Apache CouchDB

Who's Talking?

- Chris Anderson
- jchris@apache.org / @jchris
- Apache CouchDB Committer
- REST and JavaScript enthusiast
- Director, couch.io

Who Are You?

couch.io

Who Are You?

Easier way to make web applications

Scalable key/value store

Peer-based Replication

Append-only IO pattern

Agenda

- Introduction
- Installation
- API by Example
- Design Documents
- CouchApps
- Advanced Views
- Replication
- Distributed Systems
- Deployment

Agenda

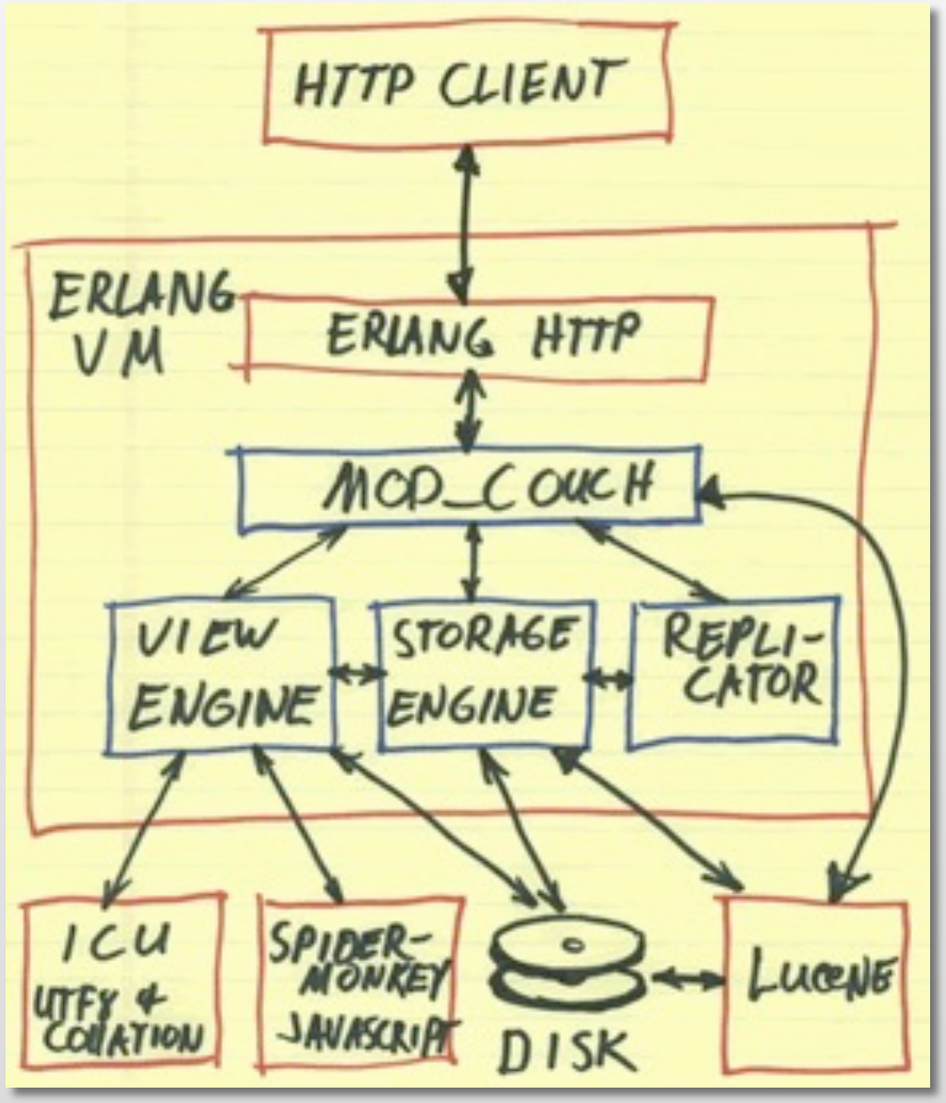
- **Introduction**
- Installation
- API by Example
- Design Documents
- CouchApps
- Advanced Views
- Replication
- Distributed Systems
- Deployment

