Using MongoDB and Python

Michael Bright
Python User Group, Grenoble

23 Fevrier 2016

Databases

Databases

Relational Databases (RDBMS) “Les SGBDRs” (OLTP – Online Transaction Processing)

- Based on relational database model invented by E.F. Codd (IBM) in 1970
- SQL (SEQUEL): expressive query language
- Achieves performance increase through “vertical scaling” (higher performance node)

NoSQL (No SQL, or “Not just SQL”?) existed before RDBMS, resurged out of a need for more Scalability for Web2.0

- Columnar, e.g. Cassandra, Hbase, Vertica
- Document Oriented, e.g. CouchDB, MongoDB, RethinkDB
- Graph, e.g. Neo4J
- Key-value, e.g. CouchDB, Dynamo, Riak, Redis, Oracle NoSQL, MUMPS
- Multi-model, e.g. Alchemy, ArangoDB
- Achieves performance increase through “horizontal scaling” (by adding nodes)

OLAP – Online Analytic Processing

Databases for scalability

The need for scalability requires distribution of processing making it more difficult to satisfy all criteria, according to the CAP Theorem (2000)

- Consistency (Atomicity)
- Availability (A request will always receive a response)
- Partition Tolerance (if nodes become disconnected, the system continues to function)

It is impossible to satisfy all three constraints (proved by MIT, 2002).

RDBMS - favour Consistency and Availability but cannot scale horizontally

NoSQL - favour Consistency and Partitioning or Availability and Partitioning

Relational vs. NoSQL

Relational databases provide

- Rich query language (SQL)
- Strong consistency
- Secondary indices

NoSQL provides

- Flexibility
  - “dynamic schema” allows missing or extra fields, embedded structures
- Scalability
  - Adding nodes allows to scale data size
- Performance
  - Adding nodes allows to scale performance

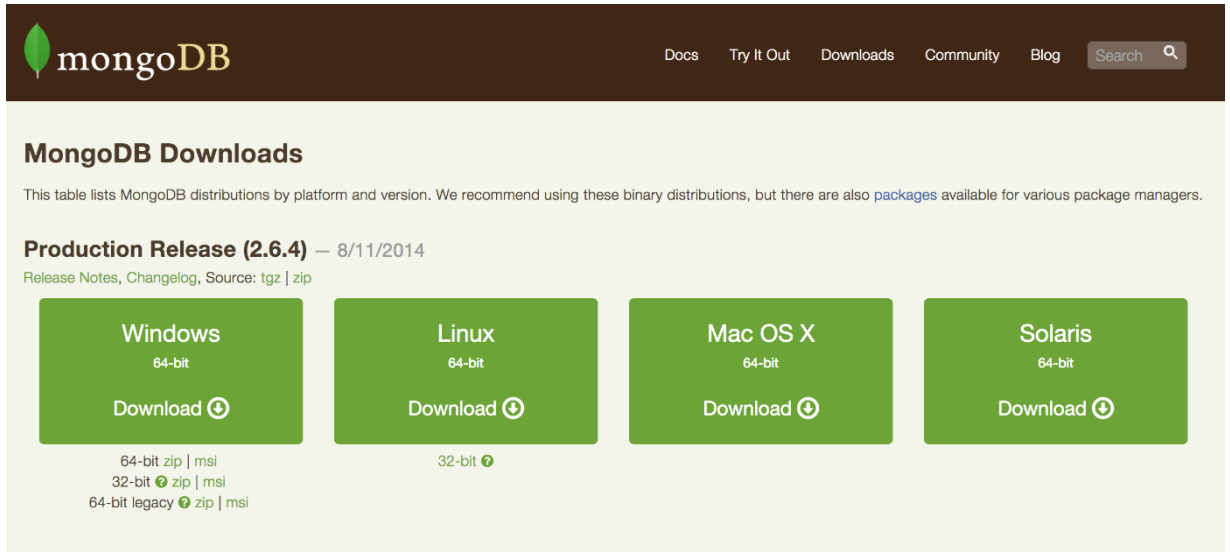
MongoDB combines both sets of advantages

MongoDB

MongoDB – a Document-Oriented DB

Open source DB on github with commercial support from MongoDB Inc (was 10gen).

