mongoDB

Intro to MongoDB & the JVM

Bringing NoSQL and Java Together

10gen, Inc.

JFokus 2012

10gen mongoDB

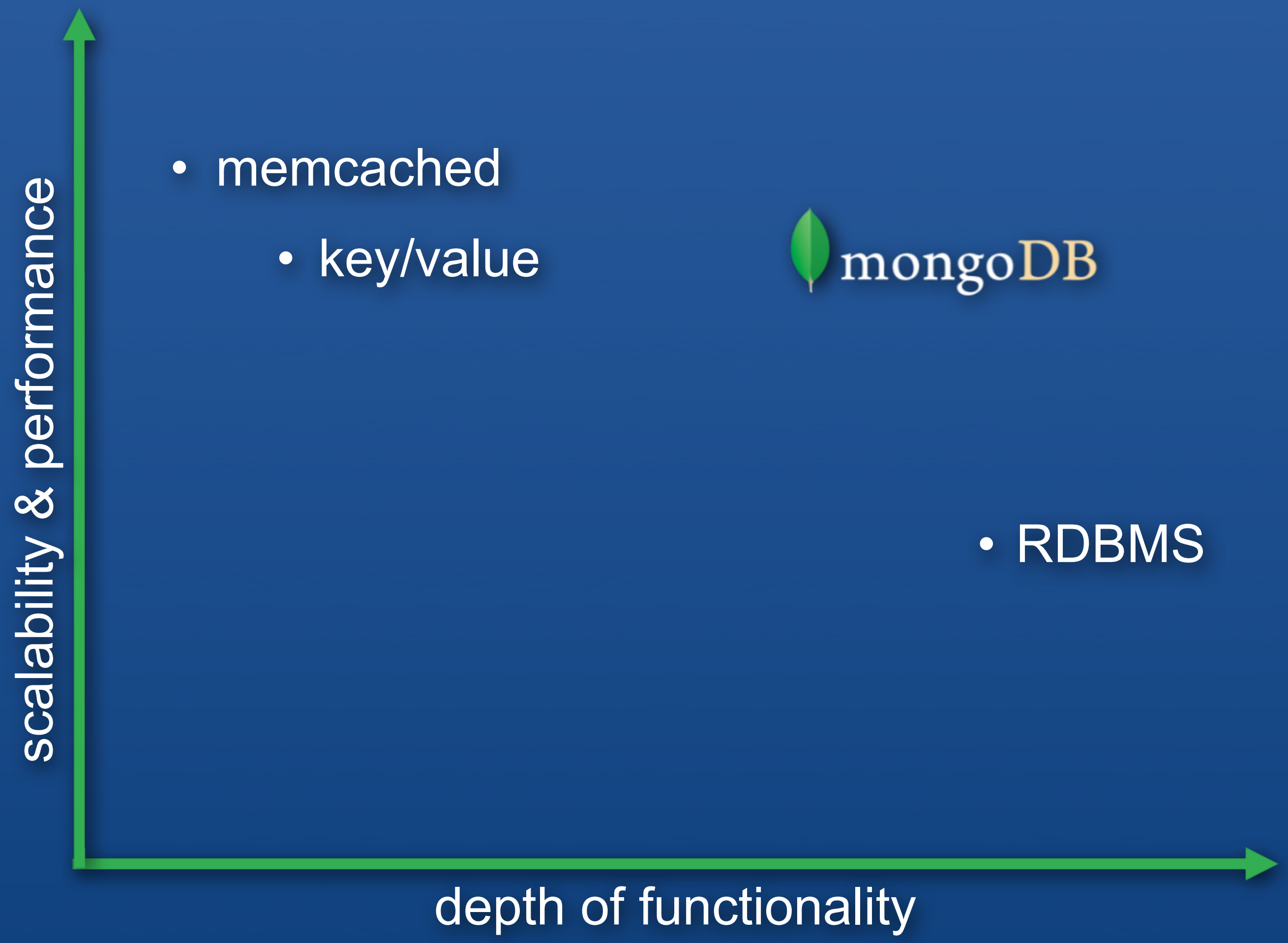
@dmroberts

daniel.roberts@10gen.com

Solution Architect
Based in London
http://www.10gen.com/

MongoDB is a....

- Document Oriented
- High Performance
- Highly Available
- Horizontally Scalable
- Operational Datastore



- memcached
- key/value

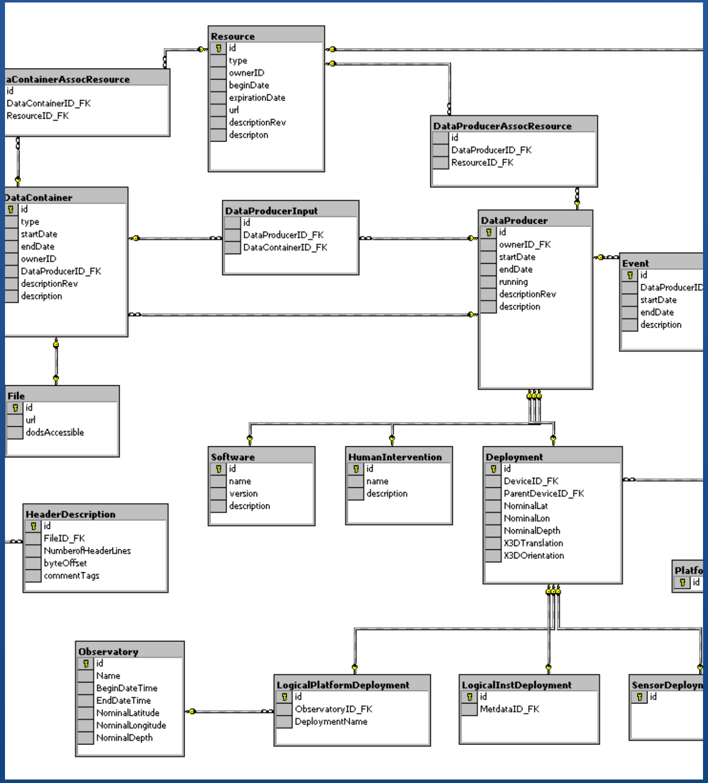
mongoDB

- RDBMS

scalability & performance

depth of functionality

Tables to Documents



```
{
  title: ‘MongoDB’,
  contributors: [
    { name: ‘Eliot Horowitz’,
      email: ‘eliot@10gen.com’ },
    { name: ‘Dwight Merriman’,
      email: ‘dwight@10gen.com’ }
  ],
  model: {
    relational: false,
    awesome: true
  }
}
```

JSON Documents

```
> var p = {    author : “roger”,
                date :  new Date(),
                text : “Spirited Away”,
                tags: [“Tezuka”, “Manga”]
             }

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

Querying with JSON Documents

```
>db.posts.find( { author : ”roger” } )

   { _id : ObjectId("4c4ba5c0672c685e5e8aabf3"),
     author : "roger",
     date : "Tue Feb 14 2012 19:47:11 GMT-0700 (PDT)",
     text : "Spirited Away",
     tags : [ "Tezuka", "Manga" ] }
```

Notes:

- _id is unique, but can be anything you’d like

Indexes

Create index on any Field in Document

```
//   1 means ascending, -1 means descending

> db.posts.ensureIndex({ author: 1 })

> db.posts.find({author: 'roger'})

{ _id      : ObjectId("4c4ba5c0672c685e5e8aabf3"),
  author   : "roger",
  ...
}
```

Query Operators

Conditional Operators

- $all, $exists, $mod, $ne, $in, $nin, $nor, $or, $size, $type, $lt, $lte, $gt, $gte

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

// find posts matching a regular expression
> db.posts.find( {author: /^rog*/i } )

// count posts by author
> db.posts.find( {author: ‘roger’} ).count()
```

Atomic Operators

$set, $unset, $inc, $push, $pushAll, $pull, $pullAll, $bit

```
> comment = { author:“fred”,
              date: new Date(),
              text:“Best Movie Ever”}


> db.posts.update( { _id:“...” },
                   $push: {comments: comment} );
```

10gen mongoDB

Nested Documents

```
{   _id : ObjectId("4c4ba5c0672c685e5e8aabf3"),
    author : "roger",
    date : "Sat Jul 24 2010 19:47:11 GMT-0700 (PDT)",
    text : "Spirited Away",
    tags : [ "Tezuka", "Manga" ],
    comments : [
      {
            author : "Fred",
            date : "Sat Jul 24 2010 20:51:03 GMT-0700 (PDT)",
            text : "Best Movie Ever"
      }
    ]
}
```

Multiple Indexes Collections

```
// Index nested documents
> db.posts.ensureIndex( “comments.author”:1 )
 ➢db.posts.find({‘comments.author’:’Fred’})

// Index on tags
> db.posts.ensureIndex( tags: 1)
> db.posts.find( { tags: ’Manga’ } )

// geospatial index
> db.posts.ensureIndex( “author.location”: “2d” )
> db.posts.find( “author.location” : { $near : [22,42] } )
```

