



# MongoDB

Open-source, High-performance,  
Schema-free, Document-Oriented Database

# Who Use MongoDB

Justin.tv



Boxed Ice

sourceforge



github  
SOCIAL CODING

BUSINESS  
INSIDER



# MongoDB & CAP Principle

## Visual Guide to NoSQL Systems



# MongoDB features

- Collection storage;
- Dynamic query;
- Complete index of support;
- Query monitoring, Query optimization;
- Replication automatic failover;
- Support binary data and large objects;
- Auto-sharding Support cloud level of flexibility;

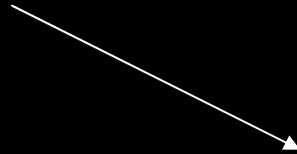
# RDBMS

- Great for many applications
- Shortcomings
- Scalability
- Flexibility

# Scalability & performance

Front  
User

•Memcached



•Key/value  
Stores



•MongoDB



•RDBMS

Back  
end

# JSON-style Documents

## Example

```
{ title : "My First Post", author: "javablogger",  
  comments : [{ by: "Abe", text: "First" },  
              { by : "Ada", text : "Good post" } ]  
}
```

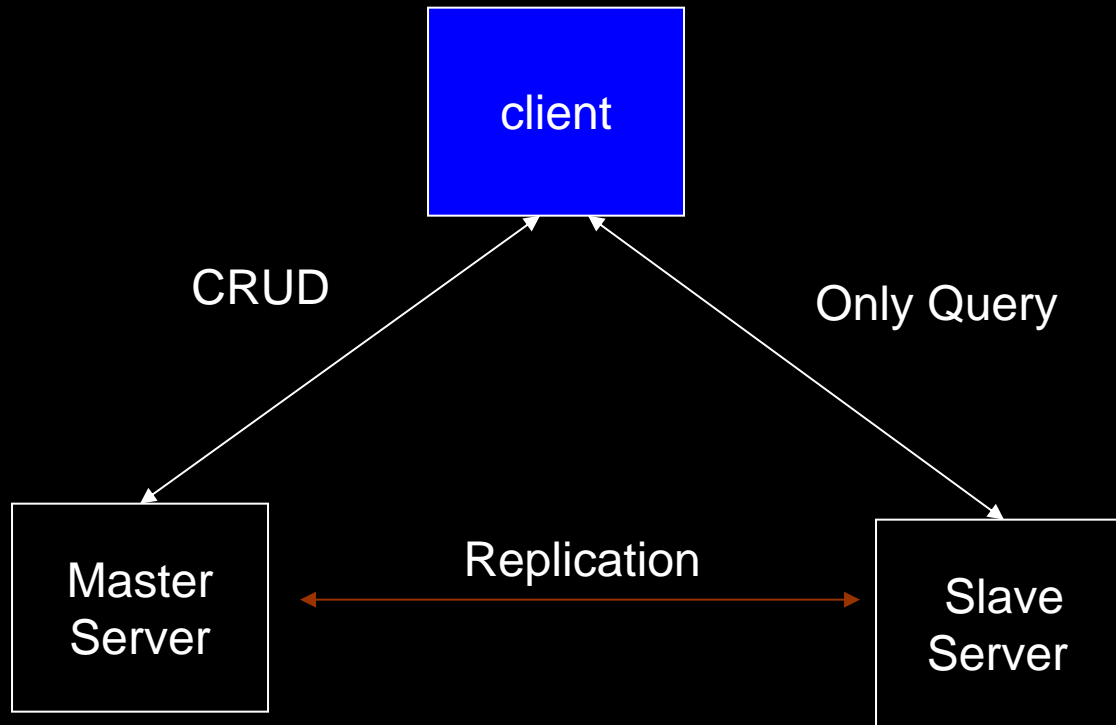
# Schema-free

- Loosening constraints - added flexibility
- Dynamically typed languages (like Ruby!)
- Migrations