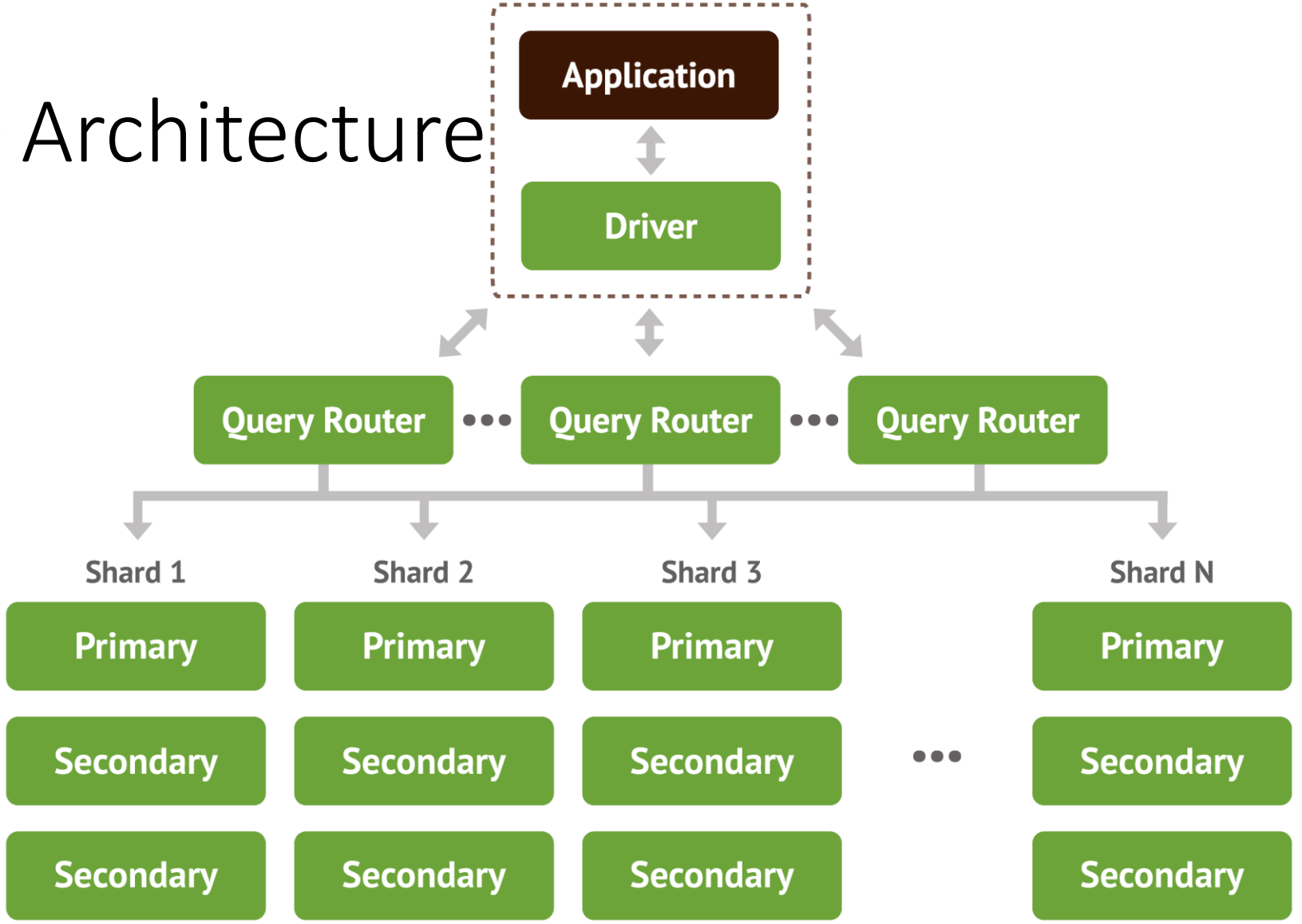
- Dynamic schema (schema-less) aids agile development
  - A document is an associative array (possibly more than 1-level deep), e.g. JSON Object
- Uses BSON (binary JSON format) for serialization (easy to parse)
- Binaries downloadable for Linux(es), Windows, OSX, Solaris
- Drivers available in many languages
- Provides an aggregation framework
- Is the "**M**" in MEAN



