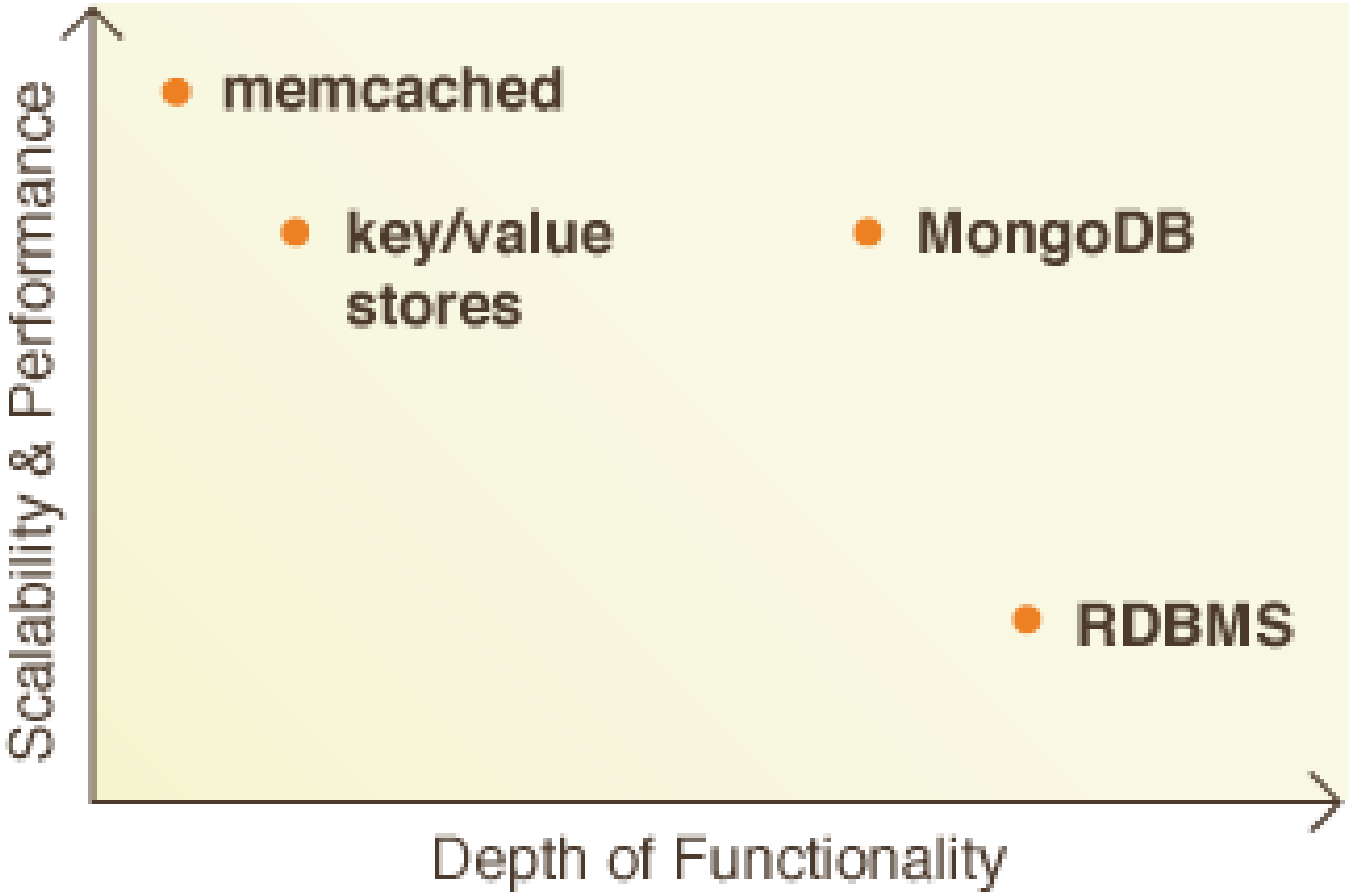
Document-Oriented DBs and MongoDB

Mathias Stearn

10gen

VolcaNoSQL EU – April 20, 2010


memcached
key/value stores
MongoDB
RDBMS
Scalability & Performance
Depth of Functionality

