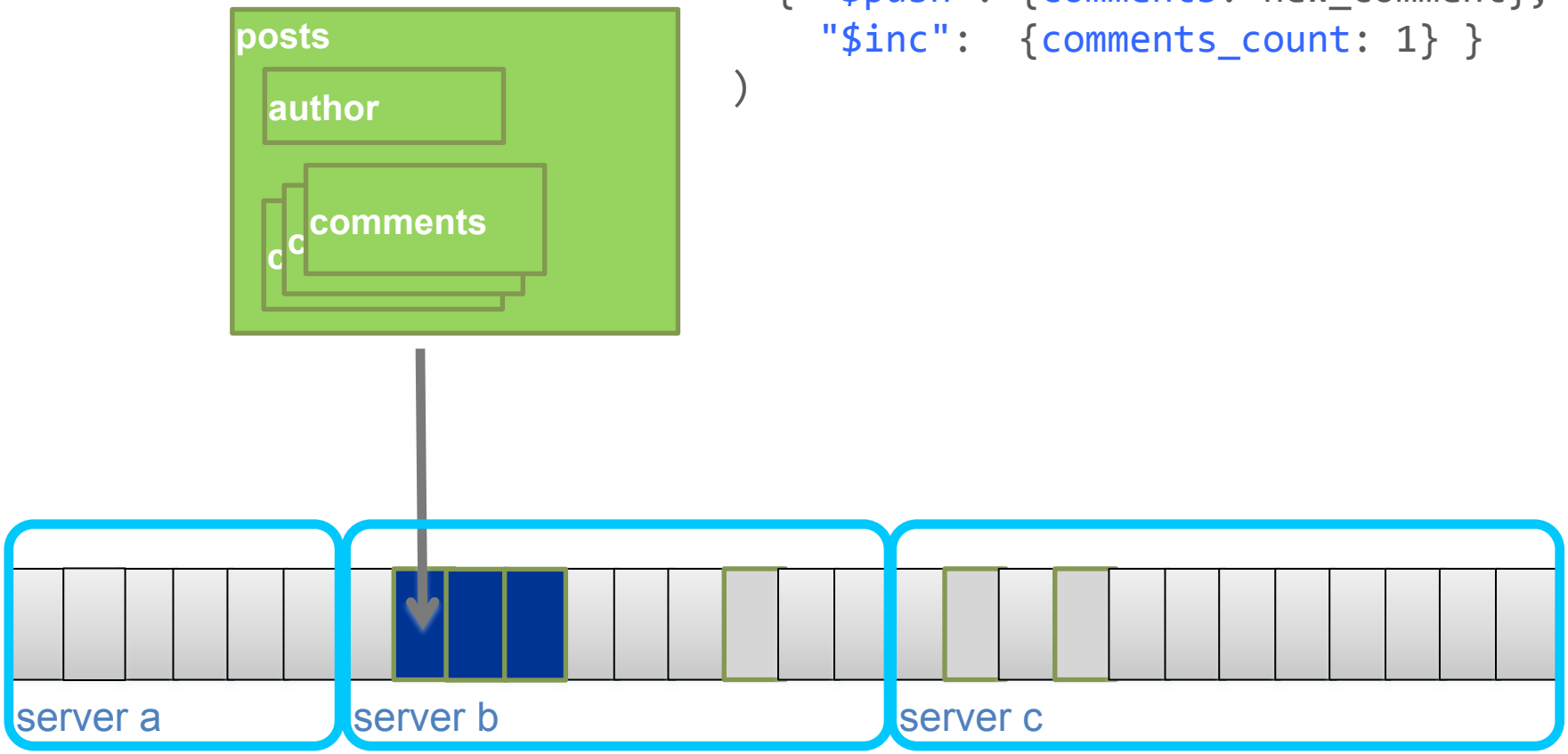
# Same Schema in MongoDB



# Mapping Object with MongoDB

```
db.posts.find({_id: 123})
```

```
db.posts.update(
  {_id: 123},
  { "$push": {comments: new_comment},
    "$inc":  {comments_count: 1} }
)
```



# Schemas in MongoDB

- Design documents that simply map to your application

```
post = {author: "Hergé",
        date: new Date(),
        text: "Destination Moon",
        tags: ["comic", "adventure"]}

> db.posts.save(post)
```

# Examples

```
// Find the object
> db.blogs.find( { text: "Destination Moon" } )

// Find posts with tags
> db.blogs.find( { tags: { $exists: true } } )

// Regular expressions: posts where author starts with h
> db.blogs.find( { author: /^h/i } )

// Counting: number of posts written by Hergé
> db.blogs.find( { author: "Hergé" } ).count()
```

# Data Manipulation

- Conditional Query Operators
  - **Scalar**: $ne, $mod, $exists, $type, $lt, $lte, $gt, $gte, $ne
  - **Vector**: $in, $nin, $all, $size
- Atomic Update Operators
  - **Scalar**: $inc, $set, $unset
  - **Vector**: $push, $pop, $pull, $pushAll, $pullAll, $addToSet

# Extending the schema

```
> db.blogs.update(
            { text: "Destination Moon" },
            { "$push": { comments: new_comment },
              "$inc":  { comments_count: 1 } } )

  { _id: ObjectId("4c4ba5c0672c685e5e8aabf3"),
    text: "Destination Moon",
    comments: [
      {
        author: "Kyle",
        date: ISODate("2011-09-19T09:56:06.298Z"),
        text: "great book"
      }
    ],
    comment_count: 1
  }
```