MongoDB – Performance

- Written in C++
- It achieves high performance, especially on dynamic queries
- Allows primary & secondary indices (main tuning element)
- Is horizontally scalable across many nodes
- Extensive use of memory-mapped files
  i.e. read-through write-through memory caching.
- Provides high availability with Master/slave replication
- Provides Sharding (horizontal partitioning of data across nodes)
  - Horizontal partitioning : complete documents (or rows) are stored on a node
  - Vertical partitioning: documents (or rows) are split across nodes

MongoDB – Architecture



MongoDB – Terminology

RDBMS		MongoDB
Table, View	→	Collection
Row	→	Document
Index	→	Index
Join	→	Embedded Document
Foreign Key	→	Reference
Partition	→	Shard

MongoDB – Enterprise Products

There are supported products for the enterprise

- MongoDB Enterprise: for production workloads

- MongoDB Compass: Data and Schema visualization tool

- MongoDB Connector for BI:

  Allows users to visualize their MongoDB Enterprise data using existing relational business intelligence tools such as Tableau.

  Foreign data wrapper with PostgreSQL to provide a relational SQL view on MongoDB data

MongoDB v3.2: new features

- The default storage engine is now WiredTiger
  - Following acquisition of WiredTiger, Inc.
  - Introduced as an optional engine in MongoDB 3.0
  - Has more granular concurrency control and native compression (lowering storage costs, increasing h/w utilization, throughput and providing more predictable performance)
- Replication election enhancements
- Config servers as replica sets
- readConcern, and document validations.
- OpsManager 2.0

MongoDB v3.2: new aggregation features

- $sample, $lookup
- $indexStats
- $filter,$slice,$arrayElemAt, $isArray, $concatArrays
- Partial Indexes
- Document Validation
- 4 bit testing operators
- 2 new Accumulators for $group
- 10 new Arithmetic expression operators

MongoDB components

Components

mongod - The database daemon process.
mongos - Sharding controller.
mongo - The database shell (uses interactive javascript).

Utilities

mongodump - MongoDB dump tool - for backups, snapshots, etc.
mongorestore - MongoDB restore a dump
mongoexport - Export a single collection to test (JSON, CSV)
mongoimport - Import from JSON or CSV
mongofiles - Utility for putting and getting files from MongoDB GridFS
mongostat - Show performance statistics

MongoDB University (http://university.mongodb.com/)

MongoDB University

PROFESSIONAL CERTIFICATION ONLINE COURSES PUBLIC TRAINING PRIVATE TRAINING

Online Course Schedule

Schedule

Course	Start	End	Level	Register
M101P: MongoDB for Developers	Mar 15, 2016	May 10, 2016	Introductory	Register
M101J: MongoDB for Java Developers	Mar 15, 2016	May 10, 2016	Introductory	Register
M101JS: MongoDB for Node.js Developers	Mar 15, 2016	May 10, 2016	Introductory	Register
M101N: MongoDB for .NET Developers	Mar 15, 2016	May 10, 2016	Introductory	Register
M102: MongoDB for DBAs	Mar 15, 2016	May 10, 2016	Introductory	Register
M202: MongoDB Advanced Deployment and Operations	Mar 15, 2016	May 10, 2016	Advanced	Register
M212: Adobe Experience Manager and MongoDB	Feb 09, 2016	Mar 15, 2016	Intermediate	Register
UD032: Data Wrangling with MongoDB (Udacity)	N/A	N/A	Intermediate	Register
M101X: Introduction to MongoDB using the MEAN Stack (edX)	N/A	N/A	Intermediate	Register

MongoDB drivers exist for many languages

12 official drivers, plus many community languages

MongoDB docs: (https://docs.mongodb.org/manual/)

mongoDB. FOR GIANT IDEAS

DOCS DRIVERS UNIVERSITY COMMUNITY BLOG ENTERPRISE

The MongoDB 3.2 Manual

ANNOUNCEMENTS:

- **MongoDB 3.2 Released**. See Release Notes for MongoDB 3.2 for new features in MongoDB 3.2.
- **MongoDB Connector for Business Intelligence Released**. See MongoDB Connector for BI Release notes for a list of new features.
- **New online course**. Adobe Experience Manager and MongoDB.