Relax

couch.io

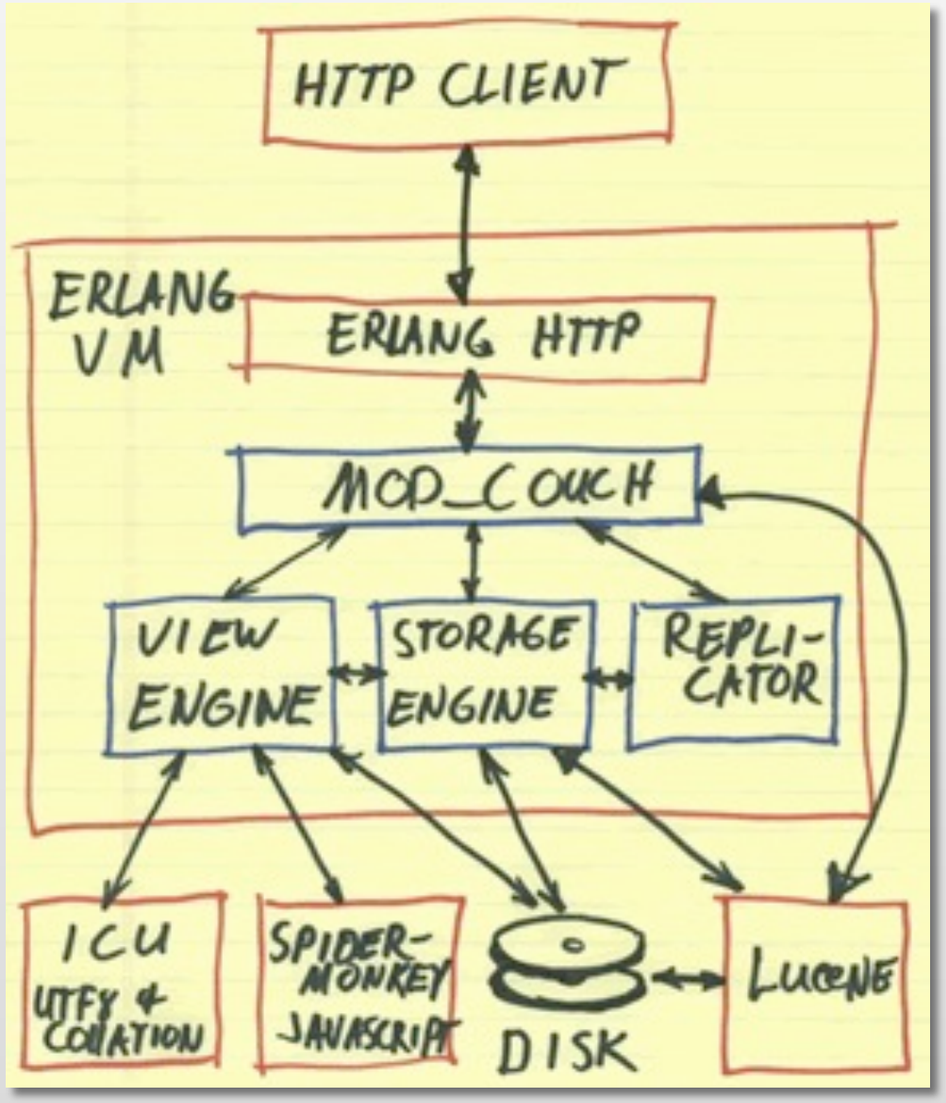
Features

- Schema Free (JSON)
- Document Oriented,
  Not Relational
- Highly Concurrent
- RESTful HTTP API
- JavaScript Powered
  Map/Reduce Views
- N-Master Replication
- Robust Storage



Features

- **Schema Free (JSON)**
- Document Oriented,
  Not Relational
- Highly Concurrent
- RESTful HTTP API
- JavaScript Powered
  Map/Reduce Views
- N-Master Replication
- Robust Storage



Schema Free (JSON)

```
{
  "_id": "BCCD12CBB",
  "_rev": "1-AB764C",

  "type": "person",
  "name": "Darth Vader",
  "age": 63,
  "headware":
    ["Helmet", "Sombrero"],
  "dark_side": true
}
```

Schema Free (JSON)

```
{
  "_id": "BCCD12CBB",
  "_rev": "2-AB764C",

  "type": "person",
  "name": "Darth Vader",
  "age": 63,
  "headware":
    ["Helmet", "Sombrero"],
  "dark_side": true
}
```

Schema Free (JSON)

```
{
  "_id": "BCCD12CBB",
  "_rev": "3-AB764C",

  "type": "person",
  "name": "Darth Vader",
  "age": 63,
  "headware":
    ["Helmet", "Sombrero"],
  "dark_side": true
}
```

Schema Free (JSON)