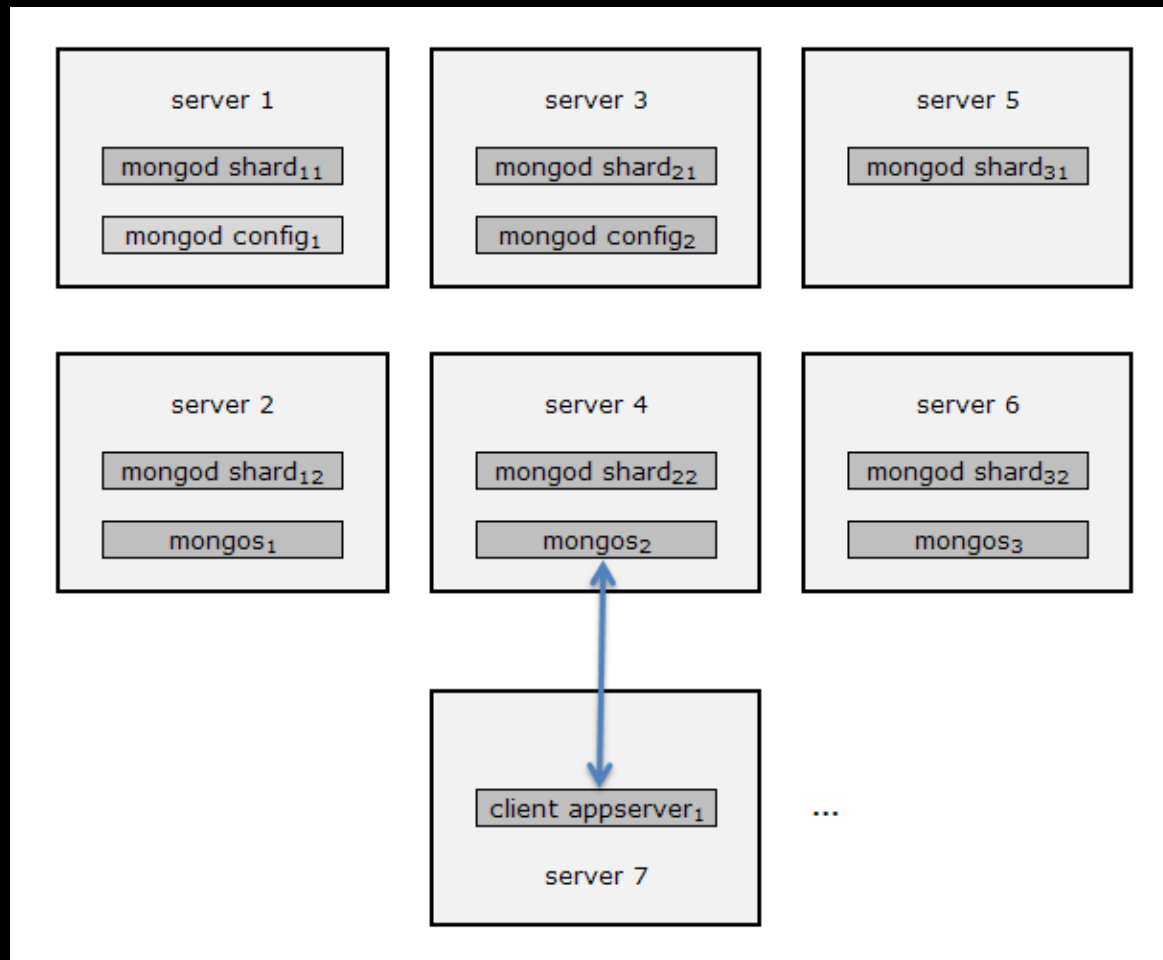
# Dynamic queries

- Administration
- Ease of development
- Familiarity

# Replication



# Auto-sharding



# MapReduce

## Command syntax:

```
db.runCommand(  
  { mapreduce : <collection>,  
    map : <mapfunction>,  
    reduce : <reducefunction>  
  [, query : <query filter object>]  
  [, sort : <sort the query. useful for optimization>]  
  [, limit : <number of objects to return from collection>]  
  [, out : <output-collection name>]  
  [, keeptemp: <true|false>]  
  [, finalize : <finalizelfunction>]  
  [, scope : <object where fields go into javascript global scope >]  
  [, verbose : true] } );
```