- Only Two Document Oriented DBs Right Now
  - MongoDB and CouchDB
  - Everyone has an opinion on what makes a DODB
    - Hard to choose "defining" characteristics
    - This is my take on the space – deal with it

- Document Oriented
  - Think JSON Documents, not Word/OOo Documents
  - Can store files through Attachments and GridFS
  - Could use XML but XML sucks

```
{
  _id: "mstearn",
  name: "Mathias Stearn",
  karma: 42,
  active: true,
  birthdate: new Date(517896000000),
  interests: ["MongoDB", "Python", "Üñíçøđě"],
  subobjects: [{foo: "bar"},
               {foo: "baz", count: 13}]
}
```

- Hierarchical
  - Can nest objects to arbitrary depth
  - Server can reach into objects
  - Whole "Object" stored at one place on disk

```
{
  comments: [
    { by: 'mstearn', body: 'text', tags: ['empty']
      votes: {good: 100, bad: 10, net: 90} },
    { by: 'mdirolf', body: 'what?', tags: ['question']
      votes: {good: 30, bad: 40, net: -10} }
  ]
}
```

- Not Relational
  - Not forced into rows/columns/tables
  - No built-in joins
  - Less need because objects can directly store lists
  - Many-to-Many still possible (learn how at workshop)
  - No SQL (no SQL injections either)
  - No Object-Relational impedance mismatch

- Not Just Key-Value Store
  - Key and value are not separate
  - Supports queries on non-primary keys
    - Secondary Indexes
  - Supports Aggregation
    - Currently via JavaScript MapReduce
    - Both DBs looking into alternatives
  - Can be as fast as a KV store if you only need KV features
    - But still have access to a real database when needed
  - Less custom code needed

- Not the same as stuffing a JSON blob in a database
  - Database understands document format
  - Can query on any field
  - "Use the right tool for the job"

WARNING

I am not an expert on CouchDB!

Pros

- HTTP RESTful Interface
- Stores and communicates in plain JSON
- Query using precomputed JS Map/Reduce views
- Fastest if you use Bulk Insert
- Uses Append-Only File

Cons

- HTTP RESTful Interface
- Stores and communicates in plain JSON
- Query using precomputed JS Map/Reduce views
- Fastest if you use Bulk Insert
- Uses Append-Only File

MongoDB

- Custom wire protocol with many supported languages
- Stores and communicates in BSON (Binary JSON)
- Rich Ad-Hoc Query Language
  - MapReduce for aggregation
- Bulk Insert available, but regular insert is very fast
- Data is updated in place

- The Mongo Shell
  - `http://try.mongodb.org` ← go here now
- Full JS shell + MongoDB extensions
- Most MongoDB documentation uses shell syntax

```
db.users.insert({_id:'mstearn',
                 name: {first:'Mathias',
                        last:'Stearn'}
                 company: '10gen',
                 knows: ['MongoDB', 'Python', 'C++'],
                 posts: 42})

db.users.find({_id: 'mstearn'})
db.users.find({company: '10gen'})
db.users.find({posts: {$gte: 40, $lte: 50}})
db.users.find({'name.last': 'Stearn'})
db.users.find({knows: 'MongoDB'})
db.users.find({knows: {$in: ['MongoDB', 'Mongo']}})
db.users.find({knows: {$all: ['MongoDB', 'Python']}})
db.find().sort({posts: -1}).skip(10).limit(10)
```

- You (probably) don't need sharding!
- At the last presentation 3 out of 50 people were interested in sharding
- Single Master + Read Slaves for Scaling reads
- Largest Mongo install is 12GB
  - Single Master + Replication
- Wordnik.com has 1.5TB and over 5 Billion docs
- Speed and Scalability are different things
  - But you only need scalability if you're too slow