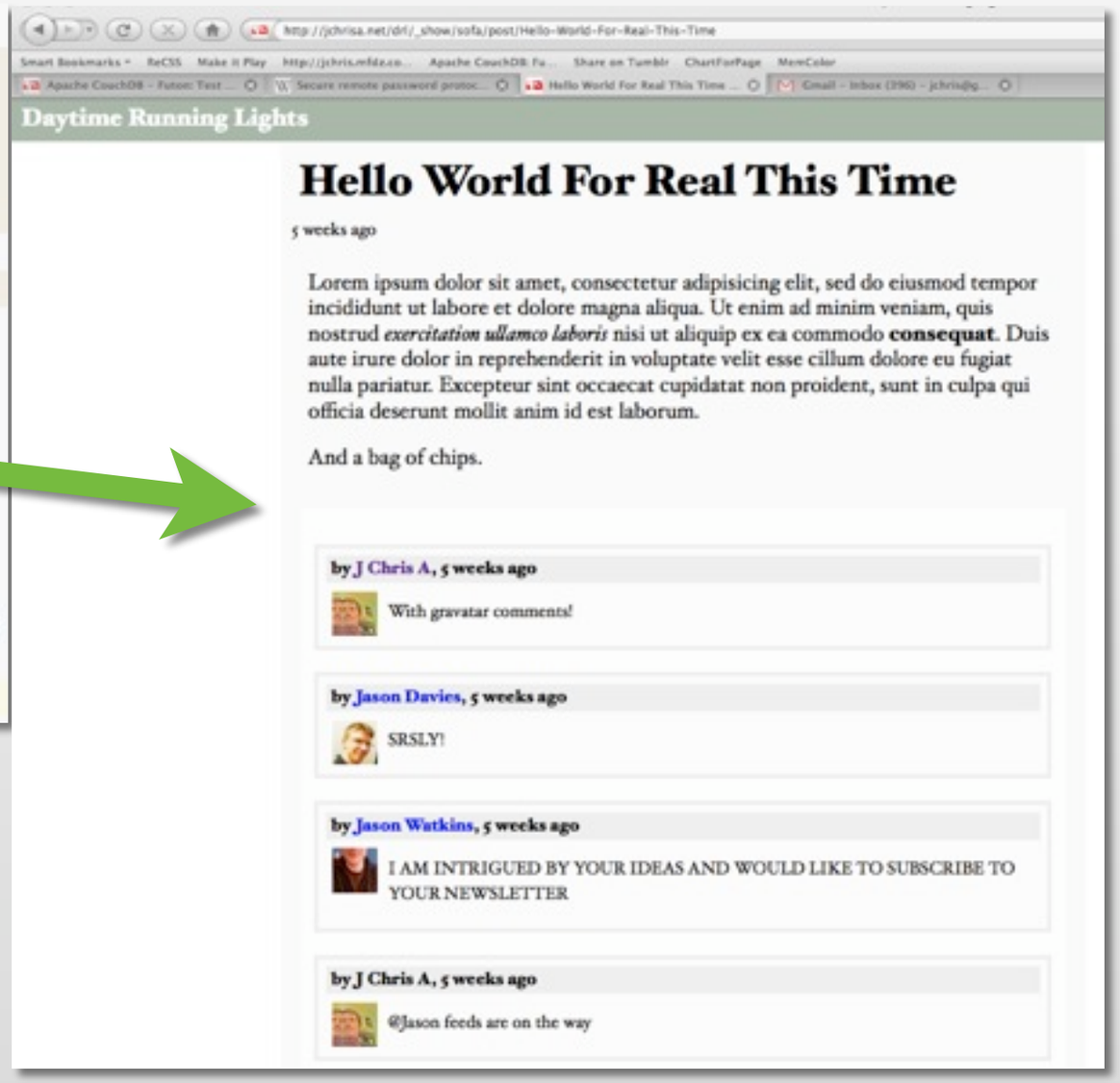
```
{
  "_id": "BCCD12CBB",
  "_rev": "3-AB764C",

  "type": "person",
  "name": "Darth Vader",
  "age": 63,
  "headware":
    ["Helmet", "Sombrero"],
  "dark_side": true
}
```