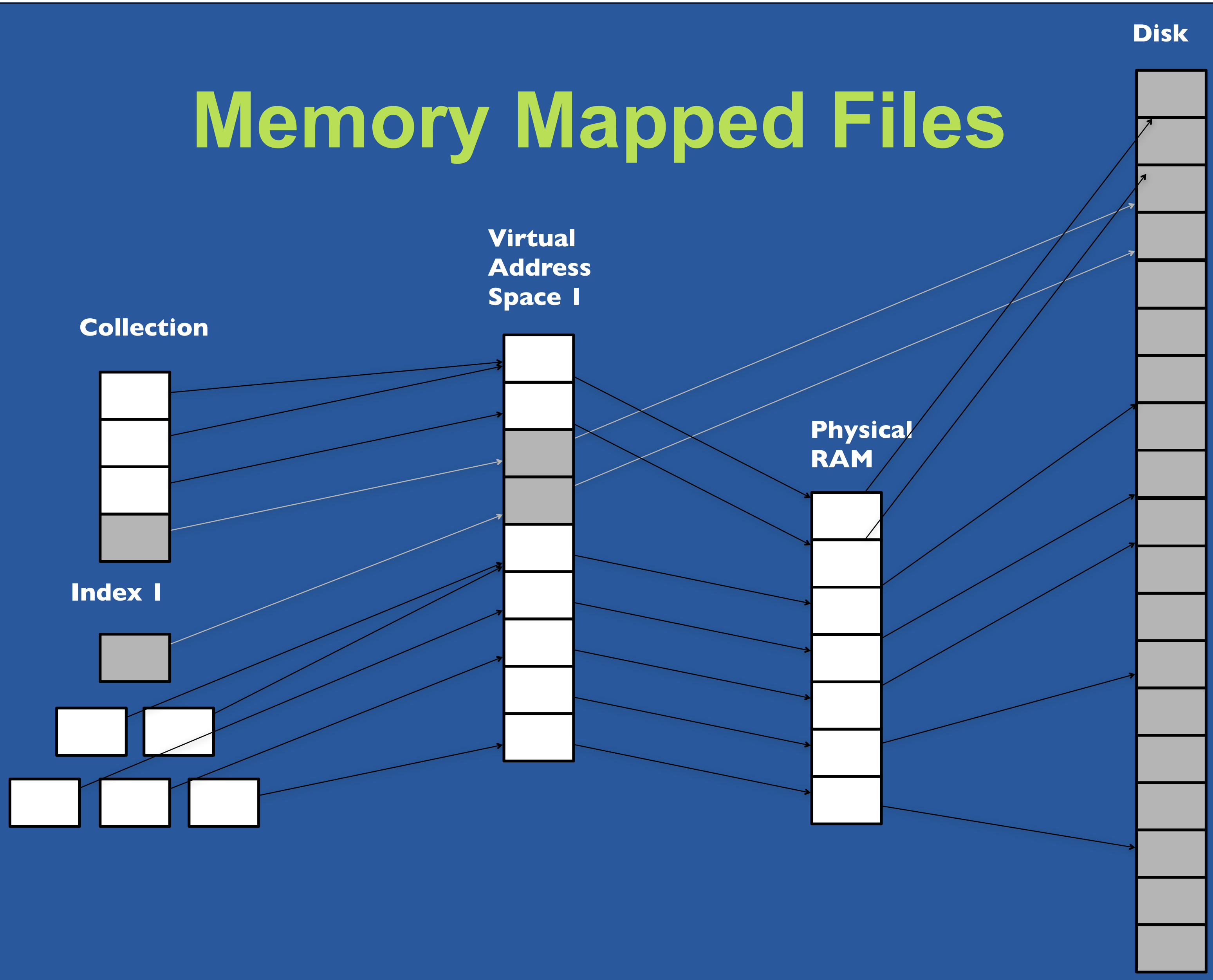
Terminology

RDBMS	MongoDB
Table	Collection
Row(s)	JSON Document
Index	Index
Join	Embedding & Linking
Partition	Shard
Partition Key	Shard Key

How does it work?

MongoDB revolves around memory mapped files

Memory Mapped Files



Operating System map files on the Filesystem to Virtual Memory

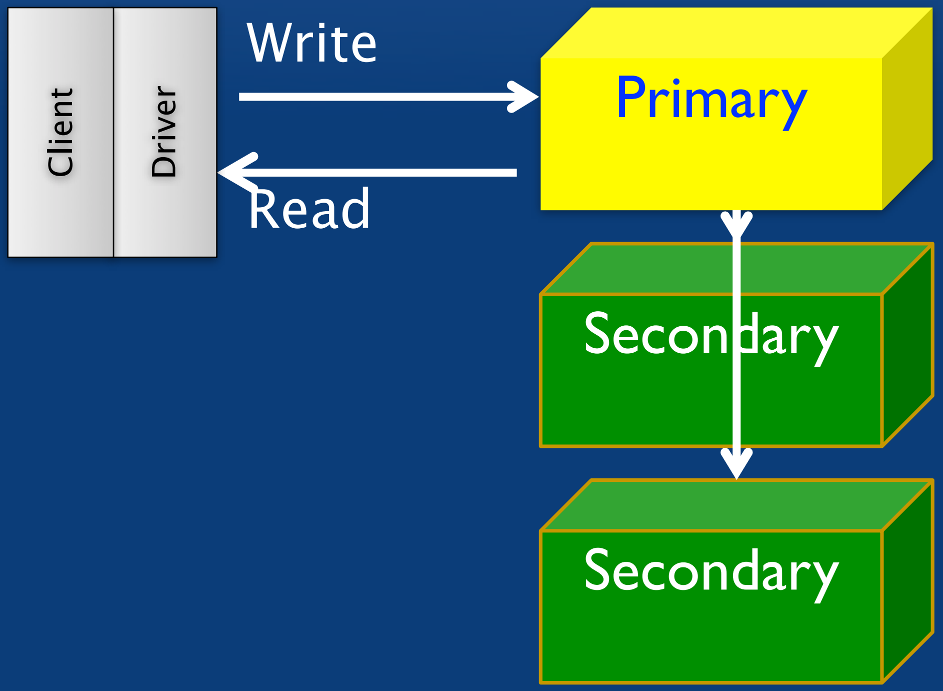
- 200 gigs of MongoDB files creates 200 gigs of virtual memory
- OS controls what data in RAM
- When a piece of data isn't found in memory
  - OS goes to disk to fetch the data
- Indexes are part of the Regular Database files

10gen mongoDB

MongoDB Replication

- MongoDB designed for high availability & data durability
    - Multi Server environment
- MongoDB earlier versions
    - Replication like MySQL replication
    - Asynchronous master/slave
- Later & Current versions
    - Replica Sets

Replication with Replica Sets



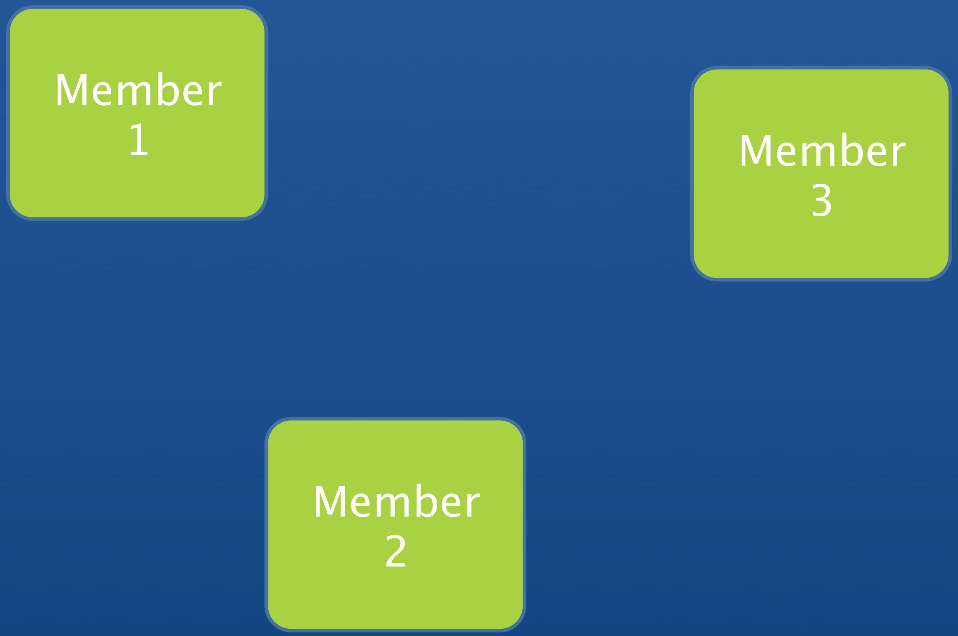
10gen

mongoDB

Replica Set features

- A cluster of N servers
- Any (one) node can be primary
- Consensus election of primary
- Automatic failover
- Automatic recovery
- All writes to primary
- Reads can be to primary (default) or a secondary

