mongoDB

Python + MongoDB

Or:

How I Learned To Stop Joining

and

Love The Document

Brendan McAdams

10gen, Inc.

brendan@10gen.com

@rit

What Is MongoDB?

10gen mongoDB

So We've Built an Application with a Database

10gen mongoDB

So We've Built an Application with a Database

How do we integrate that database with our application's object hierarchy?

10gen mongoDB

I know! Let’s use an ORM!

10gen mongoDB

I know! Let’s use an ORM!

Congratulations: Now we’ve got 2 problems!
(or is it n+1?)

10gen mongoDB

Let’s Face It ...

SQL Sucks.

10gen mongoDB

Let’s Face It ...

SQL Sucks.

For *some* problems at least.

10gen mongoDB

Stuffing an object graph into a relational model is like fitting a square peg into a round hole.

10gen mongoDB

Sure, we can use an ORM. But who are we really fooling?

10gen mongoDB

Sure, we can use an ORM. But who are we really fooling?

... and who/what are we going to wake up next to in the morning?

10gen mongoDB

This is a SQL Model

```
mysql> select * from book;
+----+-------------------------------------------------------------+
| id | title                                                       |
+----+-------------------------------------------------------------+
|  1 | The Demon-Haunted World: Science as a Candle in the Dark |
|  2 | Cosmos                                                      |
|  3 | Programming in Scala                                        |
+----+-------------------------------------------------------------+
3 rows in set (0.00 sec)

mysql> select * from bookauthor;
+---------+-----------+
| book_id | author_id |
+---------+-----------+
|       1 |         1 |
|       2 |         1 |
|       3 |         2 |
|       3 |         3 |
|       3 |         4 |
+---------+-----------+
5 rows in set (0.00 sec)

mysql> select * from author;
+----+-----------+------------+-------------+-------------+---------------+
| id | last_name | first_name | middle_name | nationality | year_of_birth |
+----+-----------+------------+-------------+-------------+---------------+
|  1 | Sagan     | Carl       | Edward      | NULL        | 1934          |
|  2 | Odersky   | Martin     | NULL        | DE          | 1958          |
|  3 | Spoon     | Lex        | NULL        | NULL        | NULL          |
|  4 | Venners   | Bill       | NULL        | NULL        | NULL          |
+----+-----------+------------+-------------+-------------+---------------+
4 rows in set (0.00 sec)
```

10gen mongoDB

Joins are great and all ...

10gen mongoDB

Joins are great and all ...

- Potentially organizationally messy

10gen mongoDB

Joins are great and all ...

- Potentially organizationally messy
- Structure of a single object is NOT immediately clear to someone glancing at the shell data

10gen mongoDB

Joins are great and all ...

- Potentially organizationally messy
- Structure of a single object is NOT immediately clear to someone glancing at the shell data
- We have to flatten our object out into *three* tables

10gen mongoDB

Joins are great and all ...

- Potentially organizationally messy
- Structure of a single object is NOT immediately clear to someone glancing at the shell data
- We have to flatten our object out into *three* tables
  - **7** separate inserts just to add “Programming in Scala”

10gen mongoDB

Joins are great and all ...

- Potentially organizationally messy
- Structure of a single object is NOT immediately clear to someone glancing at the shell data
- We have to flatten our object out into *three* tables
  - **7** separate inserts just to add “Programming in Scala”
- Once we turn the relational data back into objects ...

10gen mongoDB

Joins are great and all ...

- Potentially organizationally messy
- Structure of a single object is NOT immediately clear to someone glancing at the shell data
- We have to flatten our object out into *three* tables
  - **7** separate inserts just to add “Programming in Scala”
- Once we turn the relational data back into objects ...
- We still need to convert it to data for our frontend

10gen mongoDB

Joins are great and all ...

- Potentially organizationally messy
- Structure of a single object is NOT immediately clear to someone glancing at the shell data
- We have to flatten our object out into *three* tables
    - **7** separate inserts just to add “Programming in Scala”
- Once we turn the relational data back into objects ...
- We still need to convert it to data for our frontend
- I don’t know about you, but I have better things to do with my time.

10gen mongoDB

The Same Data in MongoDB

```
> db.books.find().forEach(printjson)
{
    "_id" : ObjectId("4dfa6baa9c65dae09a4bbda3"),
    "title" : "The Demon-Haunted World: Science as a Candle in the Dark",
    "author" : [
        {
            "first_name" : "Carl",
            "last_name" : "Sagan",
            "middle_name" : "Edward",
            "year_of_birth" : 1934
        }
    ]
}
{
    "_id" : ObjectId("4dfa6baa9c65dae09a4bbda4"),
    "title" : "Cosmos",
    "author" : [
        {
            "first_name" : "Carl",
            "last_name" : "Sagan",
            "middle_name" : "Edward",
            "year_of_birth" : 1934
        }
    ]
}
```

10gen mongoDB

The Same Data in MongoDB
(Part 2)

```
{
    "_id" : ObjectId("4dfa6baa9c65dae09a4bbda5"),
    "title" : "Programming in Scala",
    "author" : [
        {
            "first_name" : "Martin",
            "last_name" : "Odersky",
            "nationality" : "DE",
            "year_of_birth" : 1958
        },
        {
            "first_name" : "Lex",
            "last_name" : "Spoon"
        },
        {
            "first_name" : "Bill",
            "last_name" : "Venners"
        }
    ]
}
```