Render JSON as HTML

shows/post.js

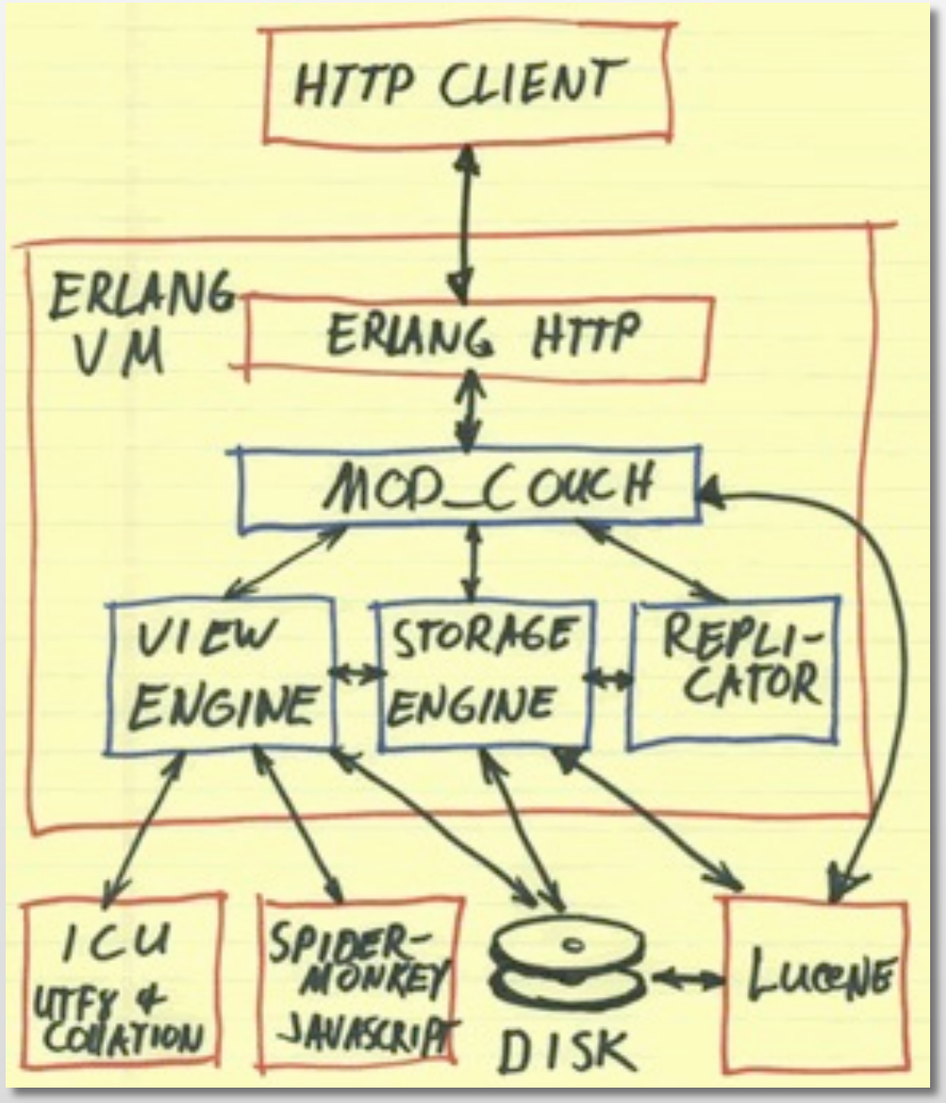
```
function(doc, req) {
  // !json templates.post
  // !json blog
  // !code helpers.template
  // !code helpers.couchapp
  // log(req.headers.Accept);

  // we only show html
  return template(templates.post, {
    title : doc.title,
    blogName : blog.title,
    post : doc.html,
    date : doc.created_at,
    author : doc.author,
    assets : assetPath(),
    editPostPath : showPath('edit', doc._id),
    index : listPath('index','recent-posts',{descending:true, limit:8})
  });
}
```

/drl/_show/sofa/post/Hello-World-For-Real-This-Time



Features

- Schema Free (JSON)
- **Document Oriented,**
  Not Relational
- Highly Concurrent
- RESTful HTTP API
- JavaScript Powered Map/Reduce Views
- N-Master Replication
- Robust Storage



Document Oriented

Not Relational

- Documents in the Real World™
  - Bills, letters, tax forms…
  - Same type != same structure
  - Can be out of date (so what?)
  - No references

