MongoDB

Department of Software Systems

Presenter:
Saira Shaheen, 227233
saira.shaheen@tut.fi
0417016438
Dated: 02-10-2012

TAMPERE UNIVERSITY OF TECHNOLOGY

Contents

- Motivation : Why noSQL ?
- Introduction : What does NoSQL means??
- Applications : Where NoSQL ?

MongoDB

- Motivation
- Introduction
  - Terminology
  - Language and platform support
- MongoDB design session
  - Blog Post Document
  - Query Posts Collection
  - Secondary Index
  - Query operators
  - Extending the schema

- Deploying MongoDB?
- Setting up MongoDB?
  - Read Scalability: Harding
  - Write Scalability : Sharding
  - Read Scalability : Replication
- Advanthges:MongoDB makes building applications easy
- Drawbacks
- Summary


Business
Intelligence
benchmarking
reporting
information
Intelligence
program
Business
data
software
management
internet
metrics
modeling
used
company
builds
mining
performance
system
computer
Strategic
analytics
predictive
Knowledge
process
processes
technologies
Frequently
technology
BI
competitive
decision
needed
support
involves
Regulatory
applications
decisions
current
practices
informs
leads
incomes
OLAP
hierarchy
analyze
identifying
operations
processing
statistical
revenue
competitors
decision-making
compliance/Compliance
leaders
functions
sometimes
views
focus
create
Reporting/Enterprise
purposes
towards
topical
serve
together
MARCKM
costs
Electronic
strategies
Learning
products
associated
structured
applied
Common
Executive
include
Thus
citation
gets
NoSQL

Why NoSql??

- Linear Scalability
- Schema flexibility
- High Performance

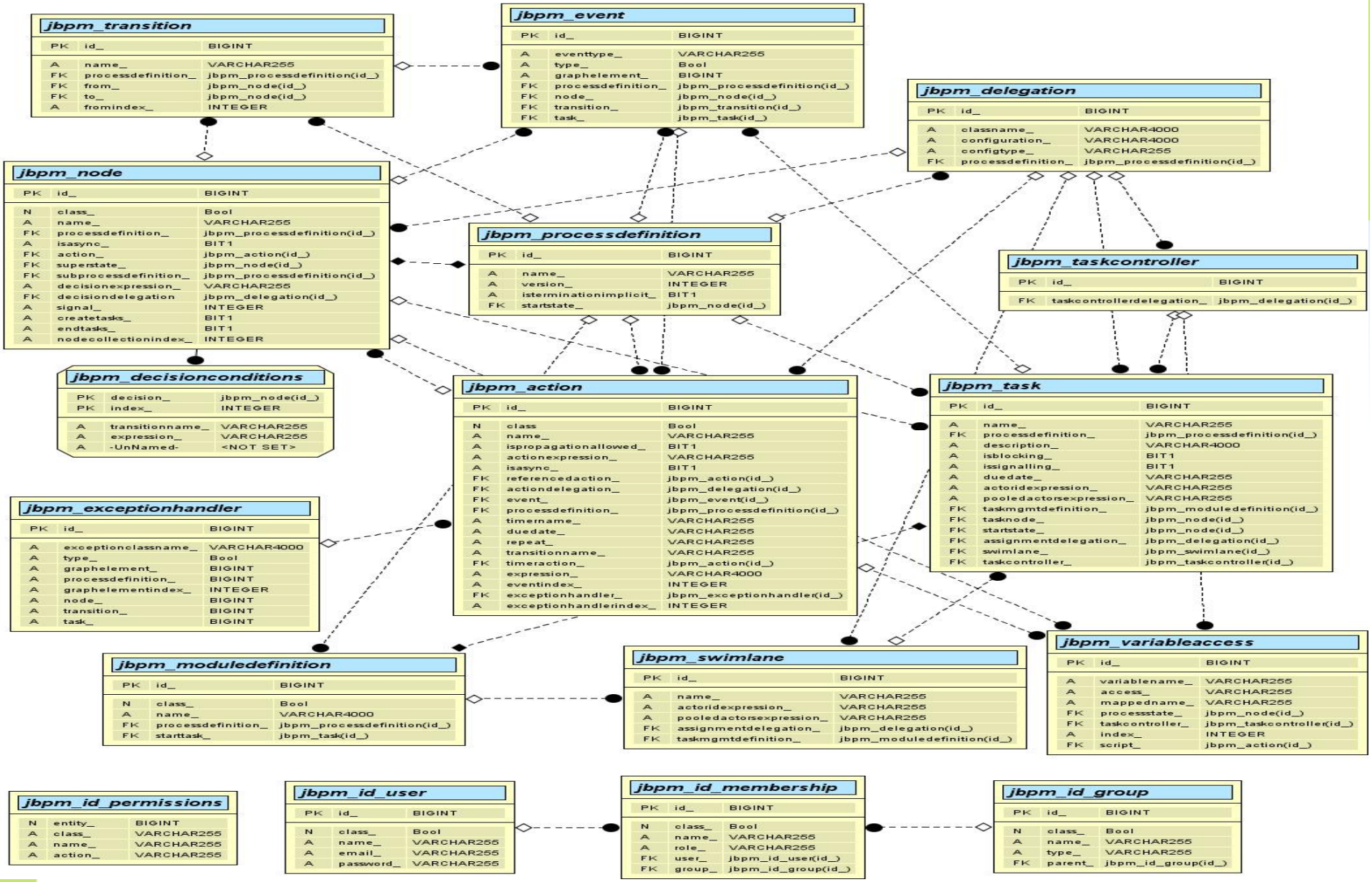
→ **NoSQL**

- Multi-document transactions
- Complex security needs
- Complex joins
- Extreme compression needs

→ **RDBMS**

- Both / depends on the data

→ **RDBMS** + **NoSQL**



For complex applications ,relational database slowing your data and SQL generation becomes very very complex

Introduction : What does NoSQL means??