How MongoDB Replication works



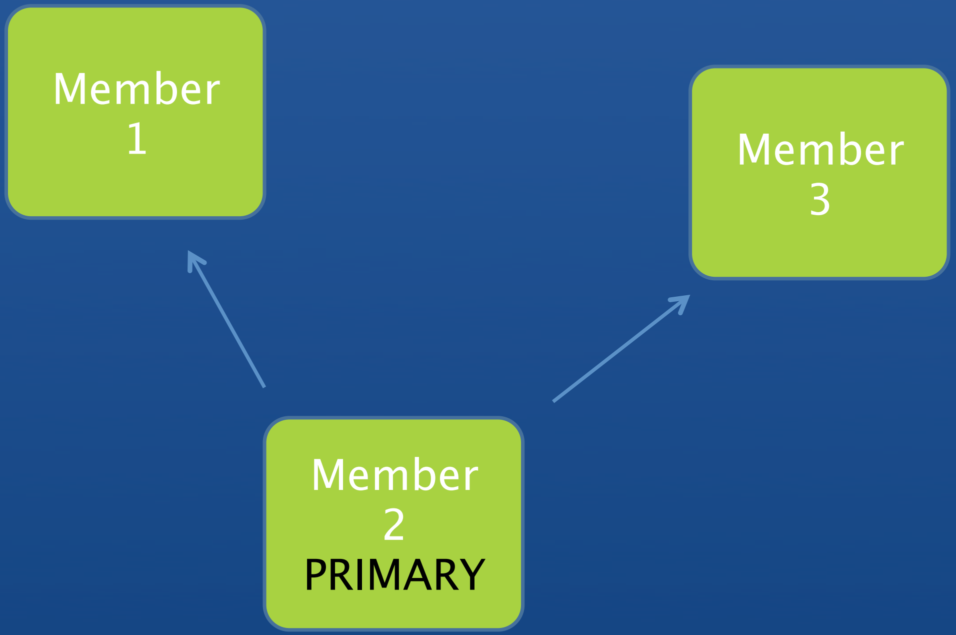
Member 1

Member 3

Member 2

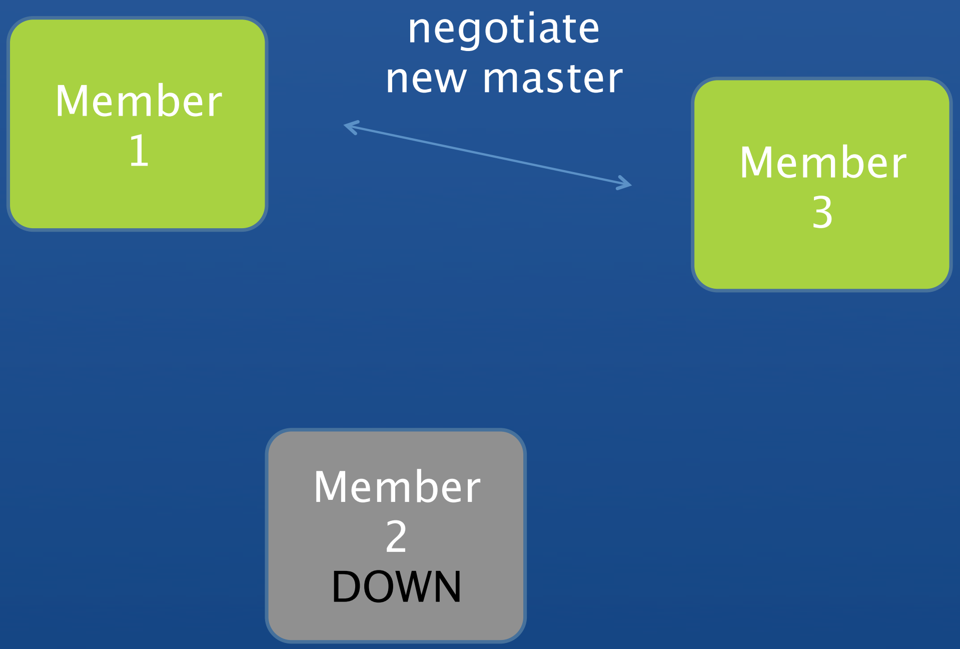
- Set is made up of 2 or more nodes

How MongoDB Replication works



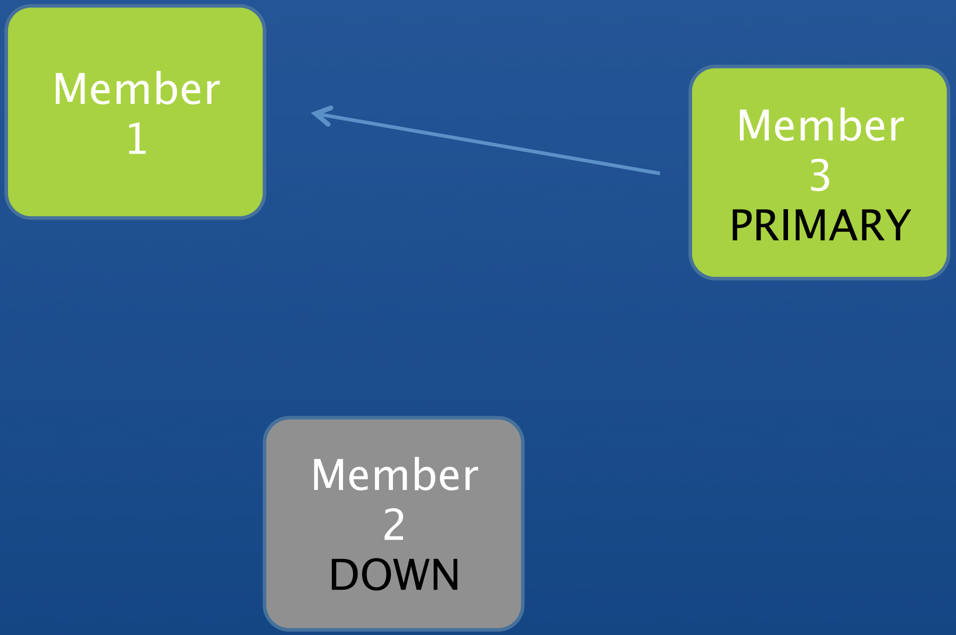
- Election establishes the PRIMARY
- Data replication from PRIMARY to SECONDARY

How MongoDB Replication works



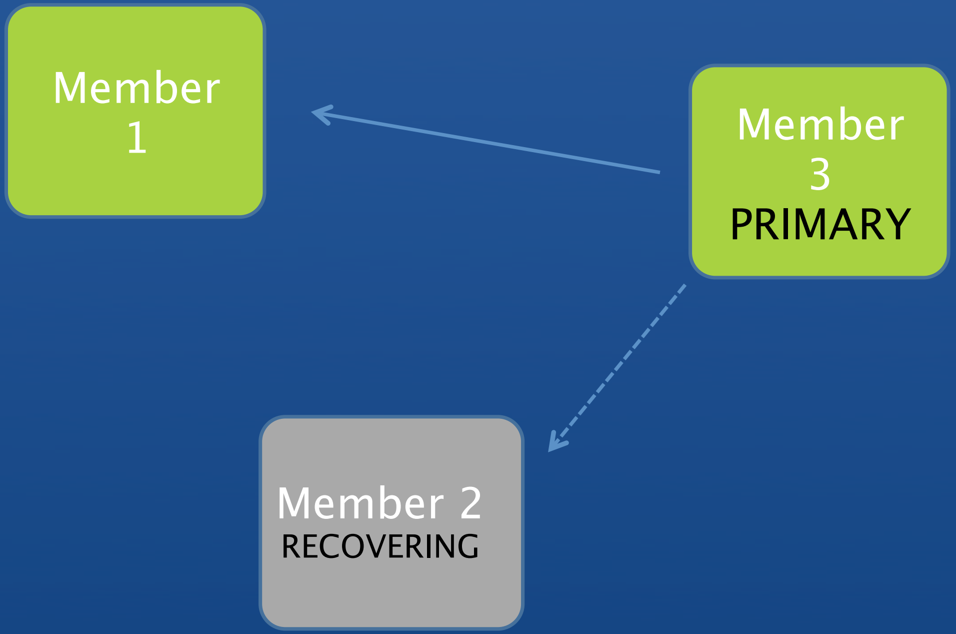
- PRIMARY may fail
- Automatic election of new PRIMARY if majority exists

How MongoDB Replication works



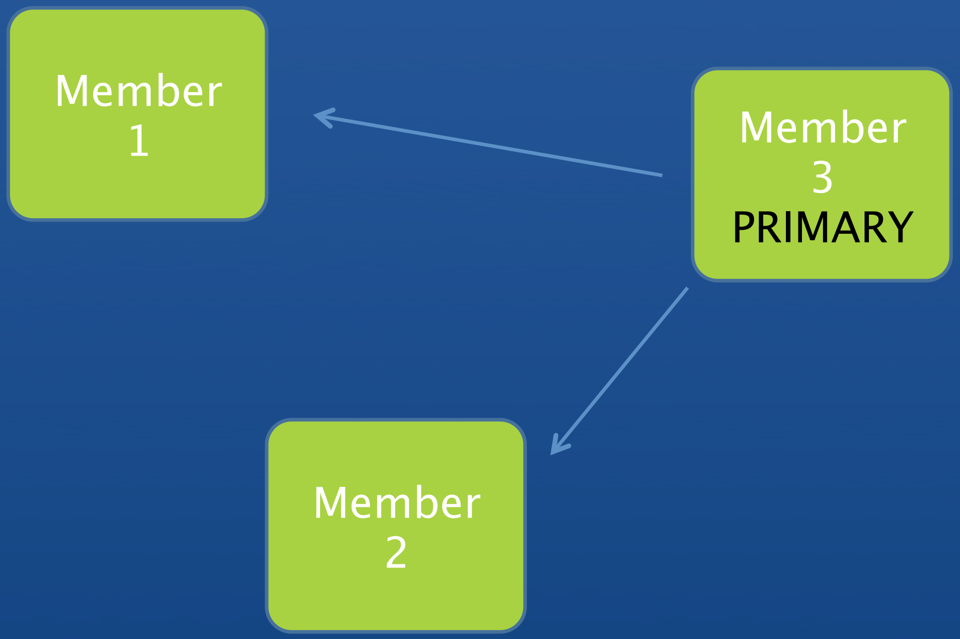
- New PRIMARY elected
- Replication Set re-established