Document Oriented

Not Relational

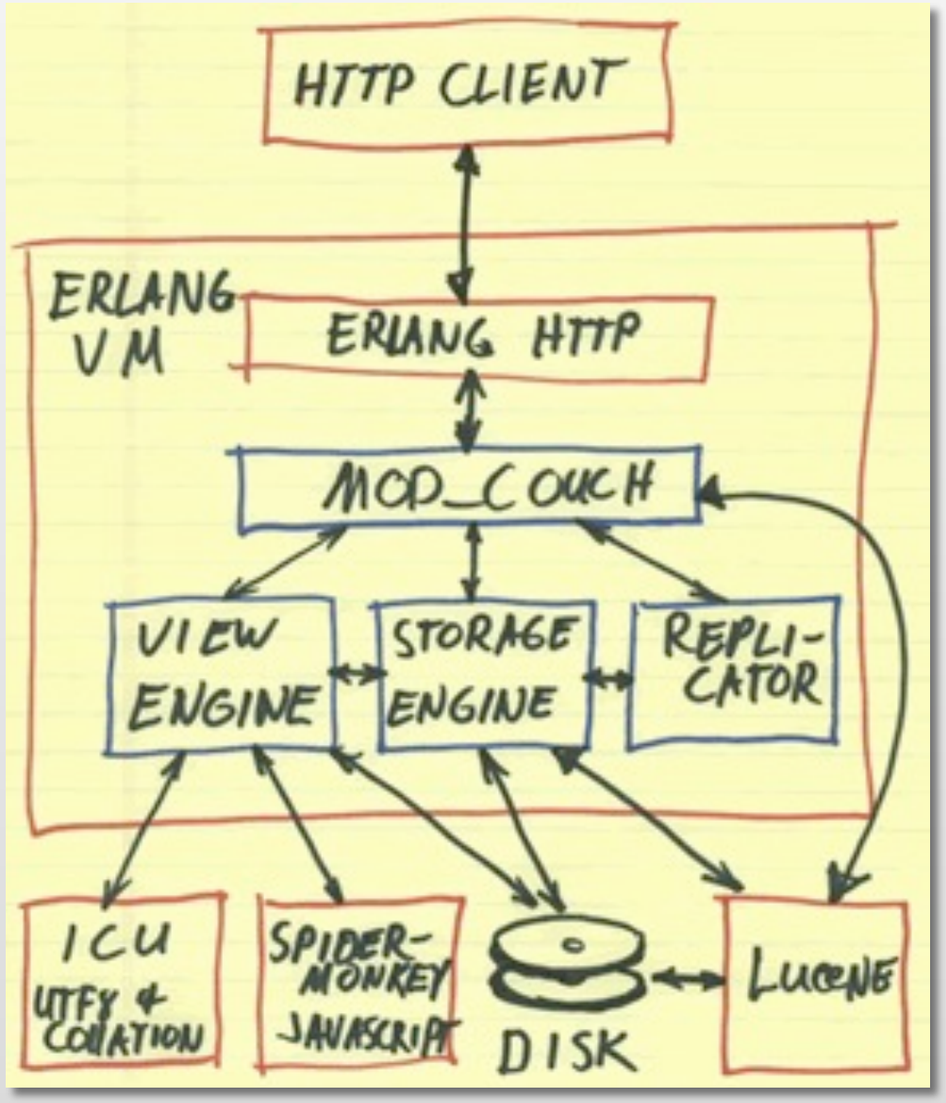
- Documents in the Real World™
  - Bills, letters, tax forms…
  - Same type != same structure
  - Can be out of date (so what)
- No references

Natural Data

Behaviour

Features

- Schema Free (JSON)
- Document Oriented,
  Not Relational
- **Highly Concurrent**
- RESTful HTTP API
- JavaScript Powered
  Map/Reduce Views
- N-Master Replication
- Robust Storage



Highly Concurrent

Erlang Praising

couch.io

Highly Concurrent

Erlang Praising

- Parallelism built into the language

Highly Concurrent

Erlang Praising

- Parallelism built into the language
- Makes programming multi-core easy(er)