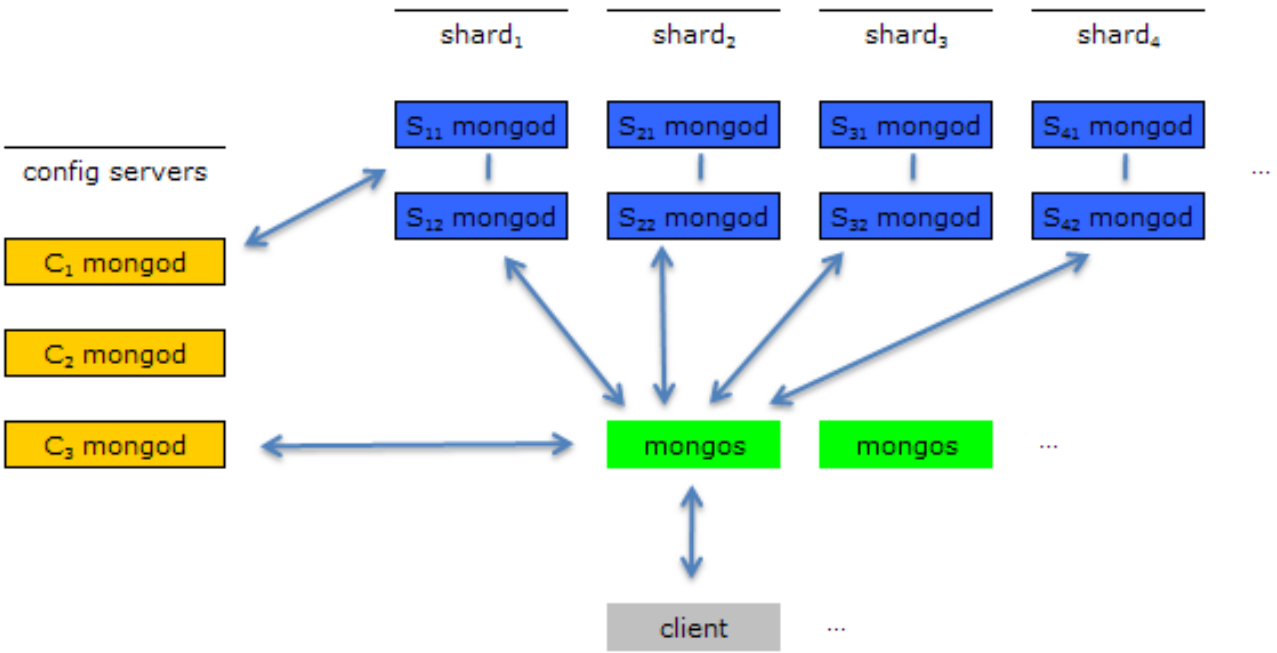
JavaScript used for:

- Shell and Documentation
- (Very) Advanced Queries
- “Group By” Queries
- MapReduce

```
db.users.find({$where: "this.a + this.b >= 42"});
db.posts.group(
  { key: "user"
  , initial: {count:0, comments:0}
  , reduce: function(doc,out){
      out.count++;
      out.comments += doc.comments.length; }
  , finalize: function(out){
      out.avg = out.comments / out.count; }
  });
```

Fast, Scalable, Available, and Reliable

- Master-Slave replication for Availability and Reliability
  - Replica-Pairs support auto-negotiation for master
- Auto-Sharding for Horizontal Scalability
  - Distributes based on specified field
  - Currently alpha
- MMAP database files to automatically use available RAM
- Asynchronous modifications


shard1
shard2
shard3
shard4
S11 mongod
S21 mongod
S31 mongod
S41 mongod
S12 mongod
S22 mongod
S32 mongod
S42 mongod
config servers
C1 mongod
C2 mongod
C3 mongod
mongos
mongos
client

Official

- Java/JVM
- Python
- Ruby
- C/C++
- Perl
- PHP

Community Supported

Closure, Scala, C#, Haskell, Erlang, *and More*

JSON

- String (UTF8)
- Double
- Object (hash/map/dict)
- Array
- Bool
- Null / Undefined

