

CSCI403

Lecture 36: NoSQL, Distributed DBs, DBs in the Cloud

So you want a
database...

Imagine “Relational” Doesn’t Exist





mongoDB

```
{name: "mongo", type: "DB"}
```

MongoDB (from "humongous") is a scalable, high-performance, open source, document-oriented database. Written in C++.

<http://www.mongodb.org/>

MapReduce?

Google's patented version of functional programming's map and reduce.

JSON?



JSON (JavaScript Object Notation) is a **lightweight data-interchange format**. It is **easy for humans to read and write**. It is **easy for machines to parse and generate**. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a **text format** that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal **data-interchange language**.

JSON

```
{  
  "chicken": {  
    "name": "howard",  
    "age": 32,  
    "chicks": [  
      {"name": "larry"},  
      {"name": "curly"},  
      {"name": "moe"}  
    ]  
  }  
}
```

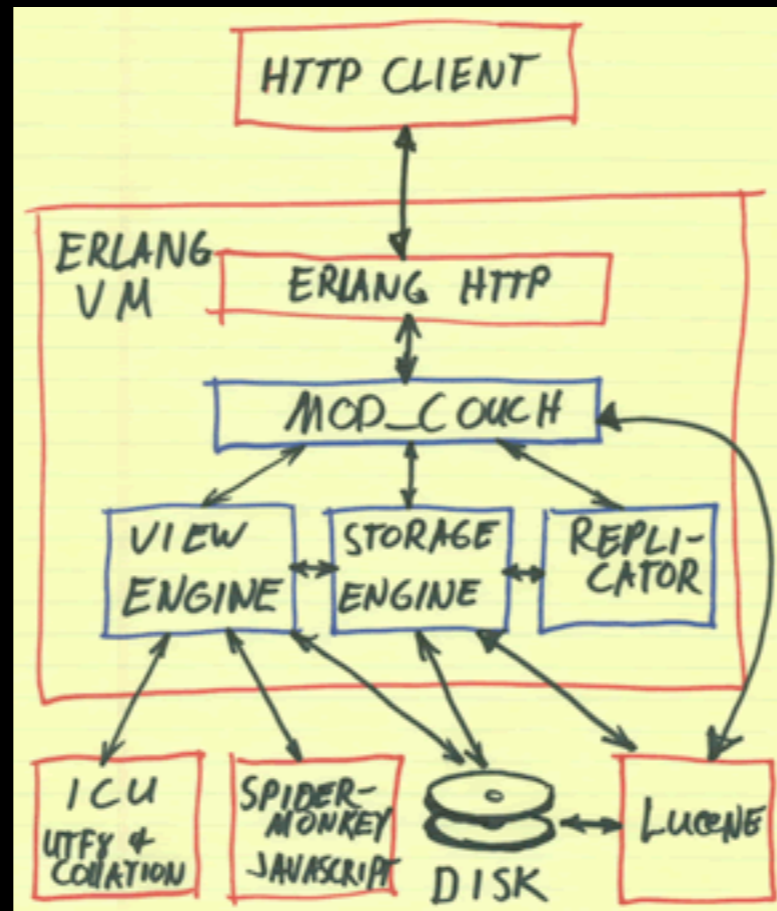
JSON

```
{
  "id": "0001",
  "type": "donut",
  "name": "Cake",
  "ppu": 0.55,
  "batters":
    {
      "batter":
        [
          { "id": "1001", "type": "Regular" },
          { "id": "1002", "type": "Chocolate" },
          { "id": "1003", "type": "Blueberry" },
          { "id": "1004", "type": "Devil's Food" }
        ]
      },
    "topping":
      [
        { "id": "5001", "type": "None" },
        { "id": "5002", "type": "Glazed" },
        { "id": "5005", "type": "Sugar" },
        { "id": "5007", "type": "Powdered Sugar" },
        { "id": "5006", "type": "Chocolate with Sprinkles" },
        { "id": "5003", "type": "Chocolate" },
        { "id": "5004", "type": "Maple" }
      ]
    }
}
```



- Document-oriented DB
- RESTful, JSON API
- Schemaless
- Distributed
- Query language: JavaScript

(Document-oriented. Not *intended* for object persistence.)