10gen mongoDB

Access to the embedded objects is integral

```
> db.books.find({"author.first_name": "Martin", "author.last_name":
"Odersky"})
{ "_id" : ObjectId("4dfa6baa9c65dae09a4bbda5"), "title" : "Programming in
Scala", "author" : [
    {
        "first_name" : "Martin",
        "last_name" : "Odersky",
        "nationality" : "DE",
        "year_of_birth" : 1958
    },
    {
        "first_name" : "Lex",
        "last_name" : "Spoon"
    },
    {
        "first_name" : "Bill",
        "last_name" : "Venners"
    }
] }
```

10gen mongoDB

As is manipulation of the embedded data

```
> db.books.update({"author.first_name": "Bill", "author.last_name": "Venners"},
...                {$set: {"author.$.company": "Artima, Inc."}})
> db.books.update({"author.first_name": "Martin", "author.last_name": "Odersky"},
...                {$set: {"author.$.company": "Typesafe, Inc."}})
> db.books.findOne({"title": /Scala$/})
{
    "_id" : ObjectId("4dfa6baa9c65dae09a4bbda5"),
    "author" : [
        {
            "company" : "Typesafe, Inc.",
            "first_name" : "Martin",
            "last_name" : "Odersky",
            "nationality" : "DE",
            "year_of_birth" : 1958
        },
        {
            "first_name" : "Lex",
            "last_name" : "Spoon"
        },
        {
            "company" : "Artima, Inc.",
            "first_name" : "Bill",
            "last_name" : "Venners"
        }
    ],
    "title" : "Programming in Scala"
}
```

10gen mongoDB

NoSQL Really Means...

non-relational, next-generation
operational datastores and databases

10gen mongoDB

NoSQL Really Means...

non-relational, next-generation
operational datastores and databases

... Let's focus on the "non-relational" bit.

10gen mongoDB

no joins
+ no complex transactions

Horizontally Scalable Architectures

10gen mongoDB

no joins
+ no complex transactions

10gen mongoDB

no joins
+ no complex transactions

10gen mongoDB

no joins
+ no complex transactions

New Data Models

10gen mongoDB

Best Use Cases

"Scaling Out"

Caching

Web Applications

High Volume Traffic

10gen mongoDB

Less Suited For

highly transactional applications

ad-hoc business intelligence

problems which require SQL

10gen mongoDB

Memory

- MongoDB revolves around memory mapped files

10gen mongoDB

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, a page fault occurs (Expensive + Locking!)
- OS goes to disk to fetch the data
- Compare this to the normal trick of sticking a poorly managed memcached cluster in front of MySQL

10gen mongoDB

A Few Words on OS Choice

- For production: Use a 64 bit OS and a 64 bit MongoDB Build
- 32 Bit has a 2 gig limit; imposed by the operating systems for memory mapped files
- Clients can be 32 bit
- MongoDB Supports (little endian only)
- Linux, FreeBSD, OS X (on Intel, not PowerPC)
- Windows
- Solaris (Intel only, Joyent offers a cloud service which works for Mongo)

10gen mongoDB

_id

if not specified drivers will add default:

```
ObjectId("4bface1a2231316e04f3c434")
```

timestamp

machine_id

process id

counter

http://www.mongodb.org/display/DOCS/Object+IDs

10gen mongoDB

BSON Encoding

```
{ _id: ObjectId(XXXXXXXXXXXX),
  hello: “world”}
```

```
\x27\x00\x00\x00\x07_id\x00
X X X X X X X X X X X X X X
\x02 h e l l o \x00\x06\x00
\x00\x00 w o r l d \x00\x00
```

http://bsonspec.org

10gen mongoDB

bsonspec.org

http://bsonspec.org/

Mail | Book | Gmail | MongoDB | JIRA | Yarddoc | View RSS Feed | Bookmark on Delicio | My Delicious | Other Bookm

BSON { 01010100 11101011 10101110 01010101 }

BSON *[bee · sahn]*, short for Binary JSON, is a binary-encoded serialization of JSON-like documents. Like JSON, BSON supports the embedding of documents and arrays within other documents and arrays. BSON also contains extensions that allow representation of data types that are not part of the JSON spec. For example, BSON has a Date type and a BinData type.

BSON can be compared to binary interchange formats, like Protocol Buffers. BSON is more "schema-less" than Protocol Buffers, which can give it an advantage in flexibility but also a slight disadvantage in space efficiency (BSON has overhead for field names within the serialized data).

BSON was designed to have the following three characteristics:

1. **Lightweight**

   Keeping spatial overhead to a minimum is important for any data representation format, especially when used over the network.

2. **Traversable**

   BSON is designed to be traversed easily. This is a vital property in its role as the primary data representation for MongoDB.

3. **Efficient**

   Encoding data to BSON and decoding from BSON can be performed very quickly in most languages due to the use of C data types.

specification | implementation | discussion

10gen mongoDB

MongoDB + Python

... and some cool MongoDB Features

10gen mongoDB

Introducing pymongo

- “pymongo”: The official Python driver for MongoDB
- Provides an optional C extension for performant BSON; pure Python fallback code
  - C extension needs a little endian system
  - A few System Packages needed
    - GCC (to compile)
    - Python “dev” package to provide `Python.h`

10gen mongoDB

Introducing pymongo

- pymongo is available on PyPi
  - `sudo easy_install pymongo`
  - `sudo pip install pymongo`
- Or, you can build from Source...
  - `git clone git://github.com/mongodb/mongo-python-driver.gi`
  - `cd mongo-python-driver`
  - `python setup.py install`