Does not use SQL as querying language

No joins

\+ No complex transaction

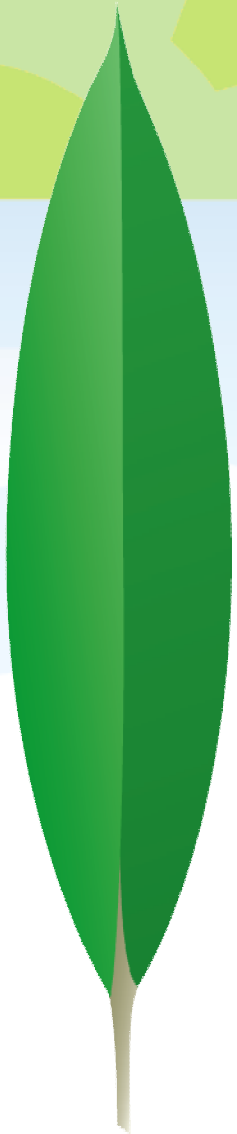
Turns out into no fixed schema means innovation of new data model and scaling becomes fairly easily

Also provides high performance

Applications: Giant applications……

a) **Facebook(Cassandra)**
b) **Netflix (SimpleDB, Hadoop/HBase, Cassandra)**
c) **Google (BigTable, LevelDB)**
d) **CERN (CouchDB)**

All use NoSQL in one way or another because of the significantly different challenges in dealing with huge quantities of data that the traditional RDBMS solutions could not cope with.



mongoDB

Contents

MongoDB

- Motivation : Why MongoDB?
- Introduction :What does MongoDB means?
  - Terminology
  - Language and platform support
- MongoDB design session
  - Inserting data into a collection
  - Accessing data from query
  - Secondary Index
  - Query operators
  - Extending the schema

Background : Motivation??

- Modern application are more networked, social and interactive the network.
- This is driving new requirement to support
  - Big data
  - Fast feature development
  - Flexible deployment strategy.

But is your database ready??

Background : Motivation??

- Applications are storing more and more data and accessing it high and higher rates.
- If your database runs on single server. You will reach a scaling limit.

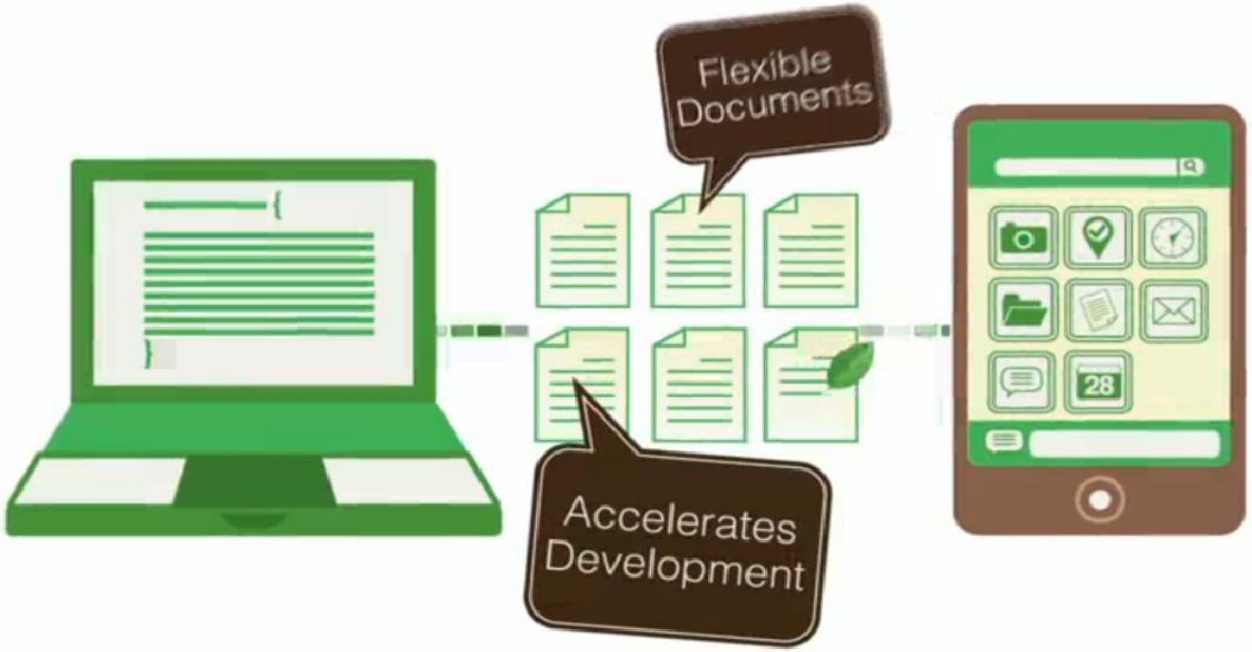
Scales

mongoDB scales by adding more servers. You can add more capacity whenever you want.

Background : Motivation??

Productivity

- Agile development and continuous deployment accelerated road maps.
- If your data model isn't flexible. You can slow development down.
- Mongo DB increases productivity
- Modeling document simpler



Background : Motivation??