<http://couchdb.apache.org/docs/intro.html>

<http://www.couchbase.com/>

erlang?

“Erlang is a programming language used to build massively scalable soft real-time systems with requirements on high availability. Some of its uses are in telecoms, banking, e-commerce, computer telephony and instant messaging. Erlang's runtime system has built-in support for concurrency, distribution and fault tolerance.”

<http://erlang.org>

<http://www.youtube.com/watch?v=uKfKtXYLG78>

(originally developed at Ericsson)

```
> webput:publish("bingo.baz.dobedo.se", "jimbo23", "waX2p34",  
                "/home/joe/html/mirror",  
                "/pub/acct2754/html").
```

ok

```
01 -module(webput).  
02  
03 -doc([{author, 'Joe Armstrong'},  
04       {title, "Publish data on a web site"},  
05       {keywords, [web, www, home, page, publish]},  
06       {date, 981112}]).  
07  
08 -export([publish/5]).  
09  
10 publish(Host, User, Password, LocalDir, RemoteDir) ->  
11     case ftp:open(Host) of  
12     {ok, Pid} ->  
13         case ftp:user(Pid, User, Password) of  
14         ok ->  
15             case ftp:cd(Pid, RemoteDir) of  
16             ok ->  
17                 case file:list_dir(LocalDir) of  
18                 {ok, Files} ->  
19                     lists:foreach(fun(I) ->  
20                                 publish(I, LocalDir, Pid)  
21                                 end, Files);  
22                 {error, _} ->  
23                     exit({bad, local, directory, LocalDir})  
24                 end;  
25                 {error, Reason} ->  
26                     exit({cannot, cd, to, RemoteDir, reason, Reason})  
27                 end;  
28                 {error, Reason} ->  
29                     exit({cannot, login, as, User, reason, Reason})  
30                 end;  
31                 {error, Reason} ->  
32                     exit({cannot, connect, to, Host, reason, Reason})  
33                 end.  
34  
35     publish(File, Dir, Pid) ->  
36         LocalFile = Dir ++ "/" ++ File,  
37         case ftp:send(Pid, LocalFile, File) of  
38         ok ->  
39             ok;  
40         {error, Reason} ->  
41             exit({cannot, send, file, File, reason, Reason})  
42         end.
```

```
count_x:file("count_x.erl").
```

```
-module(count_x).  
-author('joe@cslab.ericssons.se').  
  
%% count the number of x's in a file  
  
-export([file/1]).  
  
file(F) ->  
    lists:foldl(fun($x,N) -> N + 1;  
                (_, N) -> N  
                end,  
                0,  
                binary_to_list(element(2, file:read_file(F)))).
```

RESTful?

REpresentational State Transfer

HTTP: post, get, put, delete

CRUD: create, read, update, delete

<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

Redis



Redis is an open source, advanced **key-value** store. It is often referred to as a data structure server since keys can contain strings, hashes, lists, sets and sorted sets.

<http://redis.io/>

<http://try.redis-db.com/>

Riak



Distributed, fault-tolerant database system.

Written in Erlang and C.

Based on Amazon's "Dynamo" architecture.

<http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf>

<http://wiki.basho.com/>

Cassandra



Cassandra

Based on BigTable and Dynamo
Key-Value store
Distributed
“eventually consistent”



Eventually?

“the storage system guarantees that if no new updates are made to the object, eventually all accesses will return the last updated value.”

Simple example: MySQL Master-Slave replication

A design trade-off between availability & consistency.

<http://queue.acm.org/detail.cfm?id=1466448>

Hosting a DB Server

- Self-managed
- Colocated hardware
- Third-party managed
 - Shared host
 - Dedicated host
 - Virtual Dedicated
 - “Cloud”

Cloud-Based Services

- Amazon SimpleDB & RDS
- IrisCouch
- MongoHQ & MongoMachine
- So many more...