10gen mongoDB

MongoDB + Python Map Beautifully

- MongoDB Documents represented as ‘`dict`’
- Arrays as ‘`list`’
- Python types map cleanly to related MongoDB types
  - `datetime.datetime` <-> BSON datetime type, etc
  - You can easily define your own custom type serialization / deserialization

10gen mongoDB

This Document from the Shell...

```
{
    "_id" : ObjectId("4dfa6baa9c65dae09a4bbda5"),
    "title" : "Programming in Scala",
    "author" : [
        {
            "first_name" : "Martin",
            "last_name" : "Odersky",
            "nationality" : "DE",
            "year_of_birth" : 1958
        },
        {
            "first_name" : "Lex",
            "last_name" : "Spoon"
        },
        {
            "first_name" : "Bill",
            "last_name" : "Venners"
        }
    ]
}
```

10gen mongoDB

This Document from the Shell...

10gen mongoDB

...Is Nearly Identical in pymongo

10gen mongoDB

...Is Nearly Identical in pymongo

```
import bson
from bson.objectid import ObjectId

scala_book = {
    "_id" : ObjectId("4dfa6baa9c65dae09a4bbda5"),
    "title" : "Programming in Scala",
    "author" : [
        {
            "first_name" : "Martin",
            "last_name" : "Odersky",
            "nationality" : "DE",
            "year_of_birth" : 1958
        },
        {
            "first_name" : "Lex",
            "last_name" : "Spoon"
        },
        {
            "first_name" : "Bill",
            "last_name" : "Venners"
        }
    ]
}
```

10gen mongoDB

MongoDB Data ‘Plays Nice’ in Python

10gen mongoDB

MongoDB Data ‘Plays Nice’ in Python

```
In [6]: scala_book
Out[6]:
{'_id': ObjectId('4dfa6baa9c65dae09a4bbda5'),
 'author': [{'first_name': 'Martin',
             'last_name': 'Odersky',
             'nationality': 'DE',
             'year_of_birth': 1958},
            {'first_name': 'Lex', 'last_name': 'Spoon'},
            {'first_name': 'Bill', 'last_name': 'Venners'}],
 'title': 'Programming in Scala'}
```

```
In [7]: type(scala_book)
Out[7]: <type 'dict'>

In [10]: scala_book['author']
Out[10]:
[{'first_name': 'Martin',
  'last_name': 'Odersky',
  'nationality': 'DE',
  'year_of_birth': 1958},
 {'first_name': 'Lex', 'last_name': 'Spoon'},
 {'first_name': 'Bill', 'last_name': 'Venners'}]
```

10gen mongoDB

The MongoDB API aims to be Pythonic

```
from pymongo import Connection
mongo = Connection() # default server; equiv to Connection('localhost',
27017)

def print_book(book):
    print "%s by %s" % (book['title'], ', '.join(book['author']))

# Let's find all of the documents in the 'bookstore' databases' "books"
collection
for book in mongo.bookstore.books.find():
    # pymongo Cursors implement __iter__, so they can be iterated
naturally
    print_book(book)

# Or we can find all the books about Python
for book in mongo.bookstore.books.find({"tags": "python"}):
    print_book(book)

# Let's add a "PyconIE" tag to *every* Python book...
mongo.bookstore.books.update({"tags": "python"}, {'$push': 'pyconIE'},
multi=True)
```

10gen mongoDB

The MongoDB API aims to be Pythonic

```
# Finally, we'll create a new book and add it in
fePy_book = {
    'title': "IronPython in Action",
    'author': [
        "Michael J. Foord",
        "Christian Muirhead"
    ],
    'isbn': "978-1933988337",
    'price': {
        'currency': "USD",
        'discount': 29.54,
        'msrp': 44.99
    },
    'publicationYear': 2009,
    'tags': [
        "ironpython",
        "python",
        "programming",
        "dotnet",
        "great projects microsoft unceremoniously murdered",
        "csharp",
        "open source",
        "pyconIE"
    ],
    'publisher': "Manning Publications Co."
}

mongo.bookstore.books.save(fePy_book)
```

10gen mongoDB

Geospatial Indexing

"Where the hell am I?"

- Search by (2D) Geospatial proximity with MongoDB
- One GeoIndex per collection
- Can index on an array or a subdocument
- Searches against the index can treat the dataset as flat (map-like), Spherical (like a globe), and complex (box/rectangle, circles, concave polygons and convex poylgons)

10gen mongoDB

Let's Play With Geospatial

- Loaded all of the NYC Subway data in Google Transit Feed Format (Not many useful feeds in this format for Ireland/UK)
- Quick Python Script to index the "Stops" data

```
connection = Connection()
db = connection['nyct_subway']
print "Indexing the Stops Data."
for row in db.stops.find():
    row['stop_geo'] = {'lat': row['stop_lat'], 'lon': row['stop_lon']}
    db.stops.save(row)

db.stops.ensure_index([('stop_geo', pymongo.GEO2D)])
```

- "stop_geo" field is now Geospatially indexed.
- How hard is it to find the 2 closest subway stops to 10gen HQ?