How MongoDB Replication works



- Automatic recovery

How MongoDB Replication works



Member 1

Member 3 PRIMARY

Member 2

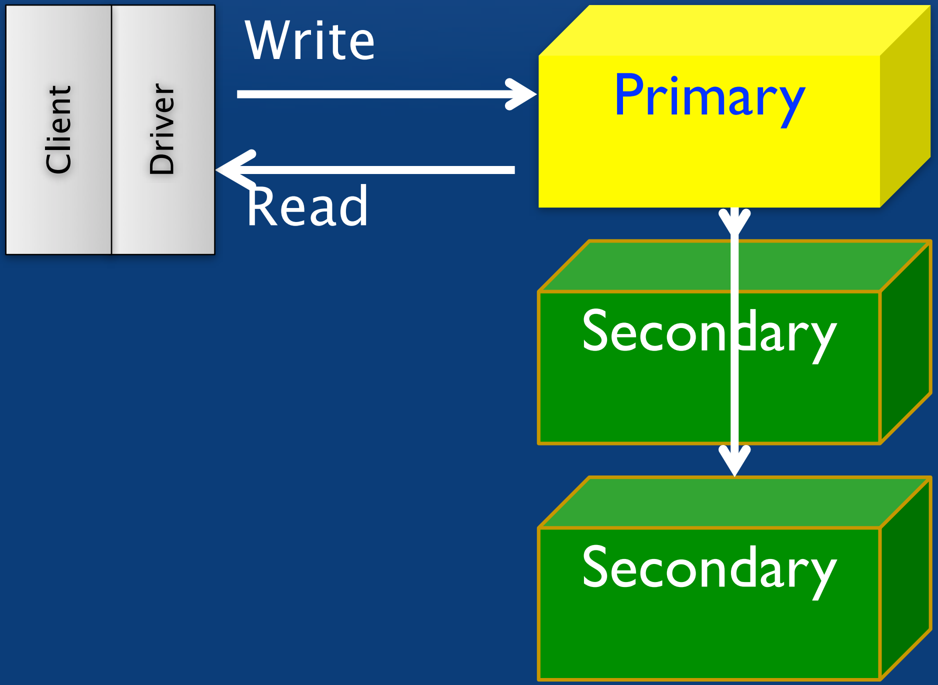
- Replication Set re-established

Configuring a Replica Set

```
> cfg = {
    _id : "acme_a",
    members : [
      { _id : 0, host : "sf1.acme.com" },
      { _id : 1, host : "sf2.acme.com" },
      { _id : 2, host : "sf3.acme.com" } ]
  }
> use admin

> db.runCommand( { replSetInitiate : cfg } )
```

