Intro to NoSQL databases with MongoDB

Hector Correa
hector@hectorcorrea.com
@hectorjcorrea

Agenda

- What are NoSQL databases
- Why NoSQL
- Examples using MongoDB
- Pros and cons
- Q&A

What are NoSQL databases

What are NoSQL databases

- A type of databases
- Don't use the relational model
- Good fit for distributed environments
- (Usually) don't use SQL
- (Most of them) are open source

Source: http://nosql-database.org

What are NoSQL databases

"NoSQL refers to an ill-defined set of mostly open-source databases, mostly developed in the early 21st century, and mostly not using SQL"

Source: NoSQL Distilled by Sadalage and Fowler

What are NoSQL databases

NoSQL is a movement
not a specific technology or
product

What are NoSQL databases

NoSQL term coined in 2009 by Johan Oskarsson while organizing a meetup

> NOSQL Meetup
>
> This meetup is about "open source, distributed, non relational databases".
>
> Have you run into limitations with traditional relational databases? Don't mind trading a query language for scalability? [...]

Source: http://nosql.eventbrite.com

What are NoSQL databases

It doesn't mean "Screw SQL"

It's more like "Not Only SQL"

What are NoSQL databases

NoSQL has very little to do with SQL (structured query language)

It should have been called Not Only Relational Databases

#NoRDBMS anyone?

Quick Demo with MongoDB

Why NoSQL?

Why NoSQL?

Very large and distributed databases

Rise of unstructured data

Ease of development

Why NoSQL? - large datasets

Massive datasets (Google, Amazon, Facebook)

Distributed environments (hundreds of nodes)

RDBMS scale up but not scale out

Why NoSQL? - large datasets

But we don't have that problem
at my company...

Why NoSQL? - large datasets

But we don't have that problem
at my company...

...true, and we also thought that
640 KB of RAM should be
enough for everybody.

Source: http://www.youtube.com/watch?v=ql_g07C_Q5I

Why NoSQL? - unstructured data

Web pages (Google, Yahoo)

Log data, scientific data

Content Management Systems

Field1, field2, field3, fieldN

Storing field definitions as rows

Tracking changes (usually a BLOB)

Why NoSQL? - ease of development

Data impedance mismatch (OO vs RDBMS)

Applies to both structured and unstructured data

Aggregates are desirable in a cluster environment

NoSQL can reduce this friction

Types of NoSQL databases

Types of NoSQL databases

- **Key-value**
  SimpleDB, Redis, Dynamo, Voldemort, Riak

- **Document-oriented**
  MongoDB, CouchDB, RavenDB

- **Column-oriented**
  BigTable, HBase, CASSANDRA, PNUTS

Sources:
Book: NoSQL Distilled by Sadalage and Fowler
Paper: NoSQL Databases: a step to database scalability in Web environment by Jaroslav Pokorny

Types of NoSQL databases

Common Characteristics

- Not using the relational model
- Run well on clusters
- Can handle huge amount of data
- Open Source
- Build for 21st century web access
- Schema-less / schema-free
- BASE (not ACID)

ACID Transactions

- **Atomicity.** Transactions are all or nothing.
- **Consistency.** The database will be in a consistent state when the transaction begins and ends.
- **Isolation.** The transaction will behave as if it is the only operation being performed upon the database.
- **Durability.** Upon completion of the transaction, the operation will not be reversed.

Source: BASE: An Acid Alternative by Dan Pritchett http://queue.acm.org/detail.cfm?id=1394128

Types of NoSQL databases

BASE

- **B**asically **A**vailable, **S**oft state, **E**ventually consistent
- BASE is diametrically opposed to ACID
- ACID is pessimistic and forces consistency at the end of every operation
- BASE is optimistic and accepts that the database consistency will be in a state of flux

Source: BASE: An Acid Alternative by Dan Pritchett http://queue.acm.org/detail.cfm?id=1394128

Types of NoSQL databases

"BASE is optimistic and accepts that the database consistency will be in a state of flux"

This is really a business requirement, not a technical one (overbooked planes, oversold items)

Source: BASE: An Acid Alternative by Dan Pritchett http://queue.acm.org/detail.cfm?id=1394128

Types of NoSQL databases

BASE sounds scary at first, but...