# How does MongoDB shape up?

✓ Build a database for scaleout
  – Run on clusters of 100s of commodity machines

✓ … that enables agile development

• … and is usable for a broad variety of applications

# Big Data = MongoDB = Solved

## Big Data





## Content Mgmt & Delivery





## Mobile & Social







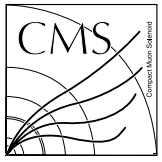
## User Data Management





## Data Hub




10gen | The MongoDB Company



mongoDB

# How does MongoDB shape up?

✓ Build a database for scaleout
  – Run on clusters of 100s of commodity machines

✓ … that enables agile development

✓ … and is usable for a broad variety of applications

# 10gen is the organization behind MongoDB

**190+** employees

**500+** customers

NEA

FLYBRIDGE CAPITAL PARTNERS

intel Capital

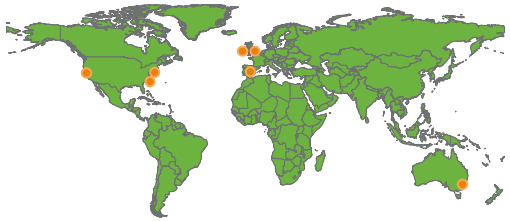
SEQUOIA CAPITAL
THE ENTREPRENEURS BEHIND THE ENTREPRENEURS

union square ventures

redhat

IQT IN-Q-TEL

Over **$81 million** in funding



**Offices** in New York, Palo Alto, Washington DC, London, Dublin, Barcelona and Sydney

10gen | The MongoDB Company



mongoDB

# 10gen Products and Services

## Subscriptions

Professional Support, Subscriber Edition and Commercial License

## Consulting

Expert Resources for All Phases of MongoDB Implementations

## Training

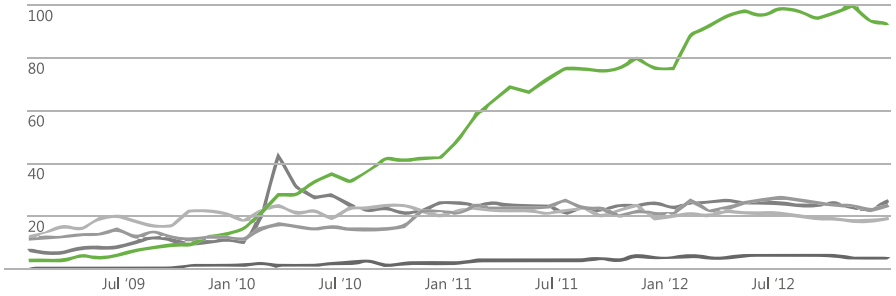
Online and In-Person for Developers and Administrators

## MongoDB Monitoring Service (MMS)

Free, Cloud-Based Service for Monitoring and Alerts

# MongoDB is the Leading NoSQL Database

## Google Search




Google

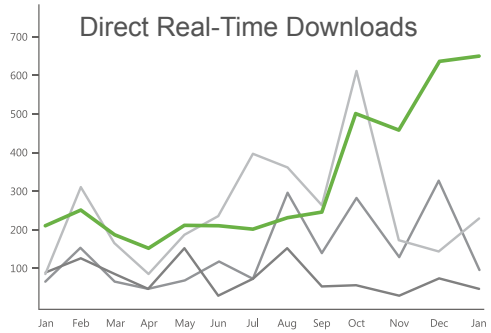
MongoDB
Competitor 1
Competitor 2
Competitor 3
Competitor 4

## LinkedIn Job Skills

451 Research

MongoDB
Competitor 1
Competitor 2
Competitor 3
Competitor 4
Competitor 5
All Others

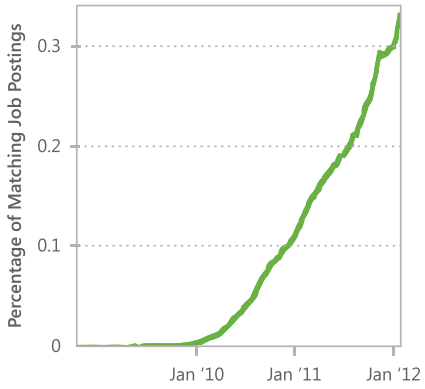
## Jaspersoft Big Data Index




MongoDB
Competitor 1
Competitor 2
Competitor 3

JASPERSOFT

## Indeed.com Trends




indeed
one search. all jobs.

**Top Job Trends**

1. HTML 5
2. **MongoDB**
3. iOS
4. Android
5. Mobile Apps
6. Puppet
7. Hadoop
8. jQuery
9. PaaS
10. Social Media

# The Evolution of MongoDB

**1.8 March '11**

- Journaling
- Sharding and Replica set enhancements
- Spherical geo search

**2.0 Sept '11**

- Index enhancements to improve size and performance
- Authentication with sharded clusters
- Replica Set Enhancements
- Concurrency improvements

**2.2 Aug '12**

- Aggregation Framework
- Multi-Data Center Deployments
- Improved Performance and Concurrency

**2.4 March '13**

- Kerberos/SASL
- Hash Shard Key
- V8
- Intersecting polygons
- Aggregation enhancements
- Text Search

10gen | The MongoDB Company



mongoDB

download at mongodb.org

10gen | the MongoDB company

Drop by on the 5th floor and meet an Engineer
and
Get a discount code for MongoDB London April 9th



Facebook | Twitter | LinkedIn

http://bit.ly/mongofb | @mongodb | http://linkd.in/joinmongo