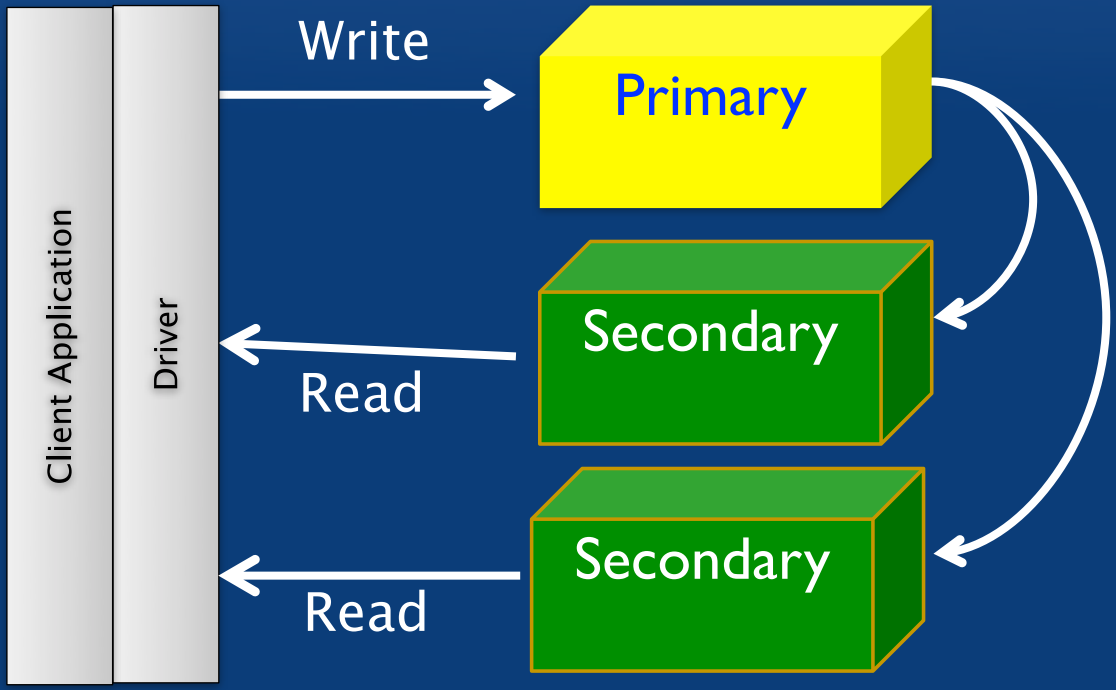
Strong Consistency



10gen

mongoDB

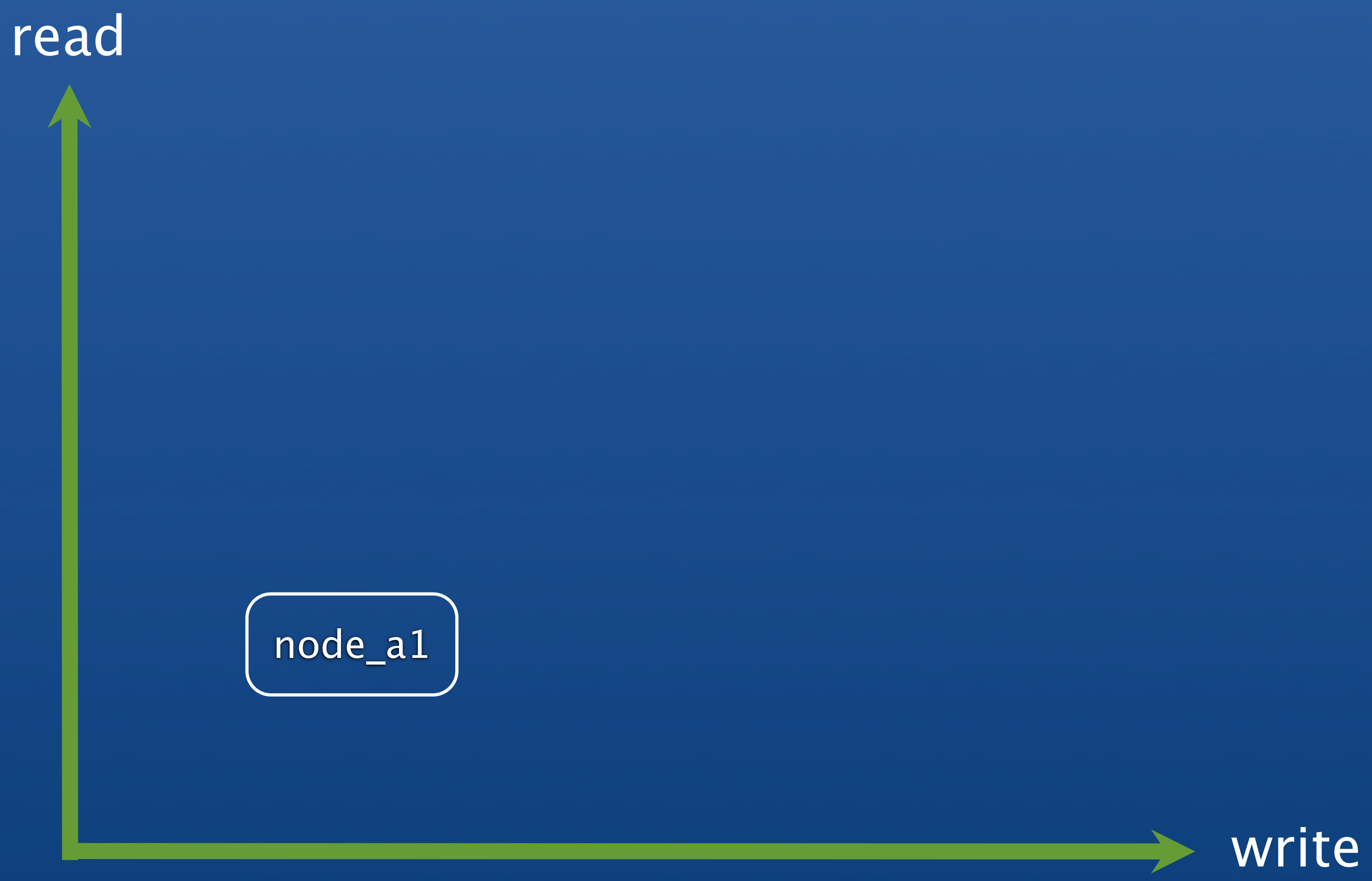
Eventual Consistency



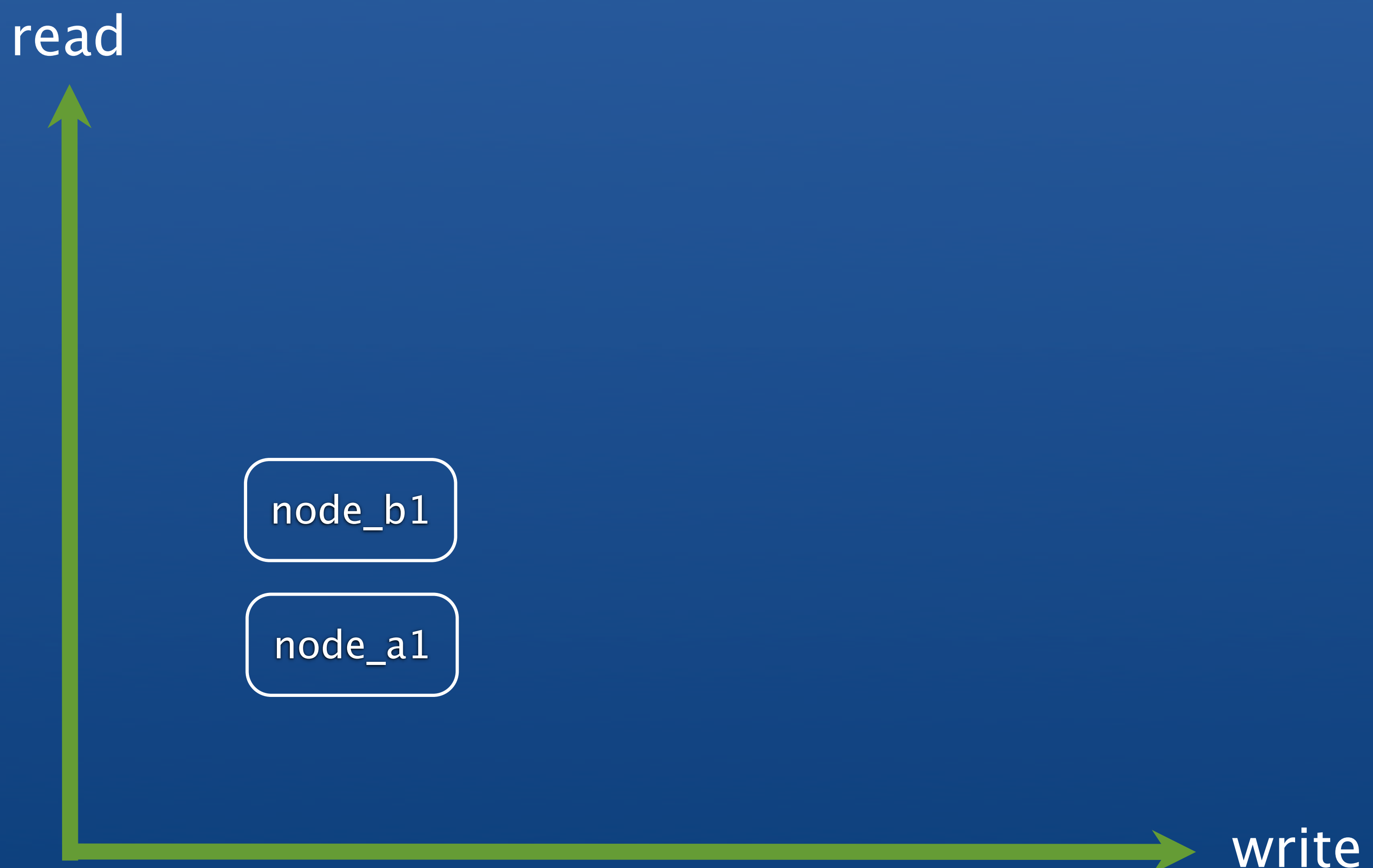
http://community.qlikview.com/cfs-filesystemfile.ashx/__key/CommunityServer.Blogs.Components.WeblogFiles/theqlikviewblog/Cutting-Grass-with-Scissors-_2D00_-2.jpg

