{name: “mongo”, type:“DB”}

An Introduction...

Version-0.0.1 Seth Floyd

What is MongoDB?

MongoDB is an open source document-oriented database system developed and supported by 10gen. It is part of the NoSQL family of database systems. Instead of storing data in tables as is done in a "classical" relational database, MongoDB stores structured data as JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster.

10gen began Development of MongoDB in October 2007. The database is used by MTV Networks, Craigslist, and Foursquare. MongoDB is the most popular NoSQL database management system.

Binaries are available for Windows, Linux, OS X, and Solaris.

Main Features

Ad hoc queries

MongoDB supports search by field, range queries, regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions. Aggregation functions (such as MapReduce), are sent directly to the database to be executed.

Indexing

Any field in a MongoDB document can be indexed (indexes in MongoDB are conceptually similar to those in RDBMSes). Secondary indices are also available.

Replication

MongoDB supports master-slave replication. A master can perform reads and writes. A slave copies data from the master and can only be used for reads or backup (not writes). The slaves have the ability to select a new master if the current one goes down.

Main Features

Load balancing

MongoDB scales horizontally using sharding.The developer chooses a shard key, which determines how the data in a collection will be distributed. The data is split into ranges (based on the shard key) and distributed across multiple shards. (A shard is a master with one or more slaves.)

MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure. Automatic configuration is easy to deploy and new machines can be added to a running database.

File storage

MongoDB could be used as a file system, taking advantage of load balancing and data replication features over multiple machines for storing files - GRIDFS.

GridFS, is included with MongoDB drivers. MongoDB exposes functions for file manipulation and content to developers. In a multi-machine MongoDB system, files can be distributed and copied multiple times between machines transparently, thus effectively creating a load balanced and fault tolerant system.

Sharding Info

Sharding distributes a single logical database system across a cluster of machines. Sharding uses range-based portioning to distribute documents based on a specific shard key.

To use replication with sharding, each shard is deployed as a replica set.

REPLICA SET - A cluster of MongoDB servers that implements master-slave replication and automated failover. MongoDB's recommended replication strategy.

More info on Sharding can be found here:

http://docs.mongodb.org/manual/faq/sharding/

Installation

MongoDB runs on most platforms, and supports 32-bit and 64-bit architectures. 10gen, the MongoDB makers, provides both binaries and packages.

MongoDB is easily installed from the Redhat Yum repositories and also through the Ubuntu repos via Apt-Get. On Mac OSX you can use MacPorts or Homebrew to install.

Must be installed as root.

Before you start mongod for the first time, you will need to create the data directory. By default, mongod writes data to the /data/db/ directory.

Tools in MongoDB

In a MongoDB installation the following commands are available:

mongo

MongoDB offers an interactive shell called mongo, which lets developers view, insert, remove, and update data in their databases, as well as get replication information, set up sharding, shut down servers, execute JavaScript, and more. Administrative information can also be accessed through a web interface, a simple webpage that serves information about the current server status. By default, this interface is 1000 ports above the database port at 28017.

mongostat

mongostat is a command-line tool that displays a summary list of status statistics for a currently running MongoDB instance: how many inserts, updates, removes, queries, and commands were performed, as well as what percentage of the time the database was locked and how much memory it is using. This tool is similar to the UNIX/Linux vmstat utility.

mongotop

mongotop is a command-line tool providing a method to track the amount of time a MongoDB instance spends reading and writing data. mongotop provides statistics on the per-collection level. By default, mongotop returns values every second. This tool is similar to the UNIX/Linux top utility.

Tools in MongoDB

Mongod is the primary daemon process for the MongoDB system. It handles data requests, manages data format, and performs background management operations.

Mongos for “MongoDB Shard,” is a routing service for MongoDB shard configurations that processes queries from the application layer, and determines the location of this data in the sharded cluster, in order to complete these operations. From the perspective of the application, a mongos instance behaves identically to any other MongoDB instance.

Getting Started

Connect to a Database

In this section you connect to the database server, which runs as mongod, and begin using the mongo shell to select a logical database within the database instance and access the help text in the mongo shell.

To start the Mongo instance do as root:

service mongod start

Getting started

To verify the mongod service is up you can do, *service mongod status* or grep from the process table for *mongod*.

The conf file for settings resides by default in:
/etc/mongod.conf

The log can be found at:
/var/log/mongo/mongod.log

Connecting To A Database

To connect to a database, at the prompt type as root or another user : (Assuming that authentication is turned off)

mongo

By default, mongo looks for a database server listening on port ***27017*** on the localhost interface. To connect to a server on a different port or interface, use the ***--port*** and ***--host*** options.

Beginning Steps

Once you are connected you will see output similar to:

MongoDB shell version: 2.2.3

connecting to: test

Welcome to the MongoDB shell.

For interactive help, type "help".

For more comprehensive documentation, see

http://docs.mongodb.org/

Questions? Try the support group

http://groups.google.com/group/mongodb-user

>

Beginning Steps

From here you can derive that you are connected to the Mongo server. To view available databases you can do:*> show dbs*

local (empty)

test (empty)

This will show you the available databases. To connect to one of the databases you will do:

USE <NAME OF DATABASE>

> use test

switched to db test

Beginning Steps

You can now show the collections within the database by issuing the command:

```
show collections
```

If you need to double check what database you are connected to you can simply do:

```
db
```

The help command will display a few help options along with various help operators.

```
help
```

Data manipulation: collections and documents

MongoDB stores structured data as JSON-like documents, using dynamic schemas (called BSON), rather than predefined schemas. In MongoDB, an element of data is called a *document*, and documents are stored in *collections*. One collection may have any number of documents.