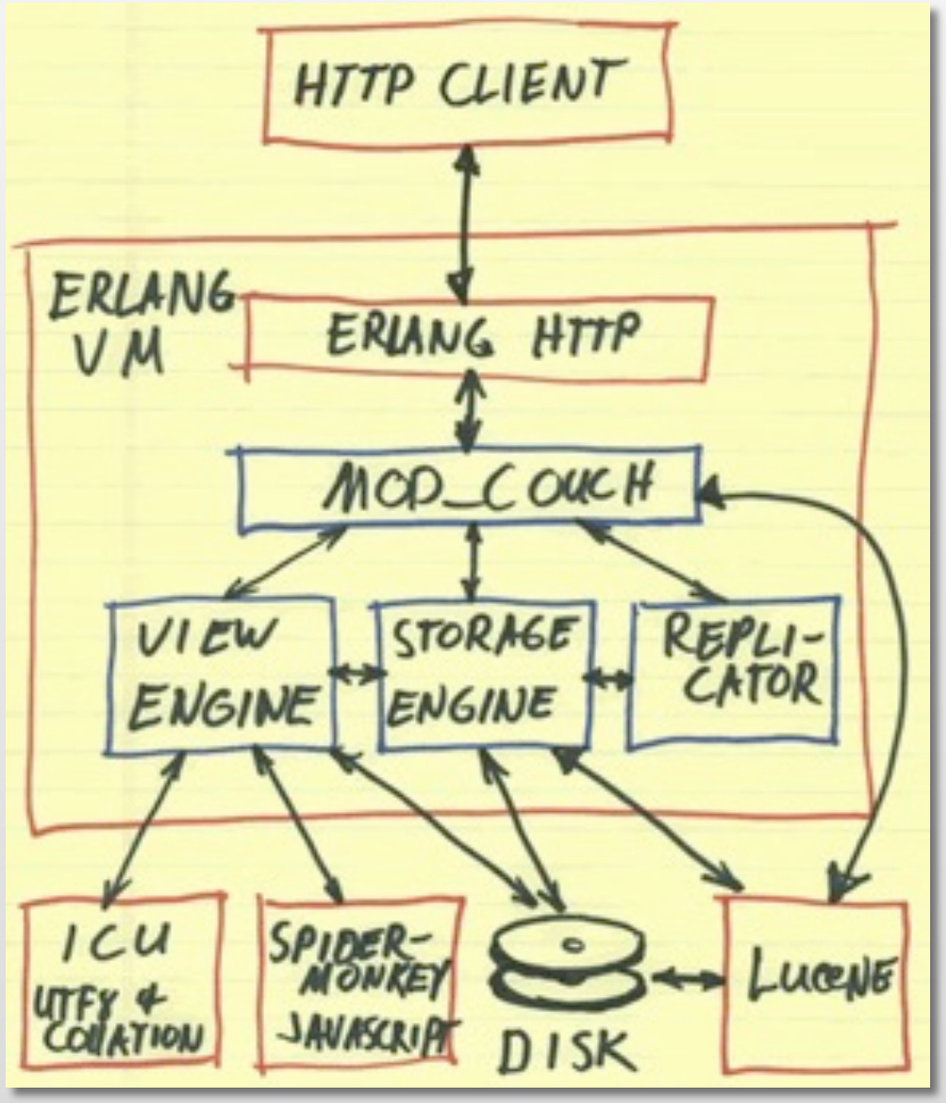
Highly Concurrent

Erlang Praising

- Parallelism built into the language
- Makes programming multi-core easy(er)
- Easy to create fault-tolerant systems

Features

- Schema Free (JSON)
- Document Oriented,
  Not Relational
- Highly Concurrent
- **RESTful HTTP API**
- JavaScript Powered
  Map/Reduce Views
- N-Master Replication
- Robust Storage



REST*ful* HTTP API
CRUD

- **C**reate
  HTTP PUT /db/mydocid
- **R**ead
  HTTP GET /db/mydocid
- **U**pdate
  HTTP PUT /db/mydocid
- **D**elete
  HTTP DELETE /db/mydocid