# Many Supported

- Platforms

Windows

Linux

Unix

bsd

- Languages

Java

C++

Ruby

PHP

- Docs

Administrative

Developer

Online API

# Good at

- The web
- Caching
- High volume data
- Scalability

# Less good at

- Highly transactional
- Ad-hoc business intelligence
- Problems that require SQL

# MongoDB Basics

# Document

- Unit of storage (think row)
- BSON (Binary JSON)
- Represented as a Hash

# Collection

- Schema-free equivalent of a table
- Logical groups of documents
- Indexes are per-collection

# `_id`

- Special key
- Present in all documents
- Unique across a Collection
- Any type you want

Blog back-end

# Post

```
{:author => "mike",  
 :date => Time.new,  
 :text => "my blog post",  
 :tags => ["mongodb", "ruby"]}
```

# Comment

```
{:author => "eliot",  
 :date => Time.new,  
 :text => "great post!"}
```

# New post

```
post = { :author => "mike",  
         :date => Time.new,  
         :text => "my blog post",  
         :tags => ["mongodb", "ruby"] }
```

```
db["posts"].save(post)
```

# Embedding a comment

```
c = { :author => "eliot",  
      :date => Time.new,  
      :text => "great post!" }
```

```
db["posts"].update({ :_id => post[:_id] },  
                  { :$push => { :comments => c } })
```

# Posts by author

```
db["posts"].find(:author => "mike")
```

# Last 10 posts

```
db["posts"].find.sort([[:date, :desc]])  
.limit(10)
```

# Posts in the last week

```
last_week = Time.utc(2009, 11, 12)  
db["posts"].find(:date => {:$gt => last_week})
```

Posts ending with  
'Ruby'

```
db["posts"].find(:text => /Ruby$/)
```

# Posts with a tag

```
db["posts"].find(:tags => "mongodb")
```

... and fast

```
db["posts"].create_index("tags")
```

# Counting posts

```
db["posts"].count
```

```
db["posts"].find(:author => "mike").count
```

# Basic paging

```
page = 2
```

```
page_size = 15
```

```
db["posts"].find.limit(page_size)  
                    .skip(page * page_size)
```

# Migration: adding titles

- Easy - just start adding them:

```
post = {:author => "mike",  
       :date => Time.new,  
       :text => "another blog post",  
       :tags => ["RubyConf"],  
       :title => "Review from RubyConf"}
```

```
post_id = db["posts"].save(post)
```

# Advanced queries

- \$gt, \$lt, \$gte, \$lte, \$ne, \$all, \$in, \$nin
- \$where

```
db["posts"].find :$where => "this.author == 'mike' ||  
                             this.title == 'hello'")
```

MongoMapper, Mongoid,  
MongoRecord, etc.



# MongoMapper

```
class User
  include MongoMapper::Document
  many :posts
end
```

```
class Post
  include MongoMapper::Document
  key :user_id, String
  key :title, String
end
```

```
user = User.create
user.posts.create(:title => 'Foo')
```

```
# would return post we just created
user.posts.find_by_title('Foo')
```