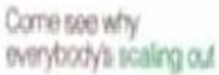
Commodity servers

- Cloud computing is changing as we deploy the applications both inside and outside the firewall.
- If your database needs complex hardware, its not ready for cloud, keeps you stuck with expensive data centers.
- MongoDb was designed to work with commodity servers, saving your money and headaches.

Come see why
everybody's scaling out

chfork

edhat




redhat
craigslist
EDITD
The New York Ti
IGN
Eventbrite
MongoDB is in the sweet spot of performance and features. Its rich data types, querying, and in place updates reduced development time to minutes from days for modeling rich domain objects. —Chandra Patni, IGN
Server Density
SQUARESPACE
bitly
songkick
SAP

MongoDB is a document database, so it really enriched our data modeling abilities and allowed for new query patterns unavailable before. —Kenny Gorman, Shutterfly


mongoDB
Scalable. Agile. Cloud Ready.

What does MongoDB really means?

MongoDB =>(from "hu**mongo**us")

• Scalable, high-performance, open source, non-relational ,document based NoSQL database

- 10g started development in **2007**
  - DoubleClick Founder
  - CTO Dwight Merriman and former DoubleClick engineer and ShopWiki Founder and CTO Eliot Horowitz.
- Adoption very strong
  - 90,000 downloads of database per month

MongoDB adoption is very strong

90,000
Database
downloads
per month

Language and Platform support

- Implemented in C++for best performance
- Support platforms 32/64 bit
  - Windows
  - Linux, Mac OS-X, FreeBSD, Solaris
- Language drivers for
  - Ruby/Ruby on Rails, Java / C# / Javascript

Terminology

- **Objects**
  - Deal with a objects
- **JSON-like documents**
  - Store JSON-like documents with dynamic schemas.
- **No Fix Schema**
  - No predefinition of fields
  - No schema for fields within documents – the fields and their value datatypes can vary.
  - No notion of an "alter table" operation which adds a "column".
  - Rarely will you need to write scripts which perform "alter table" type operations.

Dynamic Schema ("Schema Free")

- MongoDB has
  - Databases
  - Collections
  - indexes much like a traditional RDBMS.

In some cases (databases and collections) these objects can be implicitly created, however once created they exist in a system catalog (db.systems.collections, db.system.indexes).

Mongo data model

A Mongo system holds a set of databases

A database holds a set of collections

A collection holds a set of documents

A document is a set of fields

A field is a key-value pair

A key is a name (string)

A value is a

- basic type like string, integer, float, timestamp, binary, etc.,
- a document, or
- an array of values

Traditional Database	MongoDB
Relational	Document-Orientated
Server	Server
Database	Database
Table	Collection
Row	Document
Column	Attribute
SQL Query	BSON Query
Index	Index

Walkthrough

MongoDB makes building applications easy

Designed a Blog Schema
Evolved the Schema
Deployed MongoDB
Scale MongoDB

Design Session

Design Session

- Download live, Install live
- Extemely easy to get start with it, takes less than 5 minutes to get the set up

Design Session

Inserting data into a collection

Collection : posts

Object : p

Commands at shell prompt : '>‘

Blog post document

```
P = { author:”saira”,
      date: new Date();
      text: “cloud stack”;
      tags : [“peace”,”Love”]}

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

Inserting data into a collection(things)

Let's create a test collection and insert some data into it. We will create two objects, j and t, and then save them in the collection *things*.

```
> j = { name : "mongo" };
{"name" : "mongo"}
> t = { x : 3 };
{ "x" : 3 }

> db.things.save(j);
> db.things.save(t);

> db.things.find();
{ "_id" : ObjectId("4c2209f9f3924d31102bd84a"), "name" : "mongo" }
{ "_id" : ObjectId("4c2209fef3924d31102bd84b"), "x" : 3 }
```

Let's add some more records to this collection:

```
for (var i = 1; i <= 20; i++)

>db.things.save({x : 4, j : i});
 > db.things.find();

{ "_id" : ObjectId("4c2209f9f3924d31102bd84a"), "name" : "mongo" }
 { "_id" : ObjectId("4c2209fef3924d31102bd84b"), "x" : 3 }
 { "_id" : ObjectId("4c220a42f3924d31102bd856"), "x" : 4, "j" : 1 }
{ "_id" : ObjectId("4c220a42f3924d31102bd857"), "x" : 4, "j" : 2 }
 { "_id" : ObjectId("4c220a42f3924d31102bd858"), "x" : 4, "j" : 3 }
 { "_id" : ObjectId("4c220a42f3924d31102bd859"), "x" : 4, "j" : 4 }
{"_id" : ObjectId("4c220a42f3924d31102bd85a"), "x" : 4, "j" : 5 }
 { "_id" : ObjectId("4c220a42f3924d31102bd85b"), "x" : 4, "j" : 6 }
 { "_id" : ObjectId("4c220a42f3924d31102bd85c"), "x" : 4, "j" : 7 }
```

Let's add some more records to this collection:

```
{ "_id" :ObjectId("4c220a42f3924d31102bd85d"), "x" : 4, "j" : 8 }
{ "_id" : ObjectId("4c220a42f3924d31102bd85e"), "x" : 4, "j" : 9 }
{ "_id" : ObjectId("4c220a42f3924d31102bd85f"), "x" : 4, "j" : 10 }
{ "_id" : ObjectId("4c220a42f3924d31102bd860"), "x" : 4, "j" : 11 }
{ "_id" : ObjectId("4c220a42f3924d31102bd861"), "x" : 4, "j" : 12 }
{ "_id" : ObjectId("4c220a42f3924d31102bd862"), "x" : 4, "j" : 13 }
{ "_id" : ObjectId("4c220a42f3924d31102bd863"), "x" : 4, "j" : 14 }
{ "_id" : ObjectId("4c220a42f3924d31102bd864"), "x" : 4, "j" : 15 }
{ "_id" : ObjectId("4c220a42f3924d31102bd865"), "x" : 4, "j" : 16 }
{ "_id" : ObjectId("4c220a42f3924d31102bd866"), "x" : 4, "j" : 17 }
{ "_id" : ObjectId("4c220a42f3924d31102bd867"), "x" : 4, "j" : 18 }
```

Note that not all documents were shown –

Shell limits the number to 20 when automatically iterating a cursor.

Design Session

Noticeable points

- No setup, No create table
- No predefine collection ,automatically created on the first insert
- Upon being inserted into the database, objects are assigned an object ID (if they do not already have one) in the field _id.

Design Session

Accessing data from query

Before we discuss queries in any depth, lets talk about how to work with the results of a query - a cursor object.

find() query method, which returns everything in a collection.

```
> var cursor = db.things.find();
> while (cursor.hasNext()) printjson(cursor.next());
{ "_id" : ObjectId("4c2209f9f3924d31102bd84a"), "name" : "mongo" }
{ "_id" : ObjectId("4c2209fef3924d31102bd84b"), "x" : 3 }
{ "_id" : ObjectId("4c220a42f3924d31102bd856"), "x" : 4, "j" : 1 }
```

Design Session

Accessing data from query

In the case of a forEach() we must define a function that is called for each document in the cursor.

Convert the cursor to a true array

```
var arr = db.things.find().toArray();
>arr[5];
>{ "_id" : ObjectId("4c220a42f3924d31102bd859"), "x" : 4, "j" : 4 }
```

Query language vs MongoDB

SELECT * FROM things WHERE name="mongo“

```
>db.things.find({name:"mongo"}).forEach(printjson);
>{ "_id" : ObjectId("4c2209f9f3924d31102bd84a"), "name" : "mongo" }
```

Secondary Index

Create index on any Field in Document

- // 1 means ascending ,-1 means descending
- The ensureIndex() function only creates the index if it does not exist.
- In the shell, you can create an index by calling the ensureIndex() function, and providing a document that specifies one or more keys to index.
- db.posts.ensureIndex({author: 1})
- Db.posts.find({author:’’roger})
- {

_id : ObjectId(“4c2209f9f3924d31102bd84a”),

“Name”

.}

Query operators

Conditional operates

nenin,$mod,$all,$exiss,@ne,$ge

// find the post with tags

db.posts.find({tags:{exists:true]}})

- **Regular expressions:**

//post where author starts with s

db.posts.find({author:/^s*/i})

- **Counting:**

//posts written by roger

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

Extending the Schema

Adding More Comments to Blog

➢ {

_id : ObjectId(“4c2209f9f3924d31102bd84a”),

Author : “roger”,

Date : “Sat Jul 24 2010 19:47:11 GMT-0“

Text:”Spirited away”

tags:[“Peace” , ”Love”]

Comments:[

{

Author:”Austrin Powers”,

Date: “Sat Jul 24 2010 19:47:11 GMT-0“

Text:”Yeah baby”}]}

Extending the Schema

Adding Comments to Blog

```
Comment ={author:”Saira Shaheen”,
                date:new Date();
                text:”Yeah baby”
}
Update = { ‘$push ‘:{comment : Comment}}
>db.posts.find({_id:”….”},Update)
```