10gen mongoDB

NYC Subways near 10gen HQ

```
> db.stops.find({stop_geo: { $near: [40.738744, -73.991724] }},_
...                 {'stop_name': 1}).limit(2)
{ "_id" : ObjectId("4d8a1ccbe289ae2897caf508"), "stop_name" : "14
St - Union Sq" }
{ "_id" : ObjectId("4d8a1ccbe289ae2897caf507"), "stop_name" : "23
St" }
```

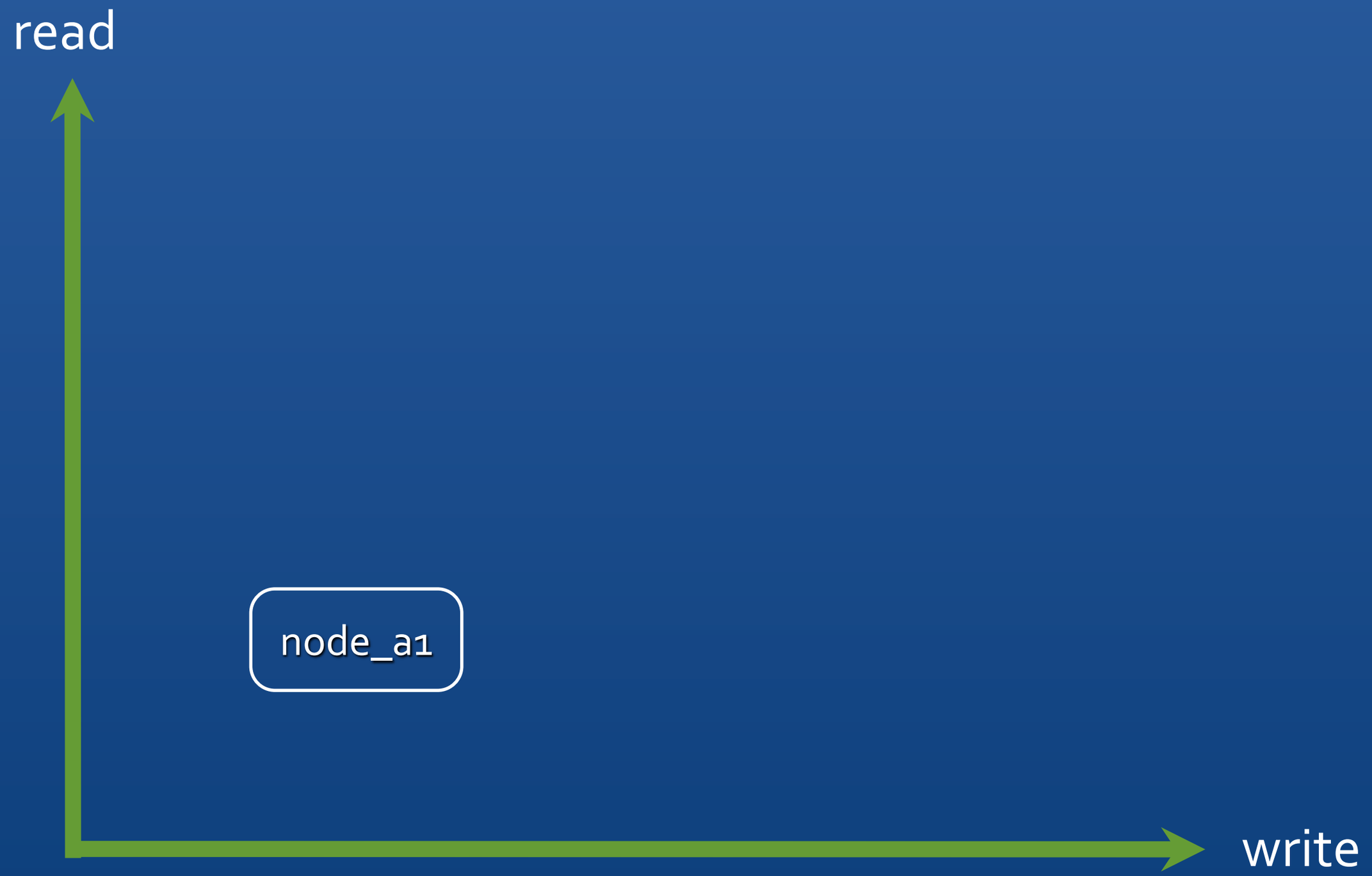
10gen mongoDB

Scalability

- Traditional Master/Slave Replication
  - Much like MySQL
- Replica Sets
  - Clusters of n servers
  - Any one node can be primary
  - Consensus election of primary (> 50% of set up/visible)
  - Automatic failover & recovery
  - All writes to primary
  - Reads can be to primary (default) or a secondary
- Sharding
  - Automatic Partitioning and management
  - Range Based
  - Convert to sharded system with no downtime
  - Fully Consistent