# Other cool stuff

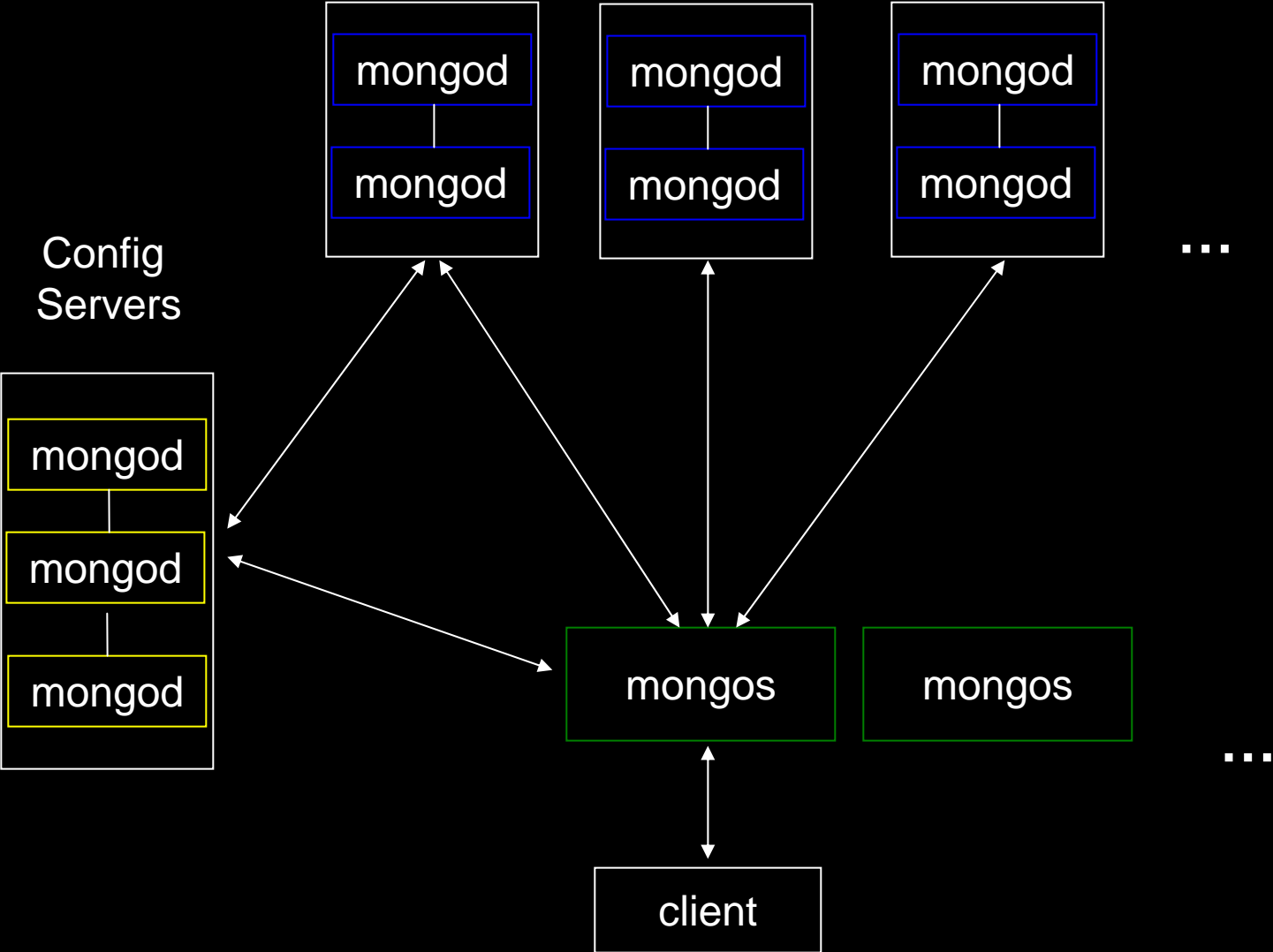
- Aggregation and map reduce
- Capped collections
- Unique indexes
- Mongo shell
- GridFS

# Sharding

# Terminology

- Shard key
- Chunk
  - Range of the value space
  - (collection, key, min\_val, max\_val)
- Shard
  - Single node (or replica pair)
  - Responsible for set of chunks

# Shards



- **Download MongoDB**  
**<http://www.mongodb.org>**
- **Try it out**
- **Let us know what you think!**

- <http://www.mongodb.org>
- <http://www.javabloger.com>
- [njthnet@javabloger.com](mailto:njthnet@javabloger.com)