"it leads to levels of scalability that cannot be obtained with ACID"

Source: BASE: An Acid Alternative by Dan Pritchett http://queue.acm.org/detail.cfm?id=1394128

mongoDB

Document-oriented
Open source
Free
Multi-platform
Maintained by 10gen

http://www.mongodb.org

Image source: http://upload.wikimedia.org/wikipedia/commons/e/eb/MongoDB_Logo.png

Goal of MongoDB

"**bridge the gap between key-value stores** (which are fast and highly scalable) **and traditional RDBMS systems** (which provide rich queries and deep functionality)." - Mike Dirolf

Source: http://www.10gen.com/presentations/webinar/introduction-to-mongodb

MongoDB Examples

MongoDB Examples

```
INSERT INTO table (f1, f2, f3)
VALUES (v1, v2, v3)
```

```
db.collection.insert(
   {f1: v1, f2: v2, f3: v3}
)
```

MongoDB Examples

```
db.blog.insert({
   url: “blog-1”,
   title:"Blog 1",
   text: “blah blah blah”,
   author: "jdoe",
   tags: ["software", "databases"],
   addedOn: "2013-04-01"
});
```

MongoDB Examples

```
UPDATE table
SET f1 = v1, f2 = v2
WHERE f3 = v3
```

```
db.collection.update(
    {f3: v3},          // where
    {$set:
      {f1: v1, f2: v2}
    }
  )
```

MongoDB Examples

```
SELECT f1, f2
FROM table
WHERE f3 = “X”
```

```
db.collection.find(
   {f3: “X”},              // where
   {f1: 1, f2: 1}          // projection
)
```

MongoDB Examples

```
SELECT i.id, i.date, i.total,
   c.name, c.address,
   t.qty, p.name, t.price
FROM invoices i
   INNER JOIN customers c ON i.custId = c.id
   INNER JOIN items t ON t.invoiceId = i.id
   INNER JOIN prods p ON t.prodId = p.id
WHERE i.id = 34
```

MongoDB Examples

id	date	total	name	address	qty	name	price
34	2013-04-01	100	Customer A	123 main	2	item A	30
34	2013-04-01	100	Customer A	123 main	1	item B	40

MongoDB Examples

```
db.invoices.find({id: 34})
```

MongoDB Examples

```
{
   id: 34,
   date: “2013-04-01”,
   total: 100,
   customer: {
      name: “Customer A”,
      address: “123 main”
   },
   items: [
      {qty: 2, name: “item A”, price: 30},
      {qty: 1, name: “item B”, price: 40},
   ]
}
```

MongoDB Examples

```
SELECT title, addedOn
FROM blog
WHERE addedOn >= “2013-04-01”
  and addedOn <= “2013-04-15”
```

```
db.blog.find(
   {addedOn:{
     $gte: "2013-04-01",
     $lte: "2013-04-15"}
   },
   {title: 1, addedOn: 1}
)
```

MongoDB Examples

- Insert/Update/Find
- Arrays & nested documents
- Java / C# Sample
- Indices
- MapReduce
- Server Functions
- Aggregation Framework

Code can be found at: https://github.com/hectorcorrea/intro-to-nosql-with-mongodb

Replication & Sharding

MongoDB Replication

Replication - saving the same data multiple times

Gives you redundancy in case one server fails
Allows you to spread your reads

[server 1]
All
Customers

[server 2]
All
Customers

[server 3]
All
Customers

MongoDB Replication

Master-Slave

- One master (you define it)
- Many slaves

```
[server 1]$ mongod --master

[server 2]$ mongod --slave --source server1

[server 3]$ mongod --slave --source server1
```

MongoDB Replication

Replica Sets

- Like master-slave...
- ...but the master is designated by the set
- Automatic failover

```
[server 1]$ mongod --replSet name/server2,server3

[server 2]$ mongod --replSet name/server1,server3

[server 3]$ mongod --replSet name/server1,server2
```

MongoDB Replication

```
$ mongo

> config = {
     _id: "rs1",
     members: [
          {_id: 0, host: "server1"},
          {_id: 1, host: "server2"},
          {_id: 2, host: "server3"}
     ]
  }

> rs.initiate(config)
> rs.status()
```

MongoDB Sharding

Sharding is a fancy word for “data partitioning”

MongoDB supports automatic sharding

```
db.runCommand({
   “shardcollection”:”customers”,
   “key”: {“region”:1}
})
```