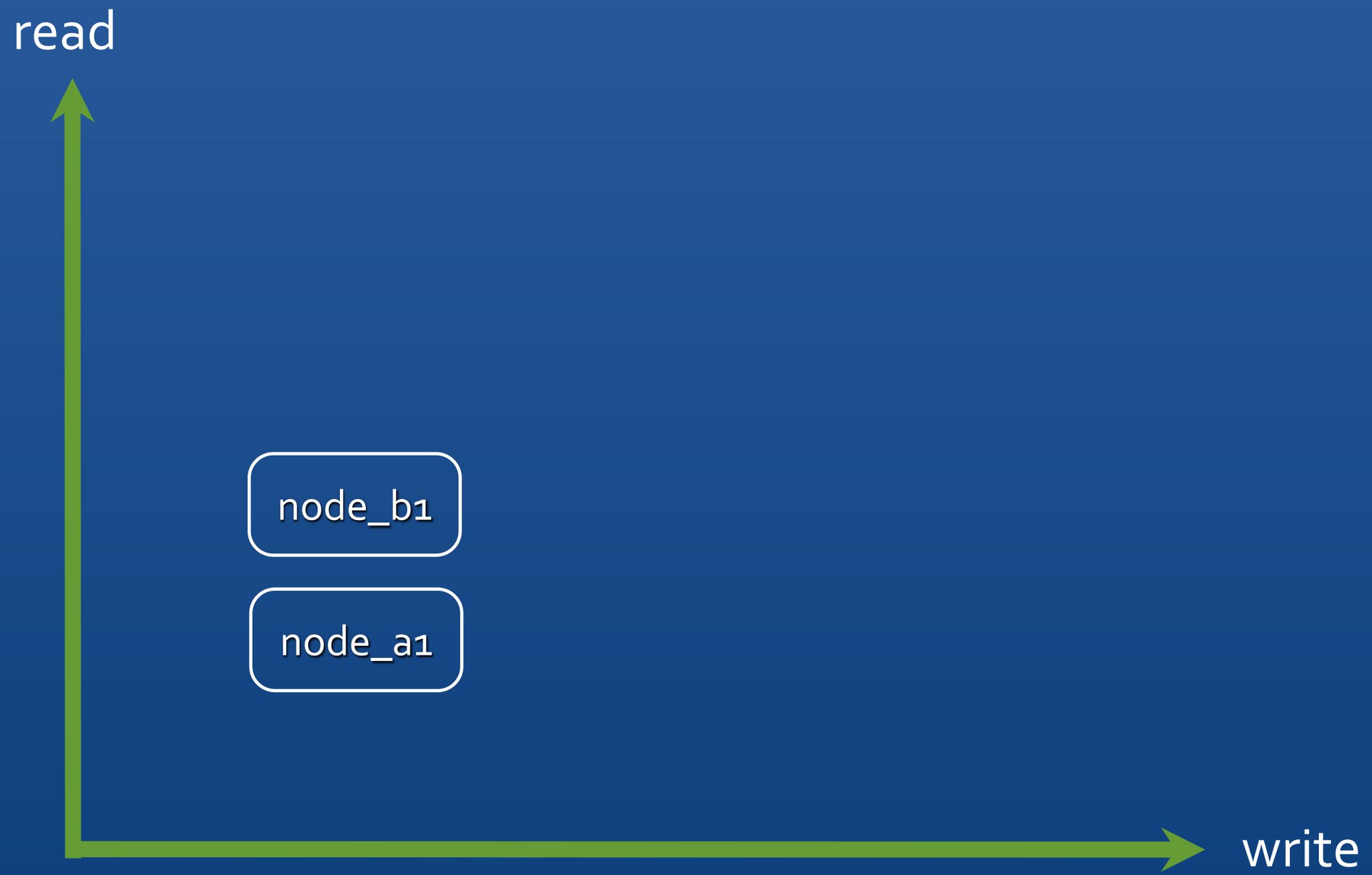
10gen mongoDB

MongoDB Scaling – Single Node



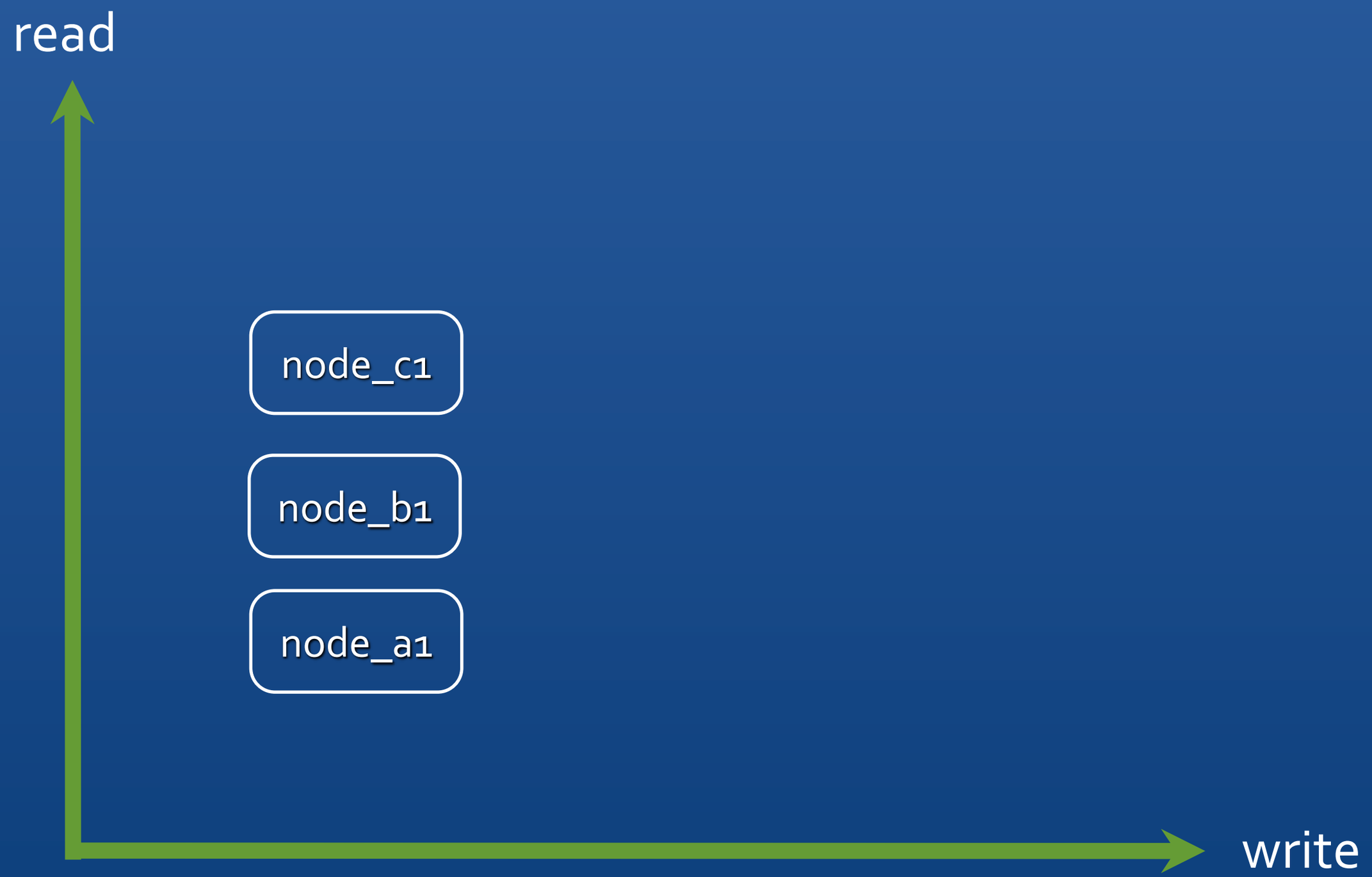
10gen mongoDB

Read scaling – add Replicas



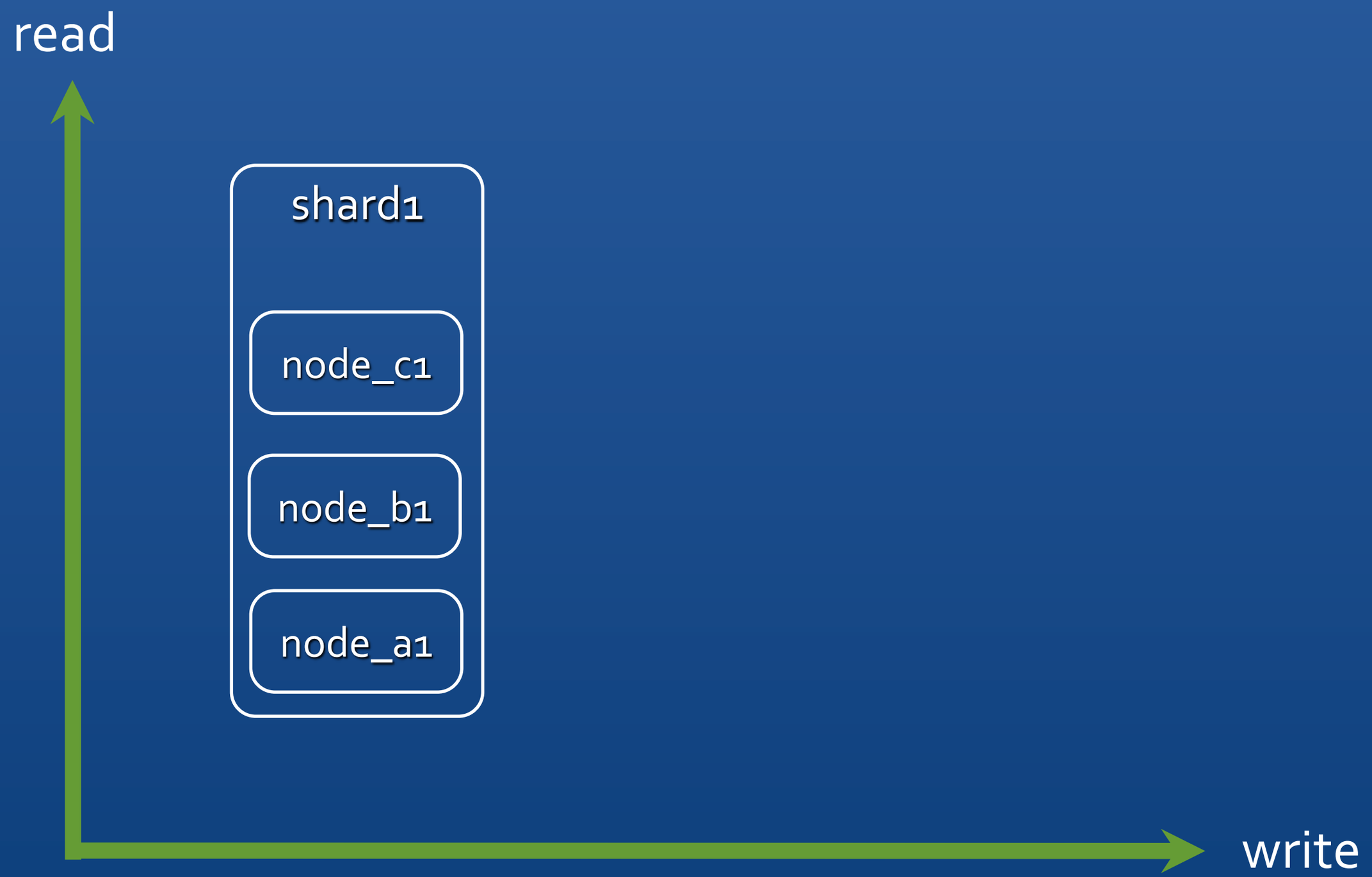
10gen mongoDB

Read scaling – add Replicas



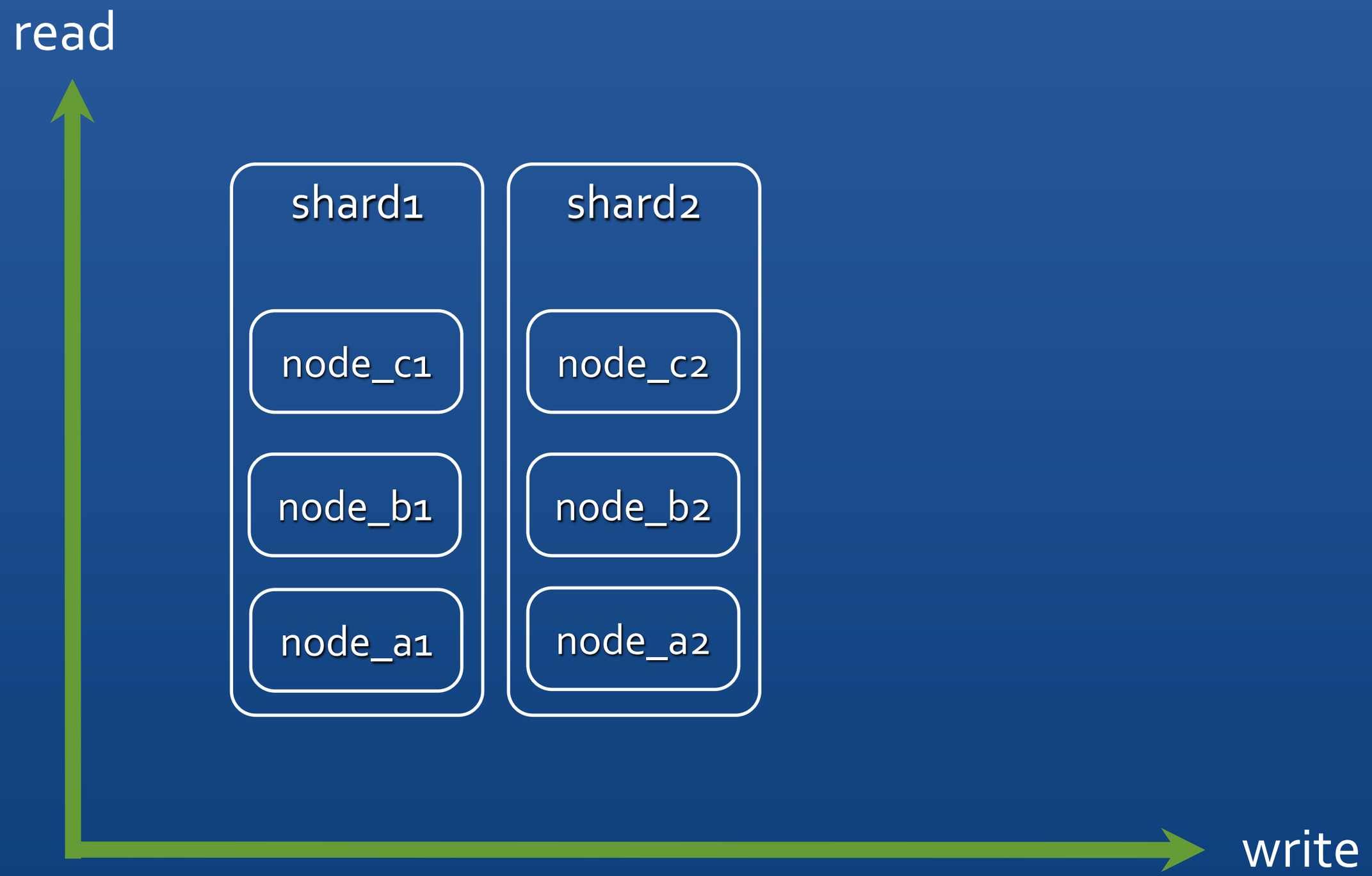
10gen mongoDB

Write scaling – Sharding



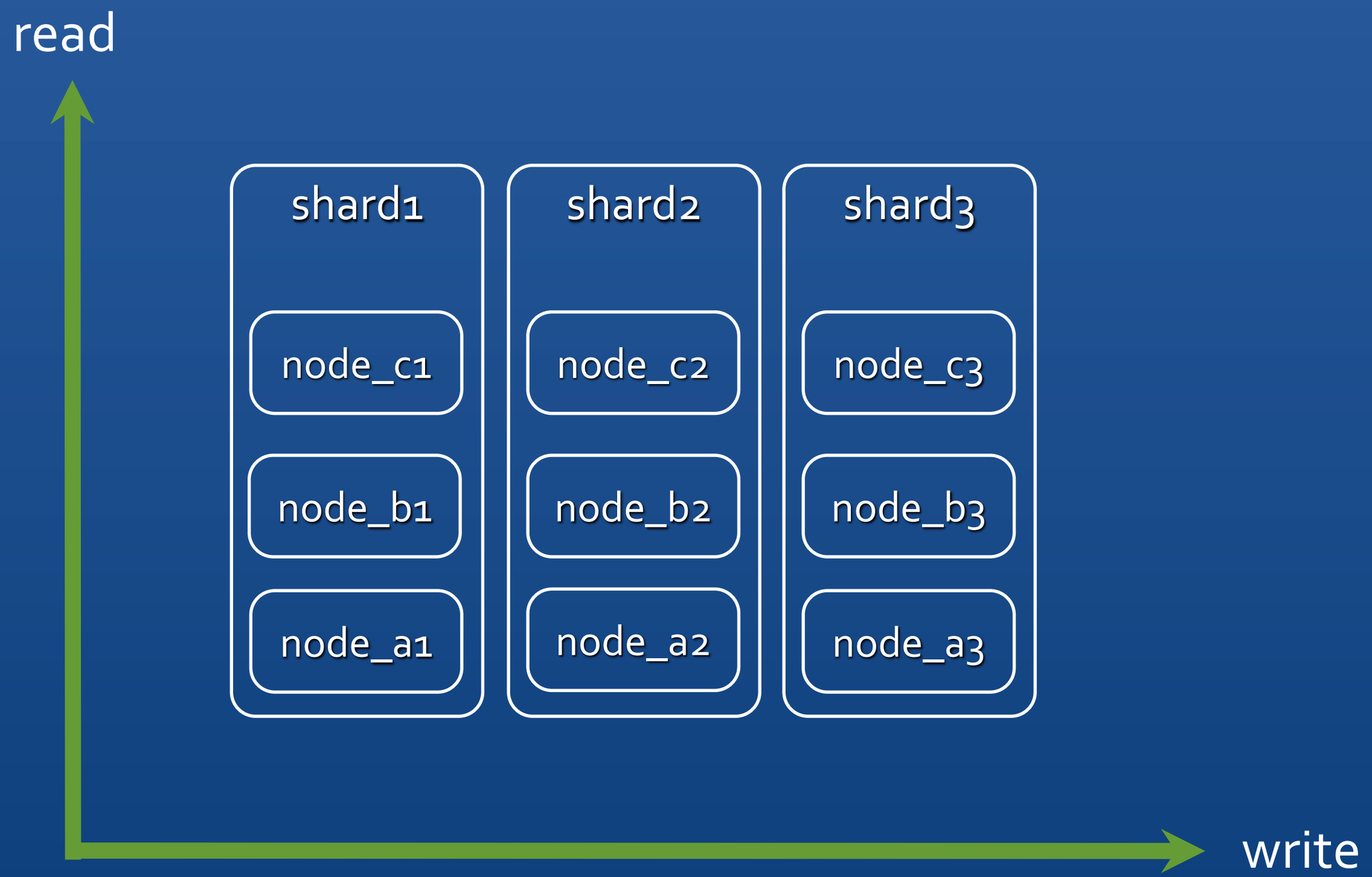
10gen mongoDB

Write scaling – add Shards



10gen mongoDB

Write scaling – add Shards



10gen mongoDB

Object Document Mapping

10gen mongoDB

A New Paradigm for Mapping Objects <-> Documents

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

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

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”

10gen mongoDB

ODM Systems for Python

10gen mongoDB

ODM Systems for Python

- Several ODMs in the Python World
  - MongoKit
  - MongoEngine
  - Ming
  - ... also a few projects to integrate with Django