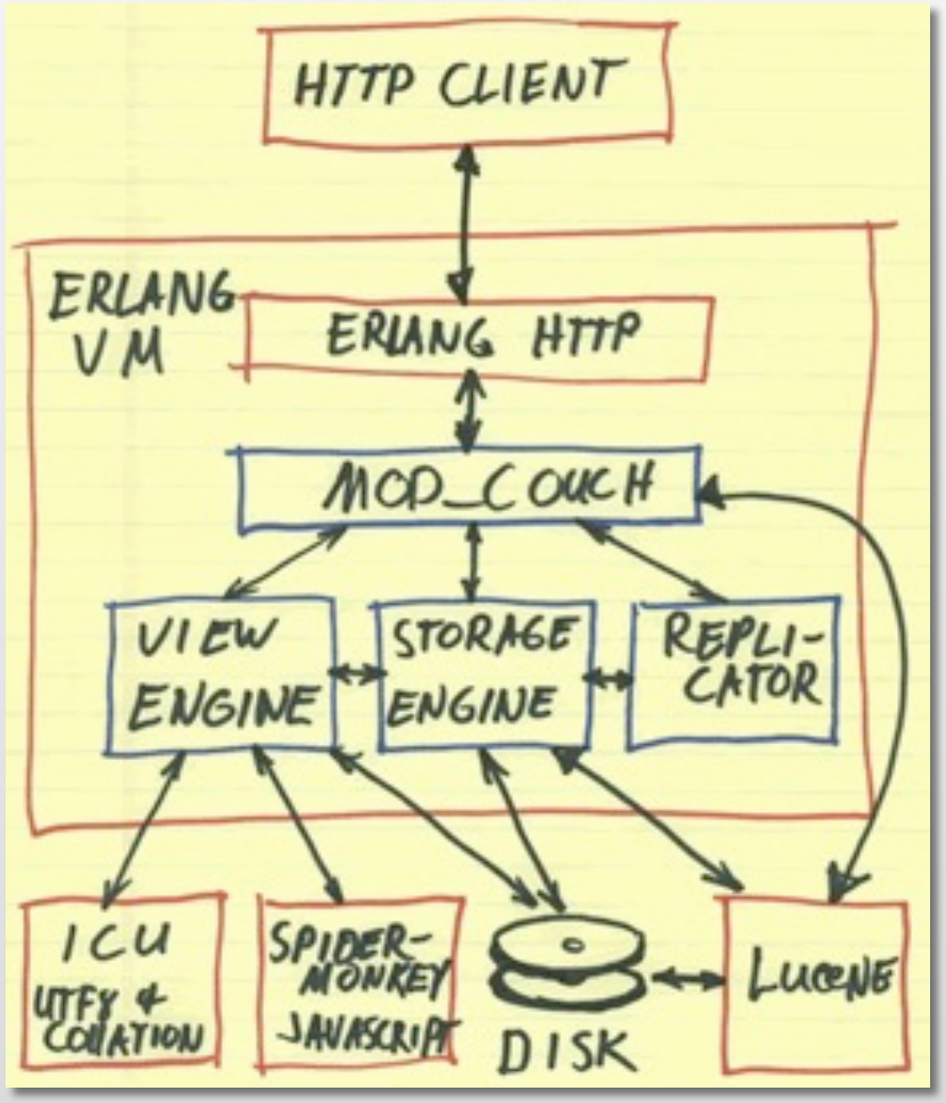
RESTful HTTP API

Relax

```
couch = CouchRest.database!("http://
127.0.0.1:5984/tweets")

tweets_url = "http://twitter.com/statuses/
user_timeline.json"
tweets = http.get(tweets_url)

couch.bulk_save(tweets)
```

Features

- Schema Free (JSON)
- Document Oriented,
  Not Relational
- Highly Concurrent
- RESTful HTTP API
- **JavaScript Powered Map/Reduce Views**
- N-Master Replication
- Robust Storage



JavaScript powered Map/Reduce

- Map functions extract data from your documents
- Reduce functions aggregate intermediate values
- The kicker: Incremental b-tree storage

Map Reduce Views

Docs


```
{"user" : "Chris",
    "points" : 3 }
  {"user": "Joe",
  "points" : 10 }
{"user": "Alice",
    "points" : 5 }
{"user": "Mary",
    "points" : 9}
{"user": "Bob",
    "points": 7}
```


Map

```
function(doc) {
    if (doc.user && doc.points) {
        emit(doc.user, doc.points);
    }
}
```


```
{"key": "Alice", "value": 5}
 {"key": "Bob", "value": 7}
{"key": "Chris", "value": 3}
 {"key": "Joe", "value": 10}
{"key": "Mary", "value": 9}
```


Reduce

```
function(keys, values, rereduce) {
  return sum(values);
}
```

Alice ... Chris: 15
Everyone: 34

Map Reduce Views

Docs

```
{"user" : "Chris",
    "points" : 3 }
  {"user": "Joe",
   "points" : 10 }
 {"user": "Alice",
    "points" : 5 }
{"user": "Mary",
    "points" : 9}
 {"user": "Bob",
    "points": 7}
```

Map

```
function(doc) {
    if (doc.user && doc.points) {
        emit(doc.user, doc.points);
    }
}
```

```
{"key": "Alice", "value": 5}
 {"key": "Bob", "value": 7}
{"key": "Chris", "value": 3}
 {"key": "Joe", "value": 10}
{"key": "Mary", "value": 9}
```

Reduce

```
function(keys, values, rereduce) {
  return sum(values);
}
```

```
Alice … Chris: 15
      Everyone: 34
```

Map Reduce Views

Docs

```
{"user" : "Chris",
    "points" : 3 }
  {"user": "Joe",
   "points" : 10 }
 {"user": "Alice",
    "points" : 5 }
  {"user": "Mary",
    "points" : 9}
 {"user": "Bob",
    "points": 7}
```