http://www.bitquill.net/blog/wp-content/uploads/2008/07/pack_of_harvesters.jpg

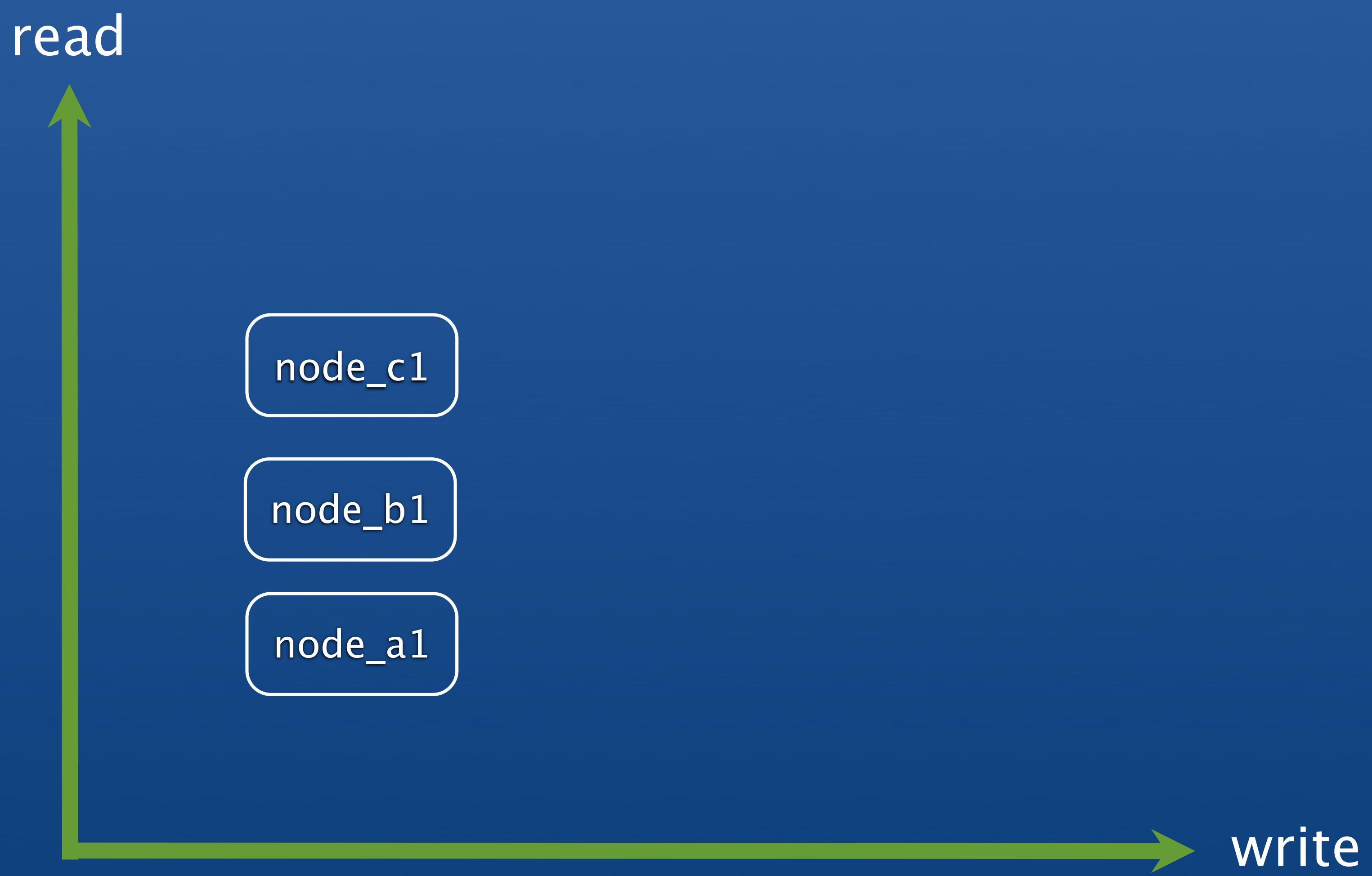
MongoDB Scaling - Single Node



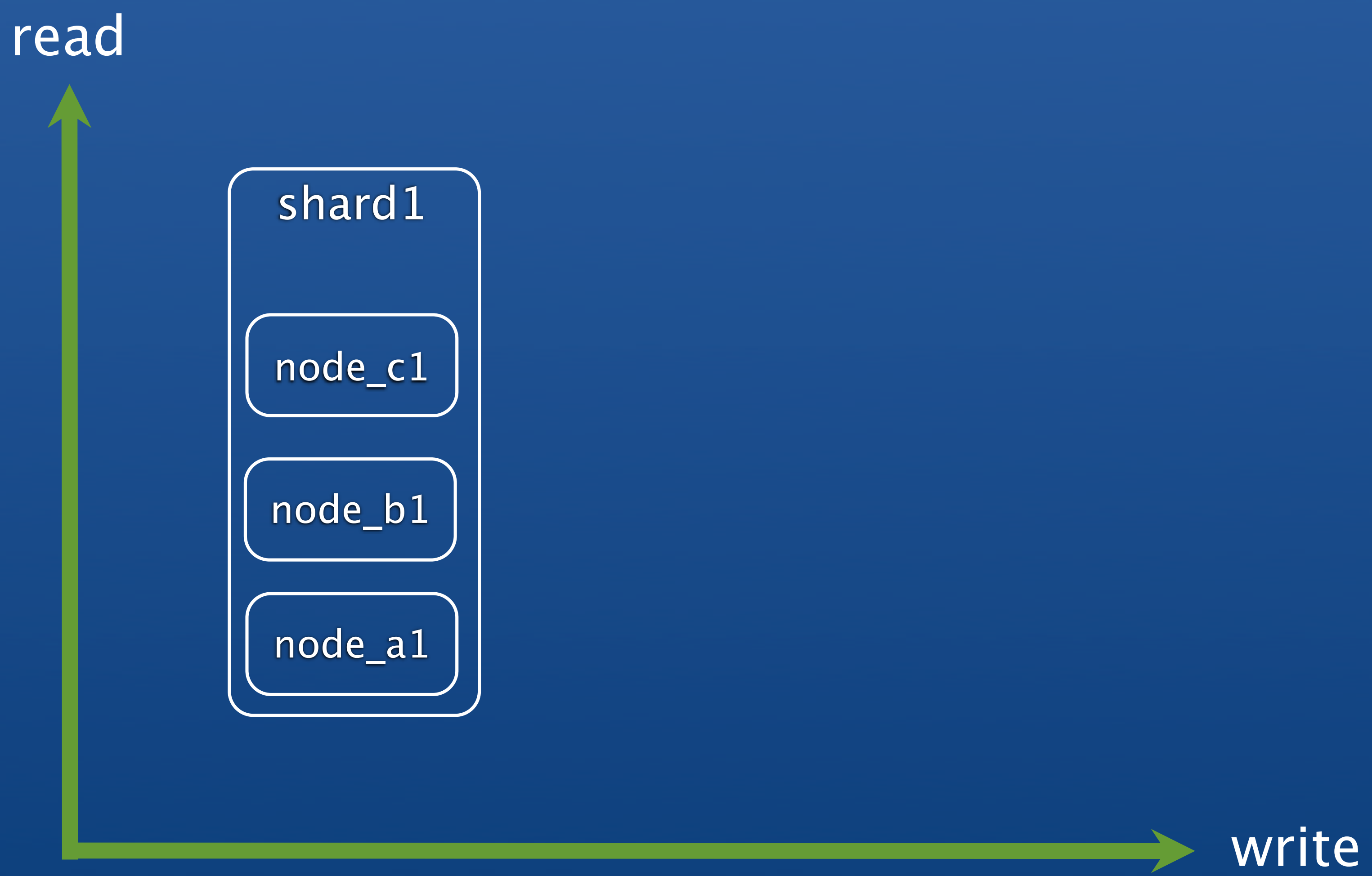
Read scaling - add Replicas



Read scaling - add Replicas

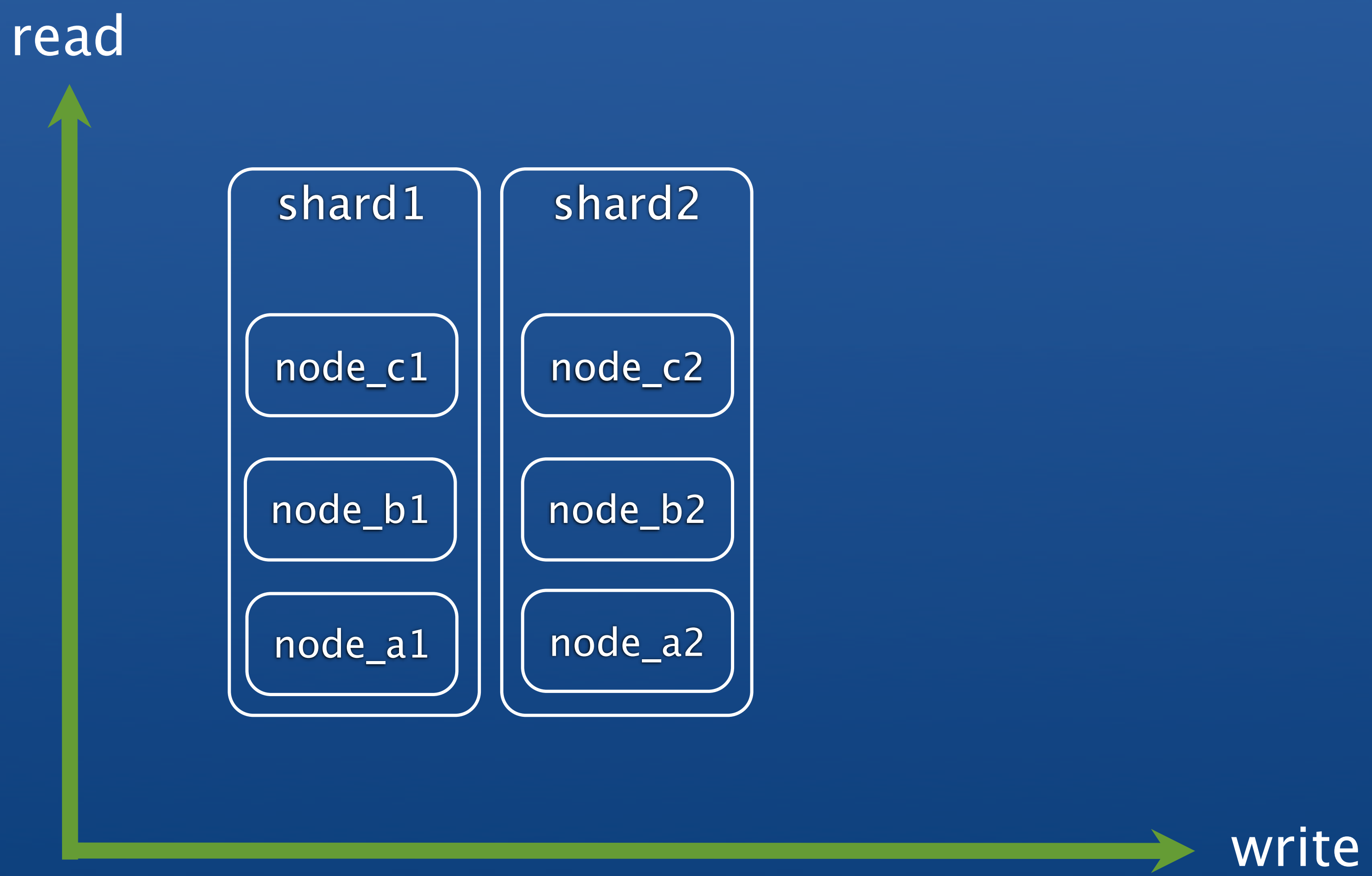


Write scaling - Sharding

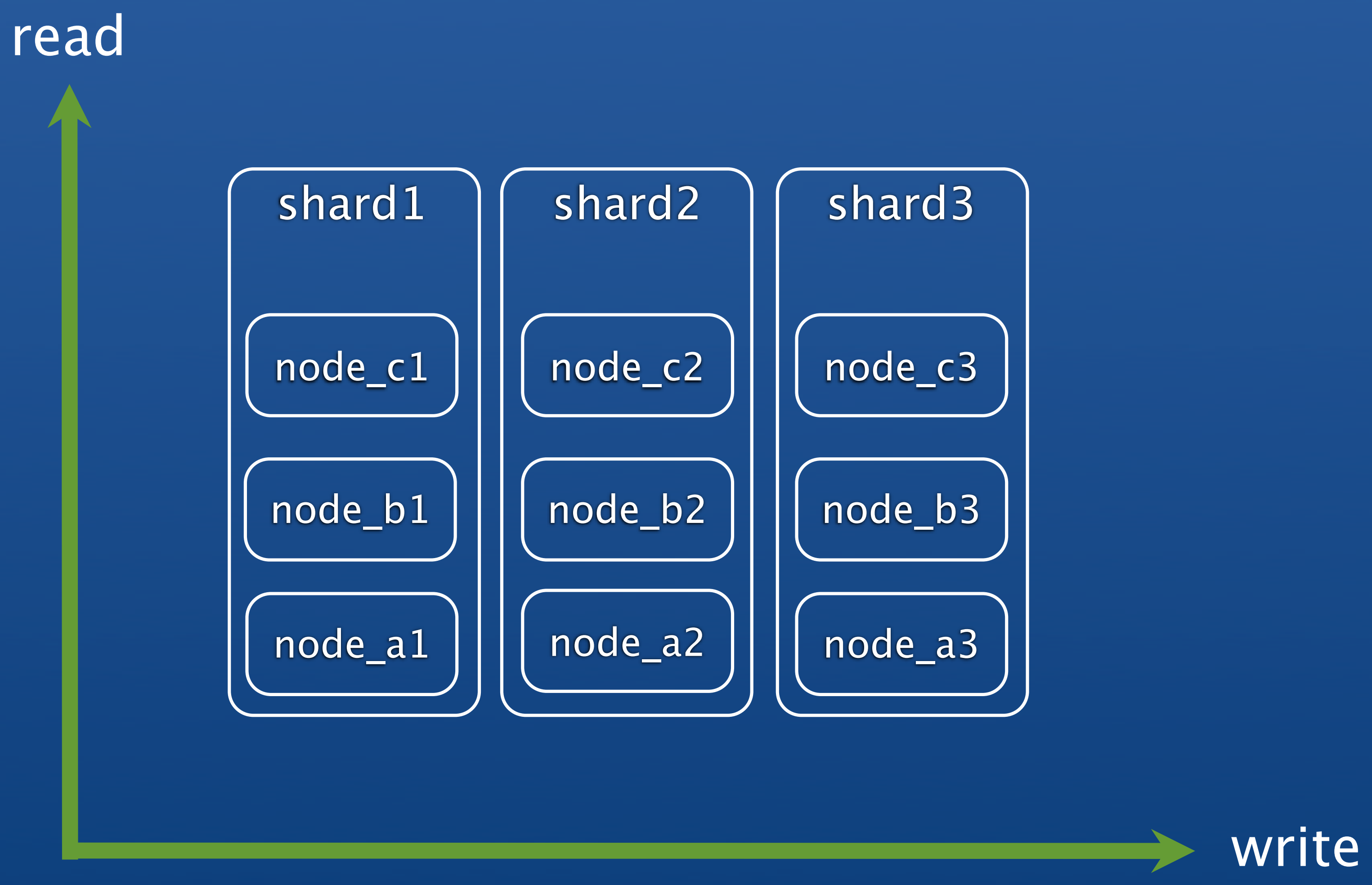


10gen mongoDB

Write scaling - add Shards



Write scaling - add Shards



read

shard1

node_c1

node_b1

node_a1

shard2

node_c2

node_b2

node_a2

shard3

node_c3

node_b3

node_a3

write

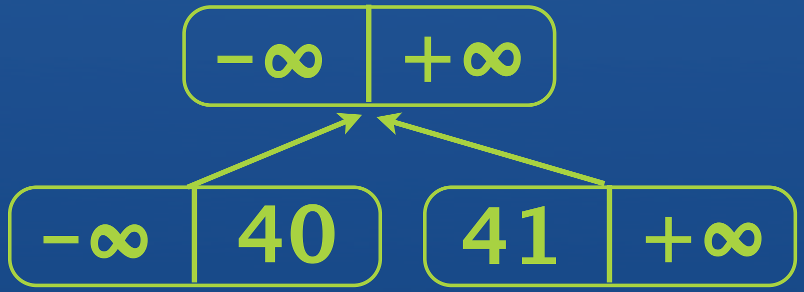
10gen mongoDB

MongoDB Sharding

- Automatic partitioning and management
- Range based
- Convert to sharded system with no downtime
- Fully consistent