```
{ $push : { field : value, field2 : value2 } }
```

Secondary Index

// create index on nested document

>db.posts.esureIndex({“comments.author”:1})

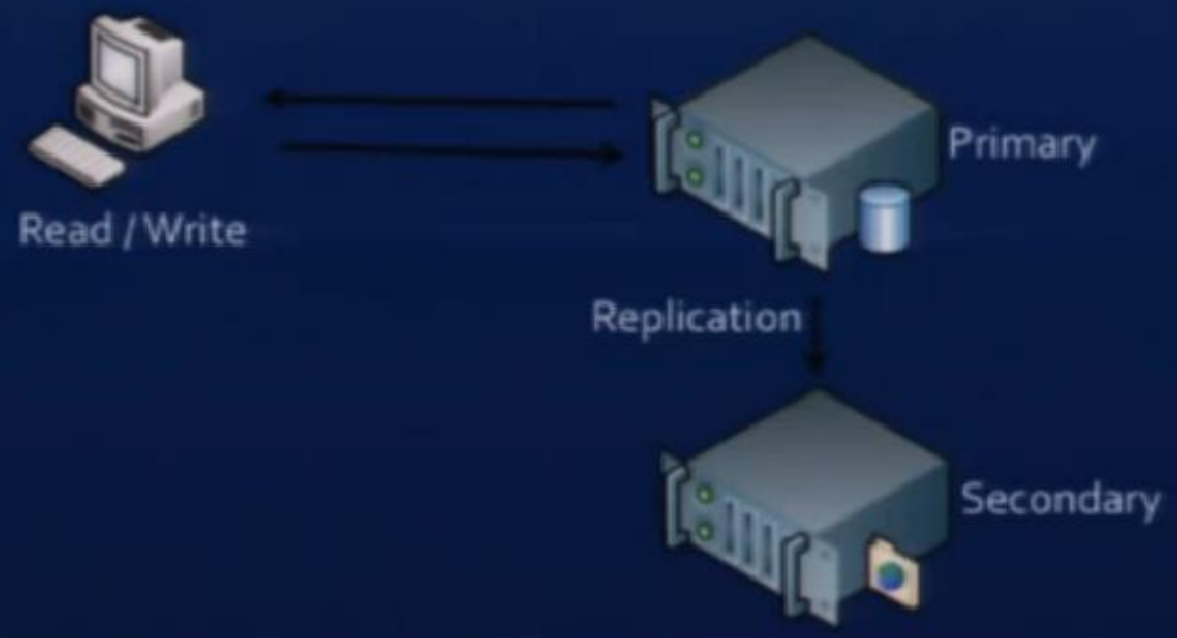
>db.posts.find({comments.author:”Austin Powers”})

Deploying MongoDB

Deploying MongoDB

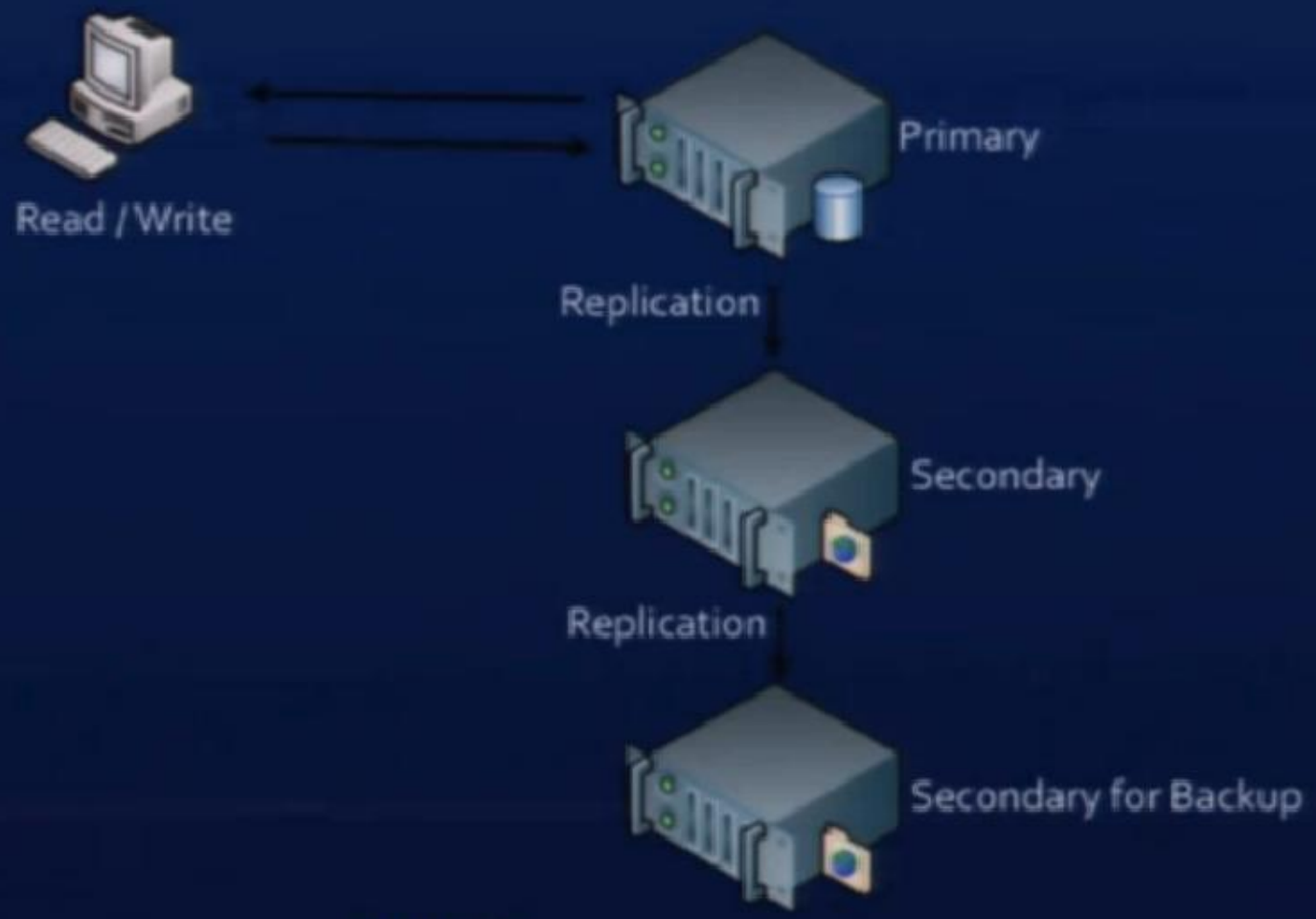
10gen mongoDB

Deploying MongoDB



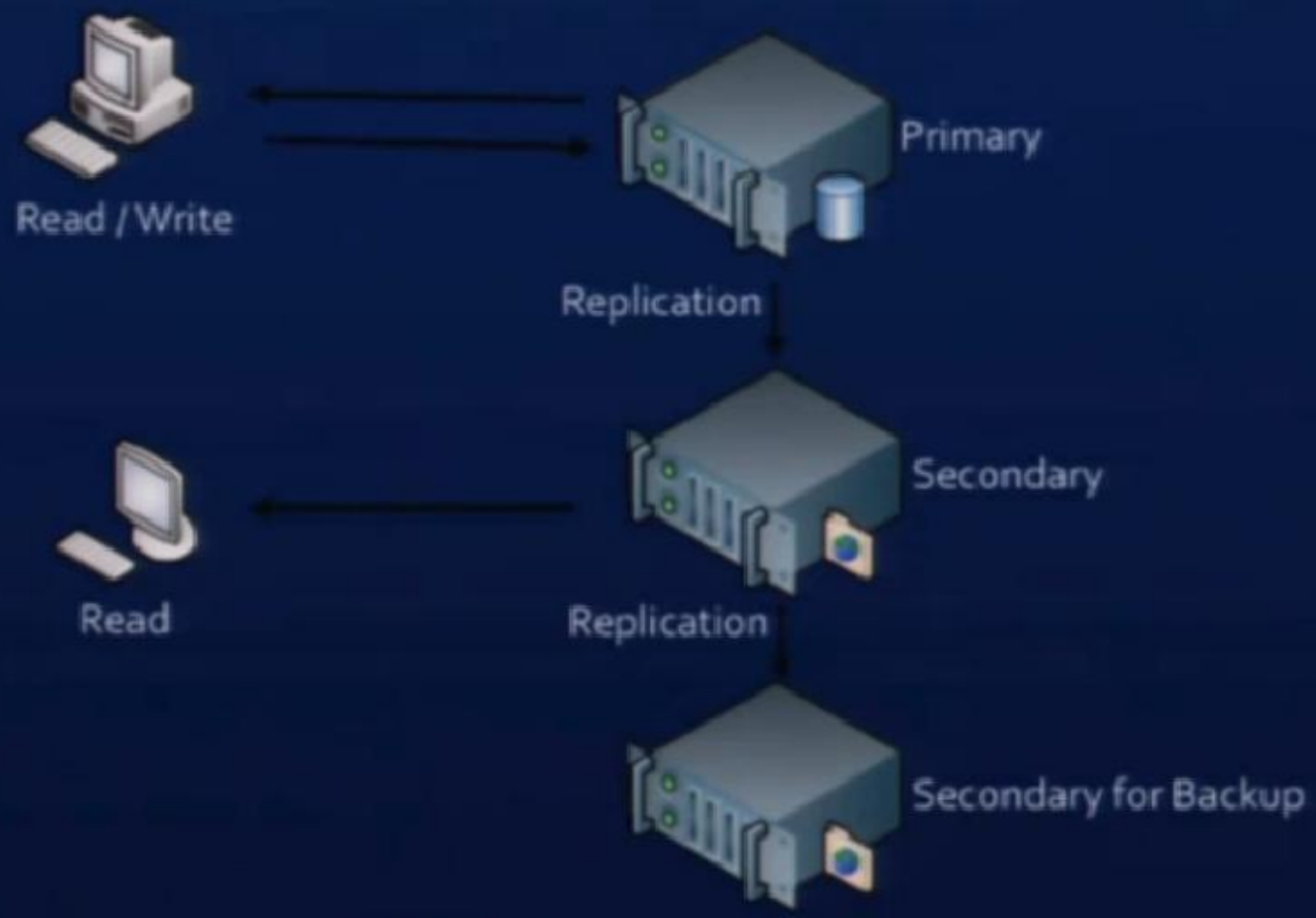
10gen mongoDB

Setting Up MongoDB



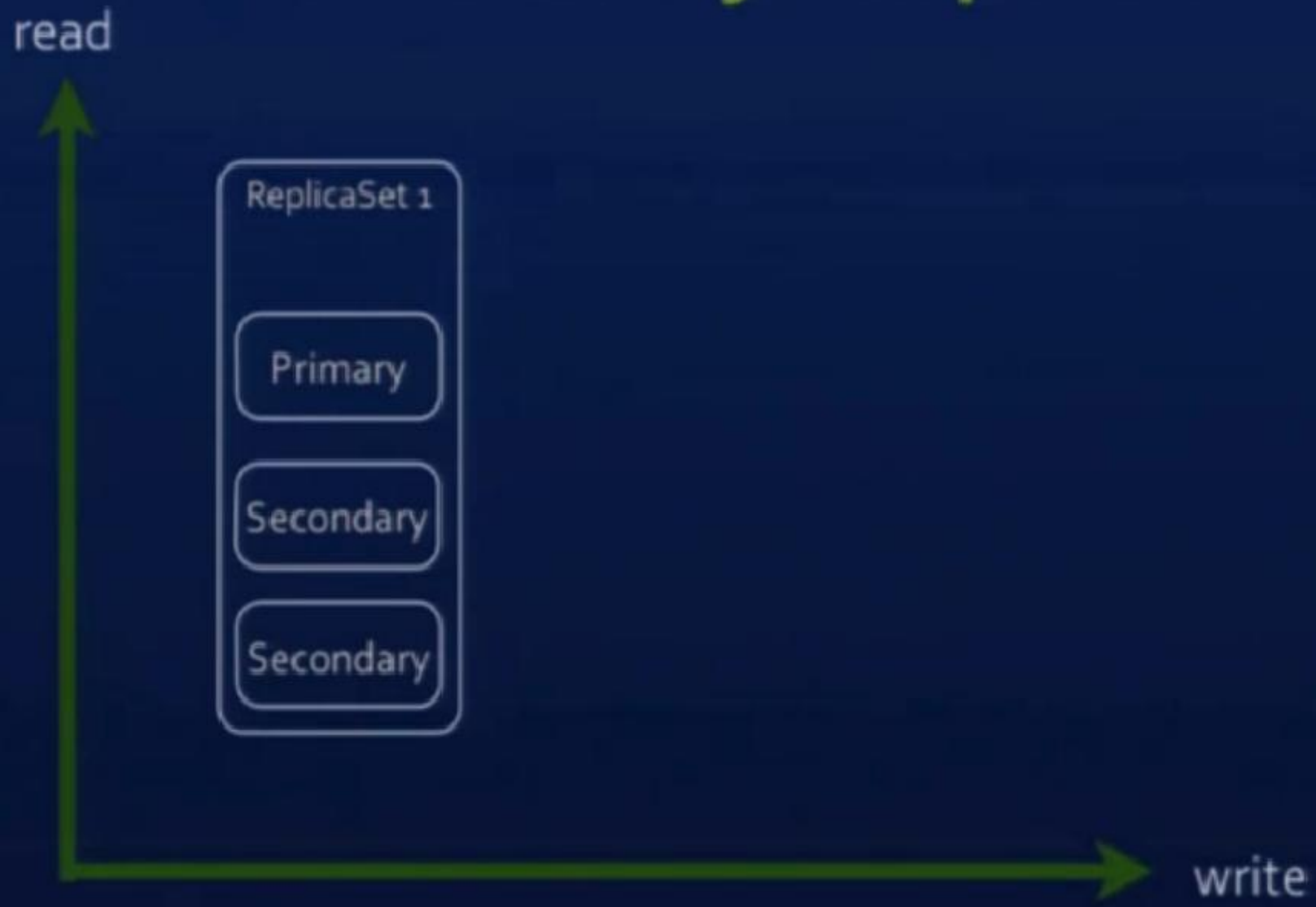
10gen mongoDB

Setting Up MongoDB

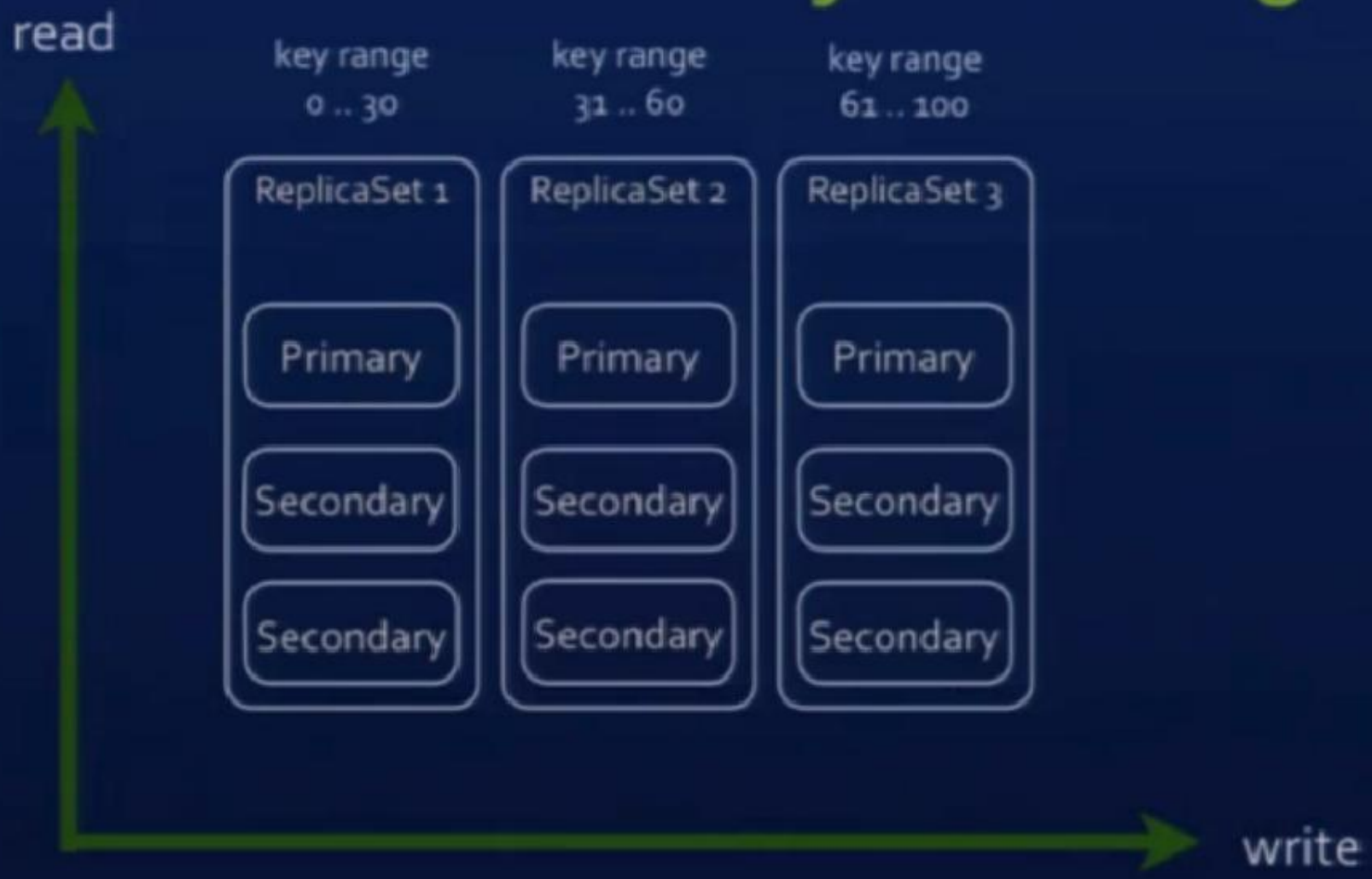


10gen mongoDB

Read Scalability : Replication

