MongoDB

Basics

Credit where credit is due

- NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence by Pramod J. Sadalage, Martin Fowler
- MongoDB: The Definitive Guide by Kristina Chodorow
- MongoDB in Action by Kyle Banker

History

- In mid-2007, a startup in New York City called 10gen began work on a platform-as-a-service (PaaS), composed of an application server and a database, that would host web applications and scale them as needed.
  - Developers didn't feel comfortable giving up so much control over their technology stacks, but users did want 10gen's new database technology.
- MongoDB v1.0 was released in November 2009. Major releases appear approximately once every three months, with even point numbers for stable branches and odd numbers for development.

History

- It is used by a number of major websites and services, including Craigslist, eBay, Foursquare, SourceForge, Viacom, and the New York Times, among others.
- MongoDB is written in C++ and actively developed by MongoDB Inc. The project compiles on all major operating systems, including Mac OS X, Windows, Solaris, and most flavors of Linux.
- As per Wikipedia, MongoDB is the most popular NoSQL database.
- Latest stable release is 2.6

- Document based database.

- The primary reason for moving away from relational model is to make scaling easier but there are some other advantages too:
  - A document oriented database replaces the concept of a "row" with a more flexible model, the document.
  - No Schema. There are no predefined schemas, a document's keys and values are not of fixed types or sizes

- Just like Couch DB, indexes in MongoDB are implemented as a B-tree data structure.

- MongoDB automatically takes care of balancing data and load across a cluster, redistributing documents automatically and routing user requests to the correct machines.
  - This allows developers to focus on programming the application, not scaling it.

- When a cluster needs more capacity, new machines can be added and MongoDB will figure out how the existing data should be spread to them.

Consistency and Transactions

- No multi document atomic transactions are supported.

- CAP theorem: Consistency and Availability compete with each other.

Terminology

- A *document* is the basic unit of data for MongoDB and is roughly equivalent to a row in a relational database management system (but much more expressive).

- a *collection* can be thought of as a table with a dynamic schema.

- A single instance of MongoDB can host multiple independent *databases*, each of which can have its own collections.

- Every document has a special key, "_id", that is unique within a collection.

Documents

- Self defining, with hierarchical structure. (like XML or JSON).

- Document can vary in structure, even in the same collection.

- You can add attributes to new documents in a collection without having to change the existing ones in the collection.

```
{
  _id: ObjectID('4bd9e8e17cefd644108961bb'),
  title: 'Adventures in Databases',
  url: 'http://example.com/databases.txt',
  author: 'msmith',
  vote_count: 20,
  tags: ['databases', 'mongodb', 'indexing'],
  image: {
    url: 'http://example.com/db.jpg',
    caption: 'A database.',
    type: 'jpg',
    size: 75381,
    data: "Binary"
  },
  comments: [
    {
      user: 'bjones',
      text: 'Interesting article.'
    },
    {
      user: 'sverch',
      text: 'Color me skeptical!'
    }
  ]
}
```

Documents

- Are an ordered set of keys and keys with associated values.

  Example:

  {"greeting" : "Hello, world!", "foo" : 3}

- Values in documents can be of different data types. In the above example value for "greeting" is "Hello, world!", which is a string, where the value for "foo" is an integer.

Documents

The keys in a document are strings. Any UTF-8 character is allowed in a key, with some exceptions:

- Keys must not contain the character \0 (the null character). This character is used to signify the end of a key.
- The . and $ characters have some special properties and should be used only in certain circumstances. In general, they should be considered reserved, and drivers will complain if they are used inappropriately.

Documents

- MongoDB is type-sensitive

```
{"foo" : 3}
{"foo" : "3"}
```

and case sensitive:

```
{"foo" : 3}
{"Foo" : 3}
```

Documents

- MongoDB cannot contains duplicate keys. Following is not legal:

```
{"greeting" : "Hello, world!", "greeting" : "Hello,
    MongoDB!"}
```

- Field order does not usually matter and you should not design your schema to depend on a certain ordering of fields (MongoDB may reorder them).

Collections

- A collection is a group of documents. Similar to a table.

- Dynamic schemas:

```
{"greeting" : "Hello, world!"}
{"foo" : 5}
```

Both can be stored in a single collection.

Why do we need separate collections at all?

Collections

- Keeping different kind of documents in the same collection can be messy. For example, if we are querying for blog posts, it is a hassle to weed out documents that have author data.

- Grouping documents of the same kind together in the same collection allows for data locality. Getting several blog posts from a collection containing only posts will likely require fewer disk seeks than getting the same posts from a collection containing posts and author data.

- By putting only documents of a single type into the same collection, we can index our collections more efficiently.

Collections

Collection names can be any UTF-8 string, with a few restrictions:

- The empty string ("") is not a valid collection name.
- Collection names may not contain the character \0 (the null character).
- You should not create any collections that start with *system.*, a prefix reserved for internal collections.
- User-created collections should not contain the reserved character $ in the name.

Subcollections

One convention for organizing collections is to use namespaced sub-collections separated by the . character.

Example:

an application containing a blog might have a collection named *blog.posts* and a separate collection named *blog.authors*.

Note: This is for organizational purposes only

Databases

- A single instance of MongoDB can host several databases, each grouping together zero or more collections.
- A database has its own permissions, and each database is stored in separate files on disk.
- Separate databases are useful when storing data for several application or users on the same MongoDB server.

Databases

Reserved Database names

- Admin:
  - If a user is added to the *admin* database, the user automatically inherits permissions for all databases.
- Local:
  - This database will never be replicated and can be used to store any collections that should be local to a single server
- Config:
  - When MongoDB is being used in a sharded setup, it uses the *config* database to store information about the shards.

When to use MongoDB?

- Medical records and other large document systems.
- Read heavy environments like analytics and mining.
- Partnered with relational databases
  - Relational for live data
  - Mongo for huge largely read only archives
- Online applications
- Massively wide e-commerce (Blogs).

Let’s install MongoDB

- Current Stable Release: 2.6

- Installing on Mac:
  - brew install mongodb
  - brew install mongodb --with-openssl

Installing MongoDB on Windows

- **MongoDB for Windows 64-bit** runs only on Windows Server 2008 R2, Windows 7 64-bit, and newer versions of Windows.

- **MongoDB for Windows 32-bit** runs on any 32-bit version of Windows newer than Windows Vista. 32-bit versions of MongoDB only support databases smaller than 2GB.

- **MongoDB for Windows 64-bit Legacy** runs on Windows Vista, Windows Server 2003, and Windows Server 2008 and does not include recent performance enhancements.

Note: Windows XP is not supported

Installing MongoDB on Windows

- Download MongoDB msi file from:

https://www.mongodb.org/downloads

You may specify an installation directory if you choose the “Custom” installation option. By default, MongoDB is installed to C:\mongodb.

Run MongoDB

- Before you start MongoDB for the first time, create the directory to which the mongod process will write data. By default, the mongod process uses the /data/db directory.

- Make sure location of binaries in the PATH variable:

  export PATH=<mongodb-install-directory>/bin:$PATH

- If your system PATH variable includes the location of the mongod binary and if you use the default data directory (i.e., /data/db), simply enter mongod at the system prompt:

  > mongod

- Or you can specify a default data directory:

  > mongod --dbpath <path to data directory>

Run MongoDB

- To connect to MongoDB through the shell, open another **Command Prompt** and give the command : mongo.exe (Windows) or mongo (Mac)

- To stop MongoDB, press Control+C

- GUI:

http://docs.mongodb.org/ecosystem/tools/administration-interfaces/