10gen mongoDB

ODM Systems for Python

- Several ODMs in the Python World
  - MongoKit
  - MongoEngine
  - Ming
  - ... also a few projects to integrate with Django
- Core concept is to let you define a schema
  - Optional and Required Fields
  - Valid Datatype(s)
  - Validation Functions
  - Bind to Objects instead of Dictionaries

10gen mongoDB

ODM Systems for Python

- Several ODMs in the Python World
  - MongoKit
  - MongoEngine
  - Ming
  - ... also a few projects to integrate with Django
- Core concept is to let you define a schema
  - Optional and Required Fields
  - Valid Datatype(s)
  - Validation Functions
  - Bind to Objects instead of Dictionaries
- Let's show simple examples of MongoKit & MongoEngine

MongoDB via MongoKit

```
from mongokit import *
class Book(Document):
    __database__ = 'bookstore'
    __collection__ = 'books'
    structure = {
        'title': unicode,
        'author': [unicode], # A heterogenous list is declared as 'list'
        'publisher': unicode,
        'isbn': unicode,
        'price': {
            'currency': unicode,
            'discount': float,
            'msrp': float
        },
        'publicationYear': int,
        'edition': unicode,
        'editor': unicode,
        'tags': [unicode]
    }

    required = ['title', 'publisher', 'price.currency', 'price.discount',_
                'price.msrp', 'publicationYear']

    default_values = {'tags': []}
```

10gen mongoDB

MongoDB via MongoEngine

```
class Book(Document):
    title = StringField(max_length=120, required=True, unique=True,
unique_with='edition')
    author = ListField(StringField(), default=list)
    publisher = StringField(required=True)
    isbn = StringField(max_length=16, required=True)
    price = EmbeddedDocumentField(Price)
    publicationYear = IntField(required=True)
    edition = StringField(required=False)
    editor = StringField()
    tags = ListField(StringField(), default=list)

class Price(EmbeddedDocument):
    currency = StringField(max_length=3, required=True)
    discount = FloatField(required=True)
    msrp = FloatField(required=True)
```

Lots of Other Fun To Be Had

- GridFS for File Storage
- Django Integration
- Beaker plugin for complex caching (built for / in use at Sluggy.com)
- Asynchronous version of pymongo from bit.ly for use with event driven libraries like Tornado and Twisted
- ... and a lot more (too much to list)

10gen mongoDB

Let’s play with a live example!

- I was a bit bored this morning and threw together a quick Flask based app to demo MongoDB...

10gen mongoDB

download at mongodb.org

mms.10gen.com

(MongoDB monitoring! free! Awesome!)

We're Hiring !

brendan@10gen.com

(twitter: @rit)

conferences, appearances, and meetups

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

Facebook | Twitter | LinkedIn

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

10gen mongoDB