Welcome to the MongoDB 3.2 Manual! MongoDB is an open-source, document database designed for ease of development and scaling. The Manual introduces key concepts in MongoDB, presents the query language, and provides operational and administrative considerations and procedures as well as a comprehensive reference section.

Getting Started

MongoDB provides a Getting Started Guide in the following editions.

mongo Shell Edition	Python Edition	Java Edition
Node.JS Edition	C++ Edition	C# Edition

Once you complete the Getting Started Guide, you may find the following topics useful.

Using MongoDB

MongoDB shell

```
> mongo
MongoDB shell version: 3.2.3
connecting to: test

> show dbs
Money_UK            0.000GB
aggregation_example  0.000GB
local               0.000GB
test                0.000GB

> use Money_UK
switched to db Money_UK

> show collections
CARD
SAVING
```

MongoDB shell

```
> use test
switched to db test


> db.newcoll.insert({ 'field1': 'an example' })
WriteResult({ "nInserted" : 1 })

> db.newcoll.find().count()
1

> db.newcoll.find()
{ "_id" : ObjectId("56cc7446bb127b86163cf226"), "field1" : "an example" }

> db.newcoll.findOne()
{ "_id" : ObjectId("56cc7446bb127b86163cf226"), "field1" : "an example" }

> db.newcoll.find({'_id':ObjectId("56cc7446bb127b86163cf226")})
{ "_id" : ObjectId("56cc7446bb127b86163cf226"), "field1" : "an example" }
```

MongoDB shell

```
> var doc2={'field1': 'example', 'field2': 'example2'}

> db.newcoll.insert(doc2)
WriteResult({ "nInserted" : 1 })

> db.newcoll.find().count()
2

> db.newcoll.find().pretty()
{ "_id" : ObjectId("56cc7446bb127b86163cf226"), "field1" : "an example"
}
{
        "_id" : ObjectId("56cc7711bb127b86163cf227"),
        "field1" : "example",
        "field2" : "example2"
}
```

MongoDB indexing

Indexing is the single biggest tunable performance factor.

Can index on a single field, e.g. on ascending or descending values:

db.newcoll.ensureIndex({ ‘field1’: **1** }) // or **-1**: descending

Or subfields:

db.newcoll.ensureIndex({ ‘arr**.subfield**’: 1 })

Or multiple fields:

db.newcoll.ensureIndex({ ‘arr.subfield’: 1, ‘**field2**’: -1 })

MongoDB indexing - 2

It is also possible to provide hints to the indexer, on uniqueness, sparseness or to request index creation as a background task, e.g.

db.newcoll.ensureIndex({ ‘field1’: 1 }, { ‘**unique**’: true})

db.newcoll.ensureIndex({ ‘field1’: 1 , ‘**field2**’: 1}, { ‘**sparse**’: true})

db.newcoll.ensureIndex({ ‘field1’: 1 }, { ‘**background**’: true})

MongoDB searching

We can search based on a single field

db.newcoll.find({ ‘field1’: **‘any match’** })

Or subfields:

db.newcoll.find({ ‘arr**.subfield**’: ‘sub field match’ })

Or multiple fields:

db.newcoll.find({ ‘arr.subfield’: ‘sub field match’, ‘**field2**’: ‘match2’ })

Available indices will be used if possible

MongoDB sorting

We can sort the results in ascending or descending order

```
db.newcoll.find().sort( { ‘field1’: 1 } )   // or -1: descending
```

or

```
db.newcoll.find().sort( { ‘field1’: 1, ‘field2’: -1 } )
```

Available indices will be used if possible

MongoDB projections

We can reduce the number of returned fields (the projection).

This may enable data access directly from the index.