Extras

- Date
- Int32 / Int64
- ObjectID (12 bytes: timestamp + host + pid + counter)
- Binary (with type byte)

- $set
- $inc
- $multiply (soon)
- $push / $pushAll
- $pull / $pullAll

```
db.posts.update({_id:SOMEID}, {$push:{tags:"mongodb"}})
db.tags.update({_id:"mongodb"}, {$inc:{count:1}},
                                {upsert:true}})
```

Simple

```
db.posts.findOne({ user: "mstearn" });

var cursor = db.posts.find({ user: "mstearn" });
cursor.forEach(function(){
  doSomething(this.text);
});
```

Sorted

```
db.posts.find(
   { user: "mstearn" }
).sort({timestamp:-1})
```

Paginated

```
db.posts.find(
   { user: "mstearn" }
).sort({timestamp:-1}).skip(10).limit(10);
```

Simple Tag Search

```
db.posts.find(
   { user: "mstearn"
   , tags: "mongo"
   }
).sort({timestamp:-1}).skip(10).limit(10);
```

Complex Tag Search

```
db.posts.find(
   { user: "mstearn"
   , tags: {$in: ["mongo", "mongodb"]}
   }
).sort({timestamp:-1}).skip(10).limit(10);
```

Nested Objects

```
db.posts.find(
   { user: "mstearn"
   , tags: {$in: ["mongo", "mongodb"]}
   , comments.user: "mdirolf"
   }
).sort({timestamp:-1}).skip(10).limit(10);
```

Regular Expressions

```
db.posts.find(
   { user: "mstearn"
   , tags: {$in: ["mongo", "mongodb"]}
   , comments.user: "mdirolf"
   , text: /windows/i
   }
).sort({timestamp:-1}).skip(10).limit(10);
```

Ranges

```
db.posts.find(
   { user: "mstearn"
   , tags: {$in: ["mongo", "mongodb"]}
   , comments.user: "mdirolf"
   , text: /windows/i
   , points: {$gt: 10, $lt: 100}
   }
).sort({timestamp:-1}).skip(10).limit(10);
```

Arbitrary JavaScript

```
db.posts.find(
   { user: "mstearn"
   , tags: {$in: ["mongo", "mongodb"]}
   , comments.user: "mdirolf"
   , text: /windows/i
   , points: {$gt: 10, $lt 100}
   , $where: "this.a + this.b >= 42"
   }
).sort({timestamp:-1}).skip(10).limit(10);
```

```
db.posts.mapReduce(
   function() {
     this.comments.forEach(c){
       emit(c.user,
            {count:1, words:c.text.split().length; } }
 , function(key, values){
     for (var i=1; i<values.length; i++){
       values[0].count += values[i].count;
       values[0].words += values[i].words; }
     return values[0]; }
 , { finalize: function(out){
       out.avg = out.words / out.count;
       return out; }
   , query: {posted: {$gt: new Date(2010,0,1)}}
   , out: 'posts.comment_stats'
   }
 });
```

Easy Hadoop-Mongo Integration

- mongoexport can export to JSON/CSV/TSV
  - Can also easily use a custom script
- Process in Hadoop
- Use mongoimport to get data back into MongoDB

Better Hadoop-Mongo Integration

- mongodump writes a stream of BSON to a file
- Write an InputFilter and RecordReader to read BSON
- Write a BSONWriter class to directly use the data
  - Just added two methods to driver to make this easier
- Process the data with the Java/Scala/Closure driver
- Write a custom RecordWriter to either:
  - Dump to a file and use mongorestore
  - Dump the output directly to MongoDB
- Optional: use renameCollection to mimic our MapReduce

Upcoming events

- NoSQL Live! from Boston (March 11)
- MongoDB Training in San Francisco (March 25)
- San Fransisco MySQL Meetup (April 12)

Links

- http://mongo.kylebanker.com (Try mongo in your browser)
- http://www.mongodb.org
- #mongodb on irc.freenode.net
- mongodb-user on google groups

- mathias@10gen.com
- @mathias_mongo on twitter