In relating to an Oracle RDBMS structure one could say that collections are like tables, and documents are like records.

But there is a big difference: any document in a collection can have completely different fields from the other documents. The only schema requirement MongoDB places on documents (aside from size limits) is that they must contain an '_id' field with a unique, non-array value.

Each document in a MongoDB collection can have different fields from the other documents (Note: "_id" field is obligatory, automatically created by MongoDB; it's a unique index which identifies the document. Its value need not be the default MongoID type — the user may specify any non-array value for _id as long as the value is unique).

Data manipulation: collections and documents

In a document, new fields can be added or existing ones suppressed, modified or renamed at any moment. There is no predefined schema. A document structure is very simple: it follows the JSON format, and consists of a series of key-value pairs, called "hash-tables" or "hashes".

The key of the key-value pair is the name of the field, the value in the key-value pair is the field's content. The key and value are separated by ":"

"_id": ObjectId("4efa8d2b7d284dad101e4bc9")

"Last Name": "DUMONT",

"First Name": "Jean",

"Date of Birth": "01-22-1963"

Data inside the document is enclosed by curly braces { } and each line is separated by a comma.

Data manipulation: collections and documents

Example of a document's contents:

DOCUMENT #1

```
{
        "_id": ObjectId("4efa8d2b7d284dad101e4bc9"),
        "Last Name": "DUMONT",
        "First Name": "Jean",
        "Date of Birth": "01-22-1963"
    }
```

DOCUMENT#2

```
    {
        "_id": ObjectId("4efa8d2b7d284dad101e4bc7"),
        "Last Name": "PELLERIN",
        "First Name": "Franck",
        "Date of Birth": "09-19-1983",
        "Address": "1 chemin des Loges",
        "City": "VERSAILLES"
    }
```

NoSQL <=> SQL Mapping Chart

Description

Count all records from orders

SQL Example

```
SELECT COUNT(*) AS count FROM orders
```

MongoDB Example

```
db.orders.aggregate( [ { $group: { _id: null, count: { $sum: 1 } } }] )
```

Simple Queries of the "Test" db

Enter some data into db.test:

```
> j = { fname : "Seth" }
> k = { x : 3 }
> db.test.insert(j)
> db.test.insert(k)
```

Now lets find what we entered in the database.

```
> db.test.find()
{ "_id" : ObjectId("5138ba8f211dbb394a6d160d"), "fname" : "Seth" }
{ "_id" : ObjectId("5138ba95211dbb394a6d160e"), "x" : 3 }
```

We entered 2 different documents into the database. You know this because the "_id"s are different.

To verify the collection was created you can do:

```
> show collections
```

Simple Queries of the "Test" db

To add multiple values at one time you would do something similar to this:

```
> db.test.insert( { x : 4 , j : i } )
```

Using javascript you can insert multiple documents at once:

```
> for (var i = 1; i <= 20; i++) db.test.insert( { x : 4 , j : i } )
```

Resulting in...

```
> db.test.find()
{ "_id" : ObjectId("5138ba8f211dbb394a6d160d"), "fname" : "Seth" }
{ "_id" : ObjectId("5138ba95211dbb394a6d160e"), "x" : 3 }
{ "_id" : ObjectId("5138bbe0211dbb394a6d160f"), "x" : 4, "j" : 1 }
{ "_id" : ObjectId("5138bbe0211dbb394a6d1610"), "x" : 4, "j" : 2 }
{ "_id" : ObjectId("5138bbe0211dbb394a6d1611"), "x" : 4, "j" : 3 }
{ "_id" : ObjectId("5138bbe0211dbb394a6d1612"), "x" : 4, "j" : 4 }
{ "_id" : ObjectId("5138bbe0211dbb394a6d1613"), "x" : 4, "j" : 5 }
...
Type "it" for more
>  it
{ "_id" : ObjectId("5138bbe0211dbb394a6d1621"), "x" : 4, "j" : 19 }
{ "_id" : ObjectId("5138bbe0211dbb394a6d1622"), "x" : 4, "j" : 20 }
```

Simple Queries of the "Test" db

Query for Specific Documents

```
> db.test.find({fname:"seth"}); <- Will not work because queries are case sensitive

> db.test.find({fname:"Seth"});

{ "_id" : ObjectId("5138ba8f211dbb394a6d160d"), "fname" : "Seth" }
```

Another example from a doc with multiple entries:

```
> db.test.find({j:9});

{ "_id" : ObjectId("5138bbe0211dbb394a6d1617"), "x" : 4, "j" : 9 }
```

Resources

Mainsite: http://www.mongodb.org/

Main Manual: http://docs.mongodb.org/manual/contents/

Admin Manual: http://docs.mongodb.org/manual/administration/

Introduction to Mongo: http://www.mongodb.org/about/introduction/

Wikipedia: http://en.wikipedia.org/wiki/MongoDB

Very useful Tumblr on Mongo: http://blog.mongodb.org/

SQL to Aggregation Framework Mapping Chart

http://docs.mongodb.org/manual/reference/sql-aggregation-comparison/