In this example we limit the returned fields to ‘field1’ (and the ‘_id’ index).

db.newcoll.find({ ‘field1’: ‘example’ }, **{ ‘field1’: 1}**)

In this example we limit the returned fields to ‘field1’ (without the ‘_id’ index).

db.newcoll.find({ ‘field1’: ‘example’ }, **{ ‘_id’: 0, ‘field1’: 1}**)

We can also override the default index

db.newcoll.find({ ‘field1’: ‘example’ }, **{ ‘_id’: ‘field1’ }**)

MongoDB explain plan

We can request explanation of how a query will be handled (showing possible index use)

```
> db.newcoll.find().explain()
{
	"queryPlanner" : {
		"plannerVersion" : 1,
		"namespace" : "test.newcoll",
		"indexFilterSet" : false,
		"parsedQuery" : {
			"$and" : [ ]
		},
		"winningPlan" : {
			"stage" : "COLLSCAN",
			"filter" : {
				"$and" : [ ]
			},
			"direction" : "forward"
		},
		"rejectedPlans" : [ ]
	},
	"serverInfo" : {
		"host" : "MJBRIGHT7",
		"port" : 27017,
		"version" : "3.2.3",
		"gitVersion" : "b326ba837cf6f49d65c2f85e1b70f6f31ece7937"
	},
	"ok" : 1
}
```

MongoDB special index types

- Geospatial Indexes (2d Sphere)
- Text Indexes
- TTL Collections (expireAfterSeconds)
- Hashed Indexes for sharding

MongoDB aggregation framework

Aggregation uses a pipeline of operations amongst

- $match
- $project
- $unwind
- $group
- $sort

MongoDB aggregation example

```
pipeline = [
    { "$project": { // Select fields of interest
            'year': { "$dateToString": { 'format': "%Y", 'date': "$date" } },
            'tags': 1,  'value': 1, },  },
    { "$match": { 'year':   str(year) }},
    { "$group": {"_id": "$tags", "total": {"$sum": {"$abs": "$value"}}} },
    { "$sort":  SON([("total", -1), ("_id", -1)]) }
  ]

cursor =    db.collection.aggregate(pipeline)
```

MongoDB aggregation example

```
pipeline = [
    { "$project": {
        'year': { "$dateToString": { 'format': "%Y", 'date': "$date" } },
        'tags': 1,  'value': 1, },  },
    { "$match": { 'year':   str(year) }},        // match on fields
    { "$group": {"_id": "$tags", "total": {"$sum": {"$abs": "$value"}}} },
    { "$sort":  SON([("total", -1), ("_id", -1)]) }
  ]

cursor =    db.collection.aggregate(pipeline)
```

MongoDB aggregation example

```
pipeline = [
    { "$project": {
        'year': { "$dateToString": { 'format': "%Y", 'date': "$date" } },
        'tags': 1,  'value': 1, },  },
    { "$match": { 'year':   str(year) }},
    { "$group": {"_id": "$tags", "total": {"$sum": {"$abs": "$value"}}} }, // ‘reduce’
    { "$sort":  SON([("total", -1), ("_id", -1)]) }
]

cursor =    db.collection.aggregate(pipeline)
```

MongoDB aggregation example

```
pipeline = [
    { "$project": {
        'year': { "$dateToString": { 'format': "%Y", 'date': "$date" } },
        'tags': 1,  'value': 1, },  },
    { "$match": { 'year':   str(year) }},
    { "$group": {"_id": "$tags", "total": {"$sum": {"$abs": "$value"}}} },
    { "$sort":  SON([("total", -1), ("_id", -1)]) }   // sort the results
]

cursor =    db.collection.aggregate(pipeline)
```

PyMongo – ca y est, on arrive !

But first …

PyMongo – ca y est, on arrive !

… there are alternatives to the official PyMongo driver.

- Motor is a full-featured, non-blocking MongoDB driver for Python Tornado applications.

- TxMongo is an asynchronous Twisted Python driver for MongoDB.

PyMongo – ca y est, on arrive !

… and other Python/MongoDB tools

ORM-like Layers

New users are recommended to begin with PyMongo.
Nevertheless other implementations are available providing higher abstraction.

Humongolus, MongoKit, Ming, MongoAlchemy, MongoEngine, Minimongo, Manga, MotorEngine

Framework Tools

This section lists tools and adapters that have been designed to work with various Python frameworks and libraries.

Django MongoDB Engine, Mango, Django MongoEngine, mongodb_beaker, Log4Mongo, MongoLog, C5t, rod.recipe.mongodb, repoze-what-plugins-mongodb, Mongobox, Flask-MongoAlchemy, Flask-MongoKit, Flask-PyMongo.

Questions?

Backup Slides