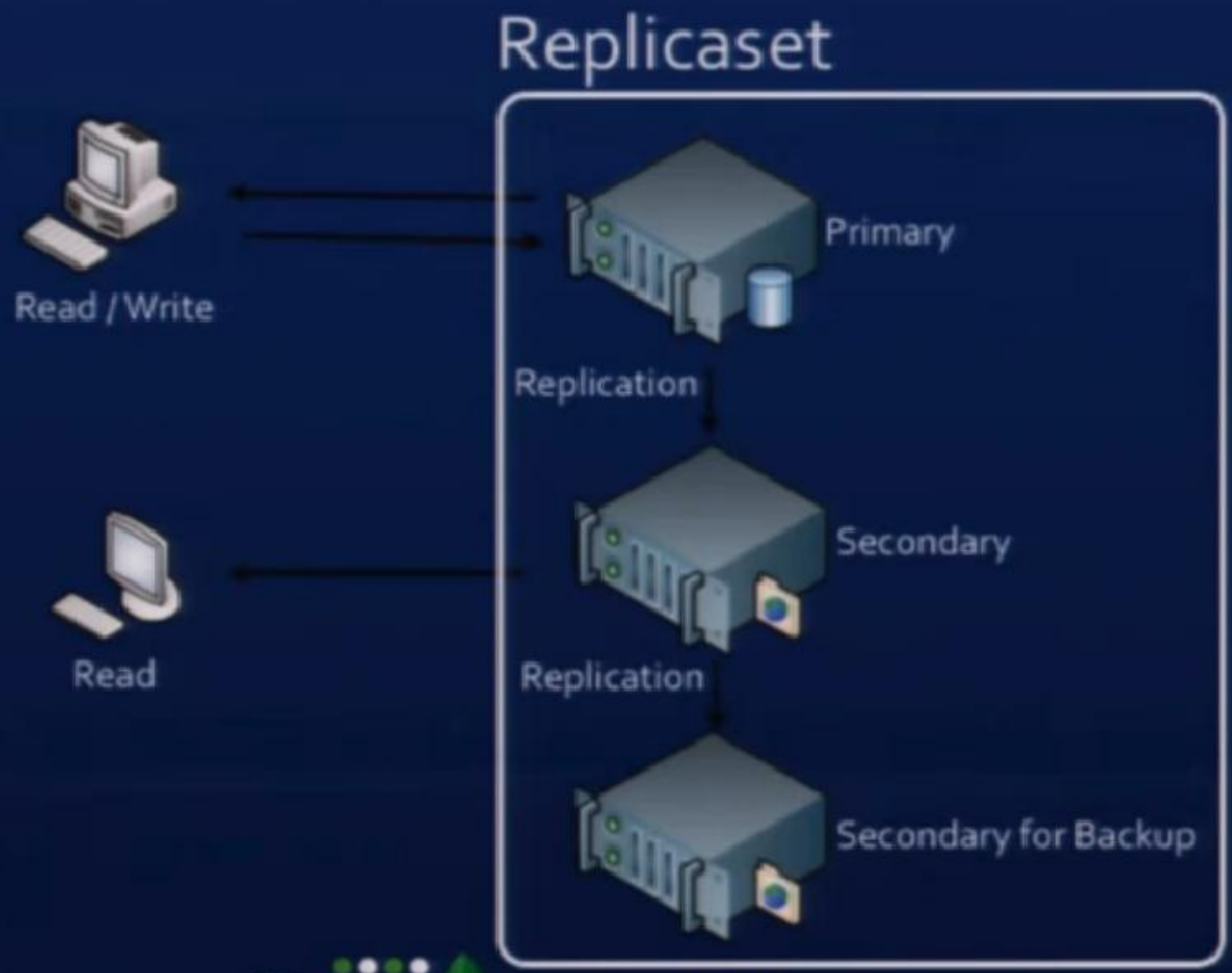


Write Scalability: Sharding



10gen mongoDB

Setting Up MongoDB



10gen mongoDB

Advantages

MongoDB makes building applications easy

Map / Reduce
Capped Collections
Tail-able Cursors
Geo Indexing

.. and much more ! ..

- **Map/Reduce**.
  - Mechanism to aggregate all dataset
  - input coming from a collection and output going to a collection. Often, in a situation where you would have used GROUP BY in SQL, map/reduce is the right tool in MongoDB.
- **Capped Collection**
  - Capped collections automatically, with high performance, maintain insertion order for the documents in the collection; this is very powerful for certain use cases such as logging.
  - Circular buffer, If you want last hour log information, Truncating the record out of it
- **Tiled able cursors**
  - Listen on result set ;as the changes happens ,you can propagate with those changes .You can publish them
- **Geo Indexing**
  - Querying the data and data points that are near to your location

Drawbacks

- MongoDB's primary disadvantage is that it's not nearly as battle-tested or mature as MySQL.
- Very unreliable.
- No single server durability. If something crashes while it's updating 'table-contents' - you loose all you data.
- Repair takes a lot of time, but usually ends up in 50-90% data loss if you aren't lucky. So only way to be fully secure is to have 2 replicas in different datacentres.
- Indexes take up a lot of RAM.

Summary

- Open source
- Document-oriented database
- JSON-like documents with dynamic schemas
- High performance
- High availability
- Easy scalability
- Rich query language
- Disadvantage are unreliable ,not mature, Indexes take up a lot of RAM
- Companies large and small rely on MongoDB for storage of their mission critical data

Thanks for being patience!!!!