How MongoDB Sharding works

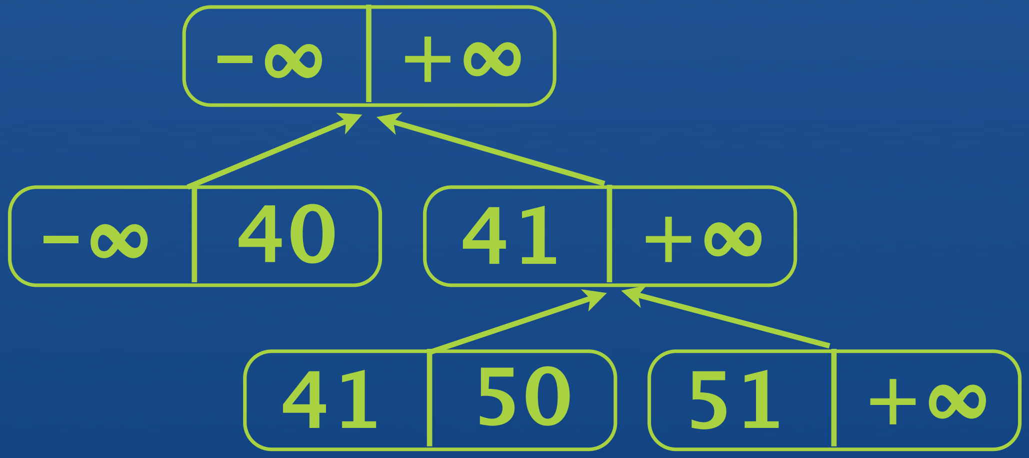
```
> db.posts.save( {age:40} )
```



- Data in inserted
- Ranges are split into more “chunks”

How MongoDB Sharding works

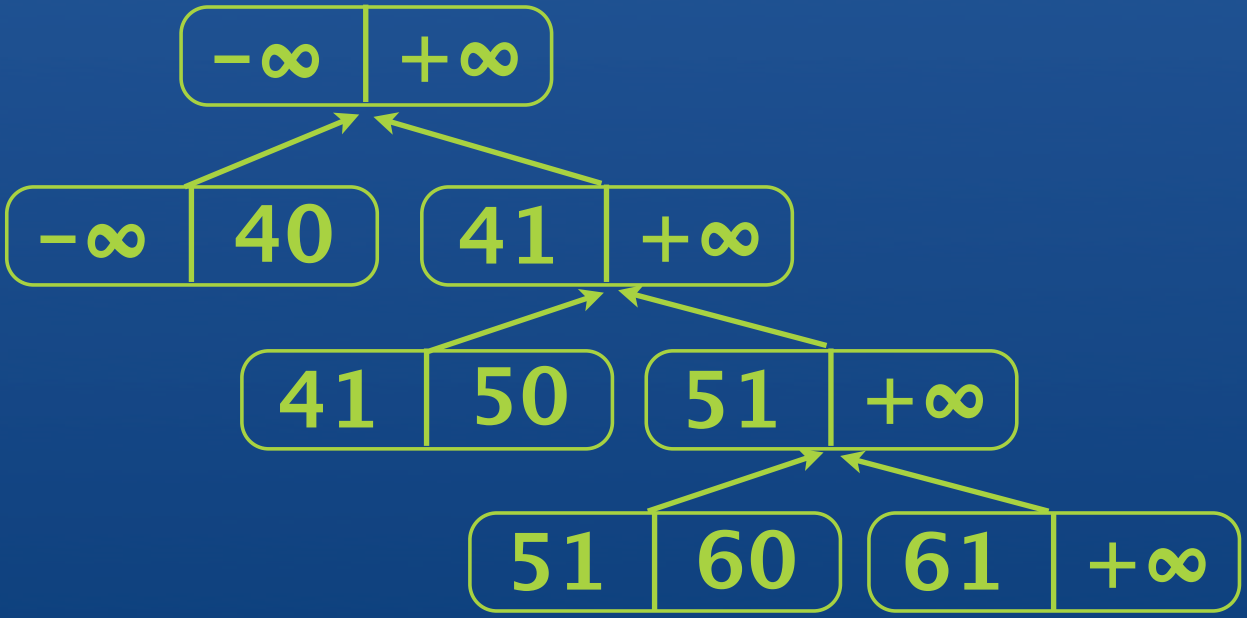
```
> db.posts.save( {age:40} )
> db.posts.save( {age:50} )
```



- More Data in inserted
- Ranges are split into more“chunks”

How MongoDB Sharding works

```
> db.posts.save( {age:40} )
> db.posts.save( {age:50} )
> db.posts.save( {age:60} )
```