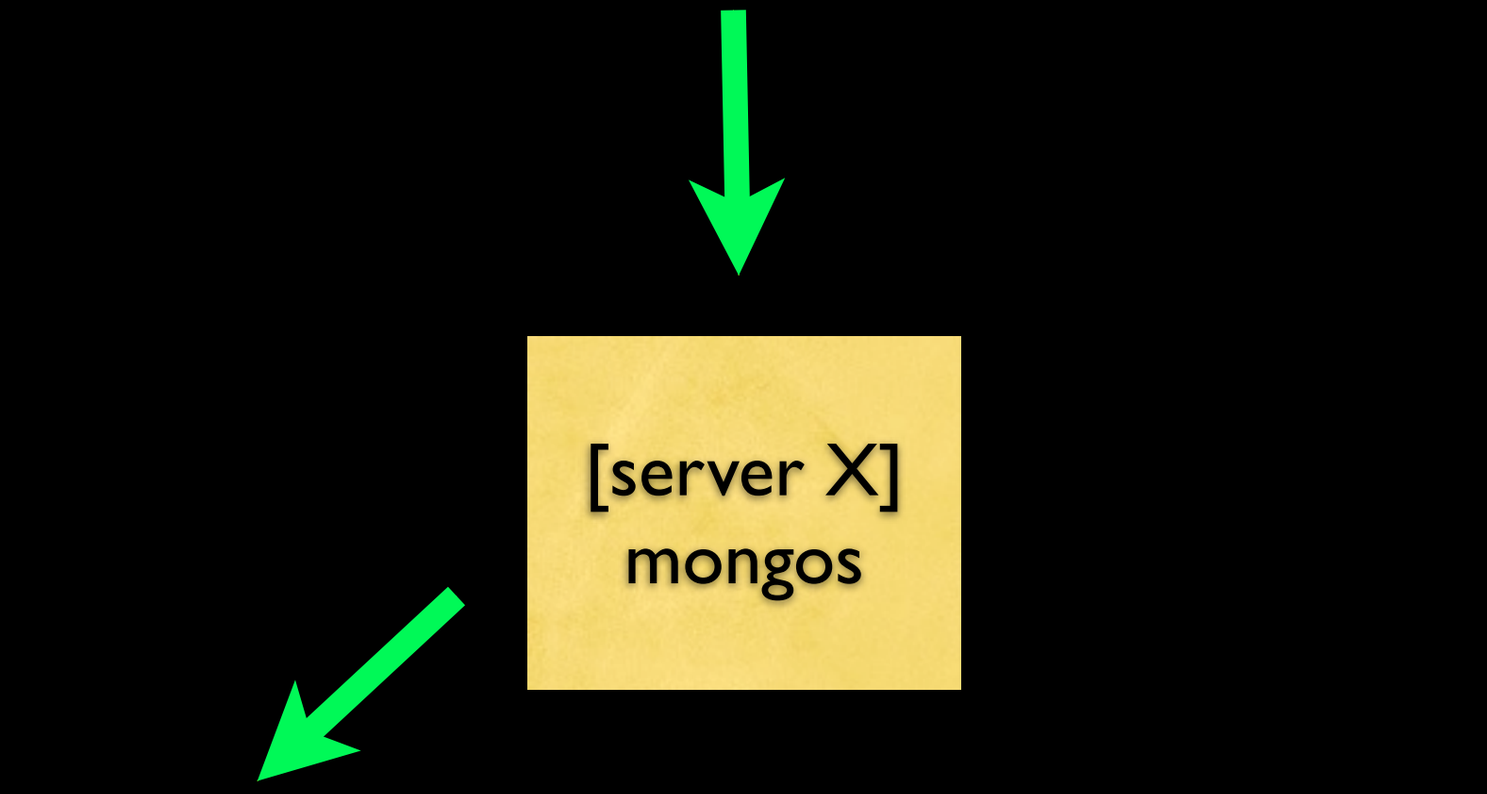
[server 1]
Customers
West

[server 2]
Customers
MidWest

[server 3]
Customers
East

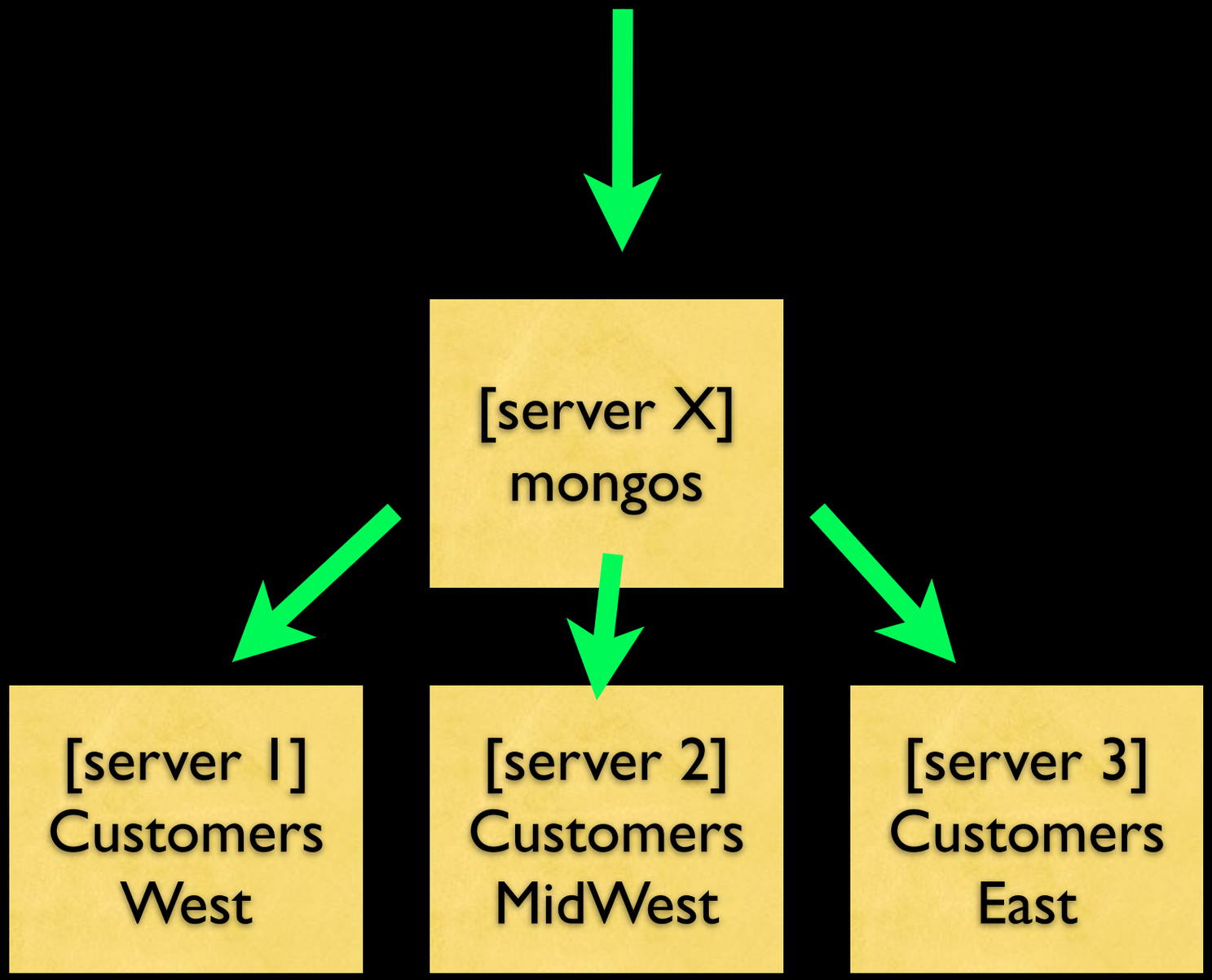
MongoDB Sharding

db.customers.find({region: "WEST"})



MongoDB Sharding

db.customers.find({name:"John"})



To SQL or no to SQL, that is the question

http://en.wikipedia.org/wiki/File:Edwin_Booth_Hamlet_1870.jpg

Advantages of Relational Databases

- Some data fits the relational model nicely
- SQL is a declarative language
- SQL is universal
- Joins
- Multi-row / multi-table ACID transactions
- One size fits most
- Well known technology

Advantages of NoSQL Databases

- Cluster friendly / scales out
- Tend to be very fast
- Handle complex data nicely
  - Reduced data impedance mismatch
  - Joins don't needed as much
  - Multi-row / multi-table transactions don't needed as much

Disadvantages of NoSQL Databases

- Many different data modes
- No standard/universal query syntax
- Each product uses a different one
  - Vendor lock-in?
- Learning curve (on your data layer!)
- Can you live with BASE?

To SQL or no to SQL

Consider NoSQL Databases if for performance reasons you are...

- Not using referential integrity
- Minimizing the use of JOINS
- Denormalizing (a lot) of your data
- Saying “no” to some features

Try One NoSQL database

Recommended Books

- **NoSQL Distilled** by Pramod Sadalage and Martin Fowler
- **MongoDB The Definitive Guide** by Kristina Chodorow and Michael Dirolf

Thank you!

Hector Correa
hector@hectorcorrea.com
@hectorjcorrea

Glossary

- Bigtable: Google NoSQL database
- Big Data: Buzz word
- HBase: Apache's NoSQL database
- MapReduce: Programming model to process data in a cluster (ETL)
- Hadoop: Apache product to run MapReduce jobs
- ACID: Mantra for Relational Databases. Properties for database transactions (atomicity, consistency, isolation, durability)
- BASE: Mantra for NoSQL databases. Basically Available, Soft state, and eventually consistent.