−∞ | +∞

−∞ | 40

41 | +∞

41 | 50

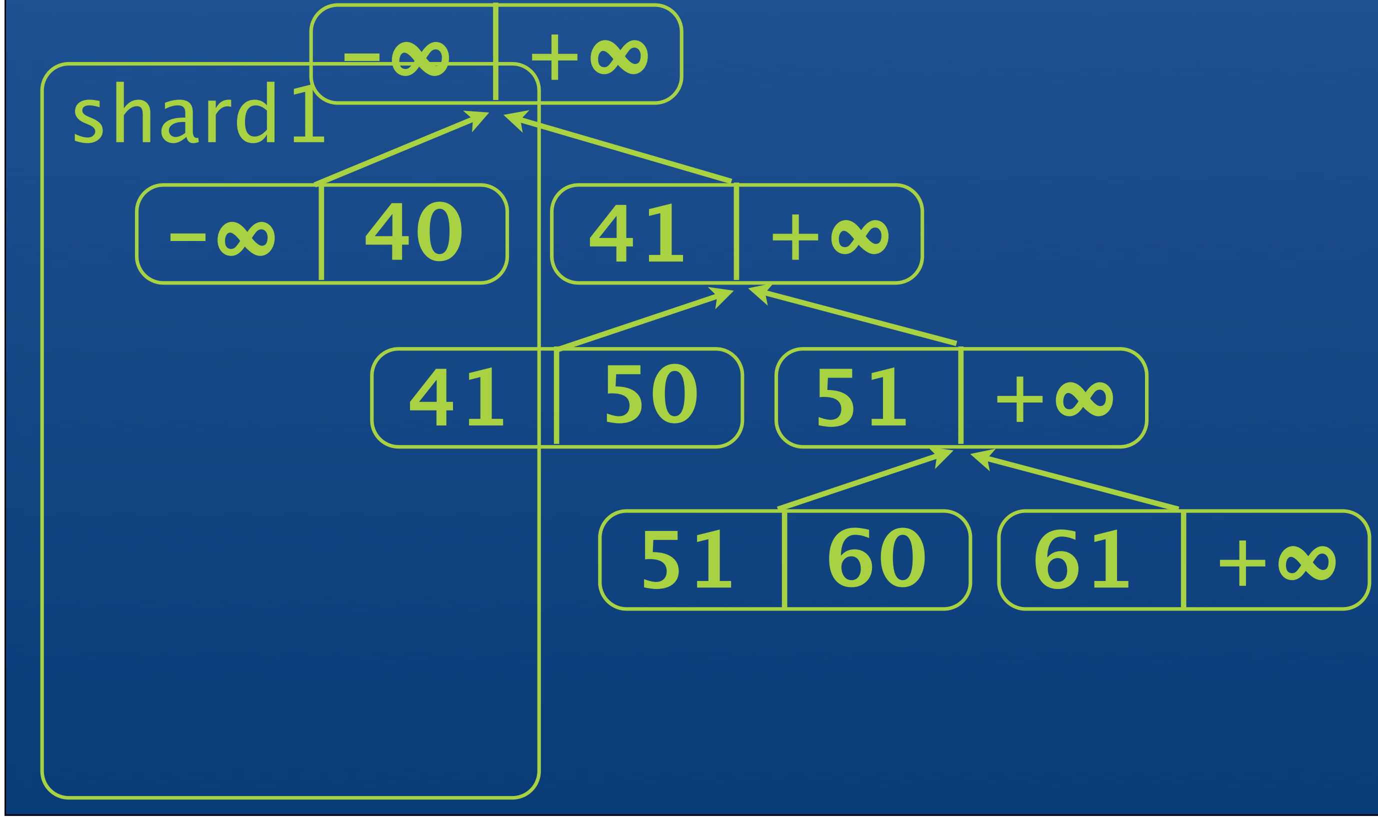
51 | +∞

51 | 60

61 | +∞

How MongoDB Sharding works

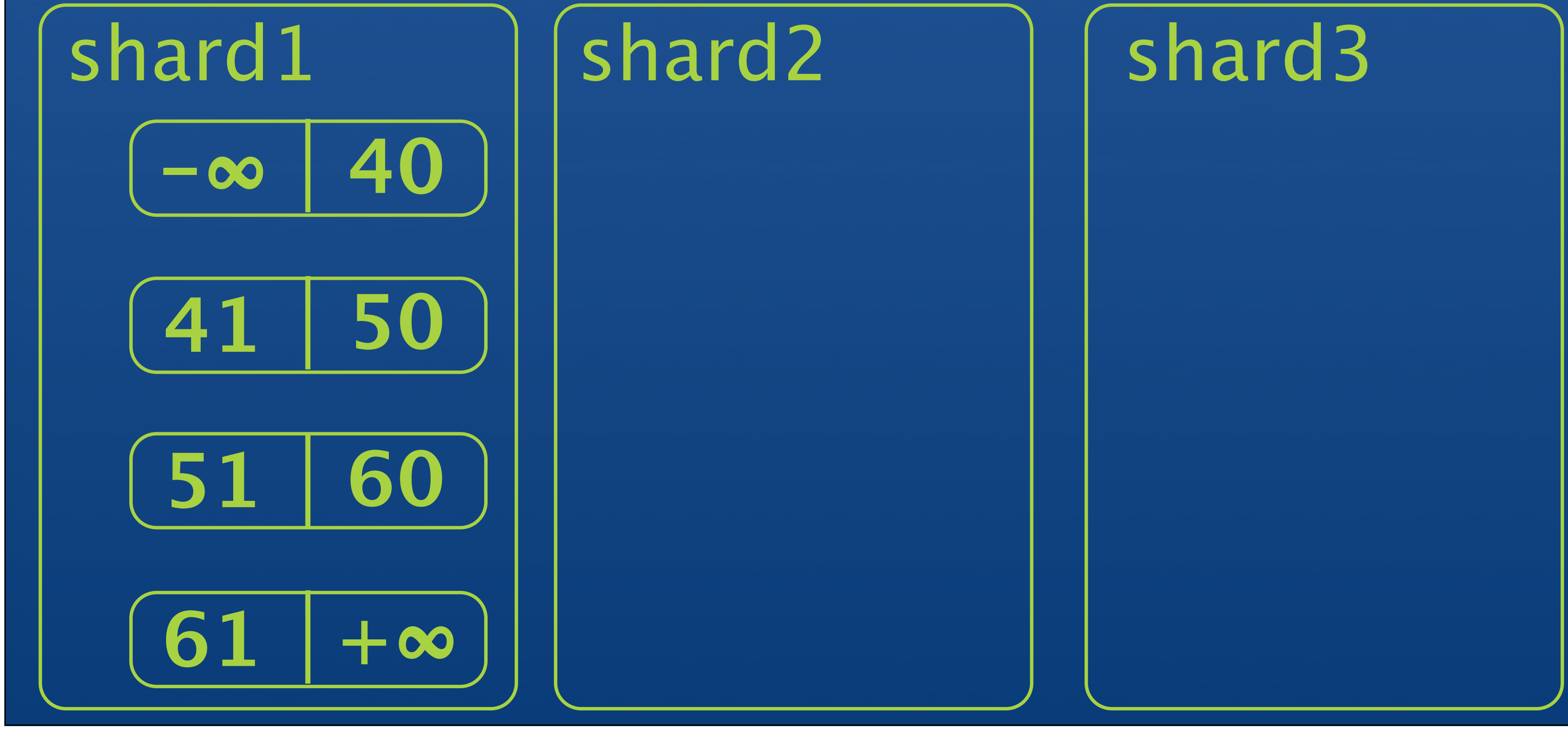
```
> db.posts.save( {age:40} )
> db.posts.save( {age:50} )
> db.posts.save( {age:60} )
```



How MongoDB Sharding works

```
> db.runCommand( { addshard : "shard2" } );

> db.runCommand( { addshard : "shard3" } );
```



shard1

-∞ | 40

41 | 50

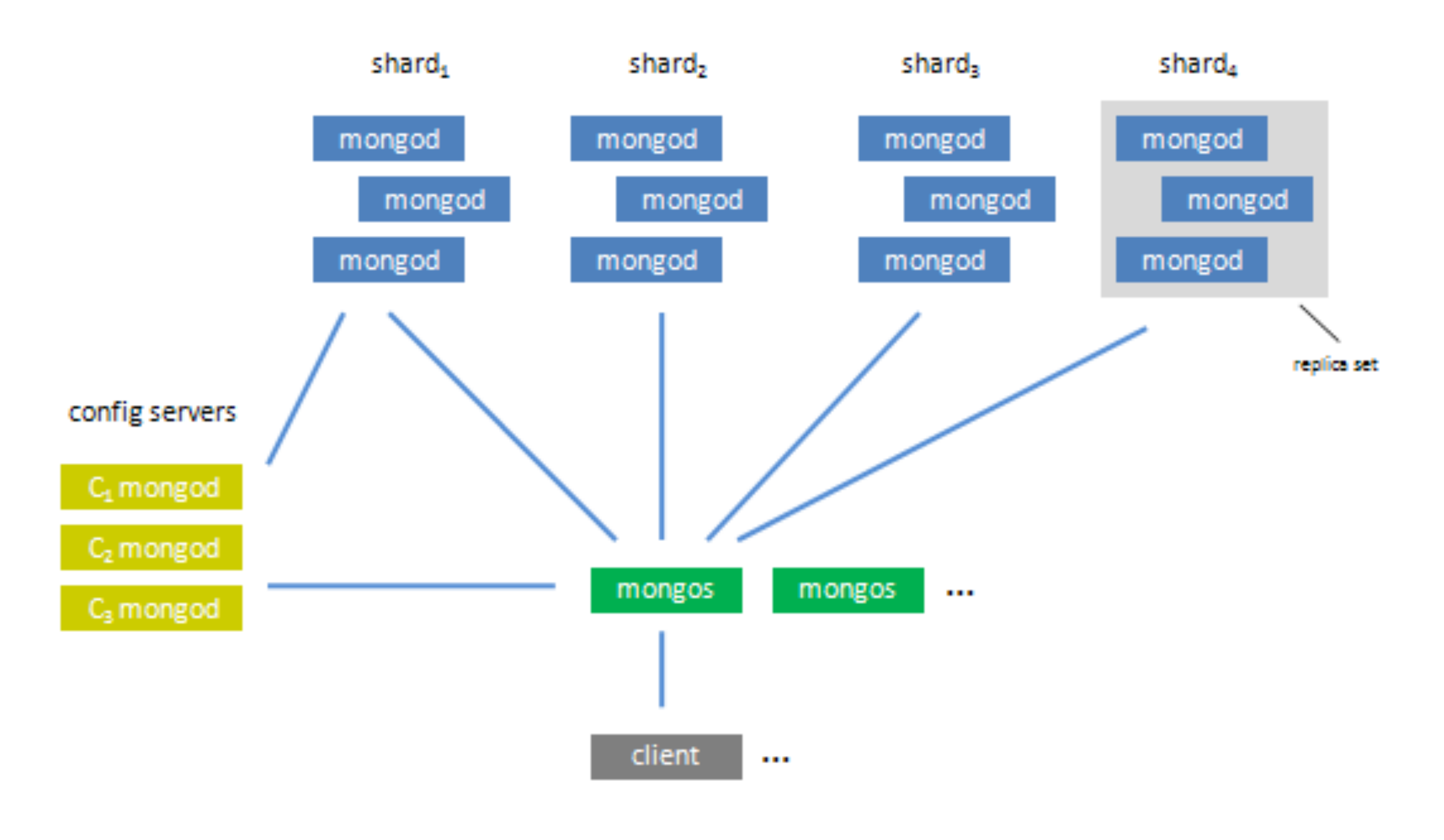
51 | 60

61 | +∞

shard2

shard3

Architecture



Accessing MongoDB...

Access from Java

- Options:
  - Use the MongoDB Java driver
    - Raw driver
      - Can submit JSON
      - Or use helper objects to but documents
    - Supported by 10gen
  - Alternatively utilise a Document Mapping layer

10gen mongoDB

A New Paradigm for Mapping Objects <-> Documents

- While the ORM Pattern can be a disaster, well designed Documents map well to a typical object hierarchy

- The world of ODMs for MongoDB has evolved in many languages, with fantastic tools in Scala, Java, Python and Ruby

- Typically “relationship” fields can be defined to be either “embedded” or “referenced”

ODM Systems for the JVM
< Java >

- Two Major ODMs in the Java World
  - Morphia
    - JPA Inspired
    - Annotation Driven; Able to integrate with existing objects
    - Written purely for MongoDB, strong coupling
  - Spring Data for MongoDB
    - Part of the Spring-Data System
    - Follows the Spring paradigms; comfortable to the Spring veteran
    - Designed for multiple datastores

ODM Systems for the JVM
< Scala >

- Two Major ODMs in the Scala World
    - Lift-MongoDB-Record
        - Based on the Record pattern
        - Requires entirely custom objects following Record paradigm
        - Strongly coupled to MongoDB but still bound by Record
    - Salat
        - Built by same team who helped start Casbah (scala driver)
        - Annotation driven, built from ground up to apply onto existing business objects cleanly
        - Strongly coupled to MongoDB

10gen mongoDB

Hadoop and MongoDB

- Input and Output Formats for MongoDB + Hadoop
- Process MongoDB data inside of Hadoop, output back out to MongoDB
- Now in RC - mongo-hadoop 1.0.0-rc0

Summary

- Document-Oriented
    - Dynamic schema
    - agile
    - flexible
- High Performance
- Highly available
    - Replica Sets
- Horizontal Scale Out
    - Sharded cluster

download at mongodb.org

We’re Hiring !

@dmroberts

daniel.roberts@10gen.com

conferences, appearances, and meetups

http://www.10gen.com/events

Facebook	Twitter	LinkedIn
http://bit.ly/mongofb	@dmroberts	http://linkd.in/joinmongo

10gen

mongoDB