Map

```
function(doc) {
   if (doc.user && doc.points) {
      emit(doc.user, doc.points);
   }
}
```

```
{"key": "Alice", "value": 5}
 {"key": "Bob", "value": 7}
{"key": "Chris", "value": 3}
 {"key": "Joe", "value": 10}
{"key": "Mary", "value": 9}
```

Reduce

```
function(keys, values, rereduce) {
  return sum(values);
}
```

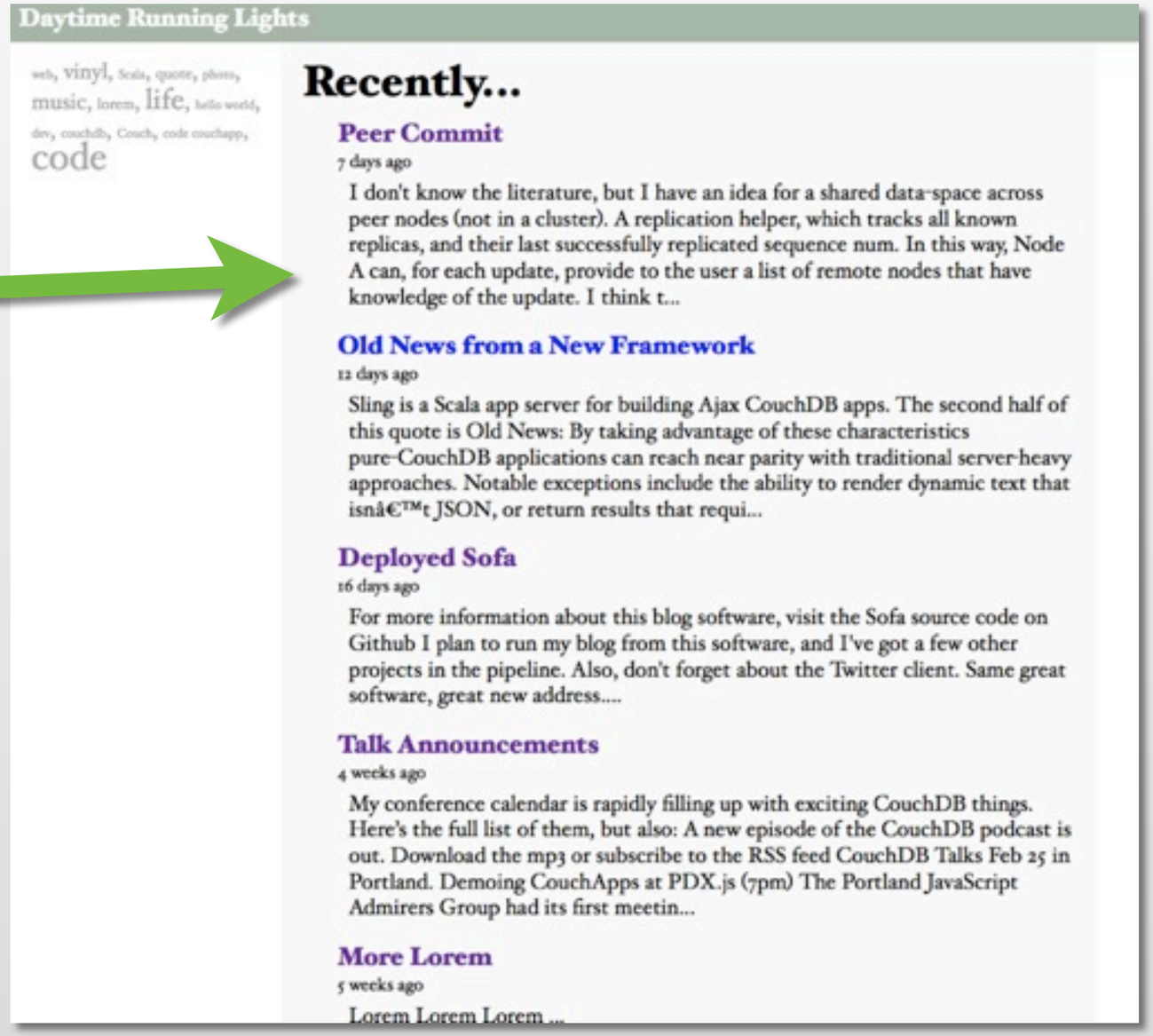
```
Alice … Chris: 15
Everyone: 34
```

Render Views as HTML

lists/index.js

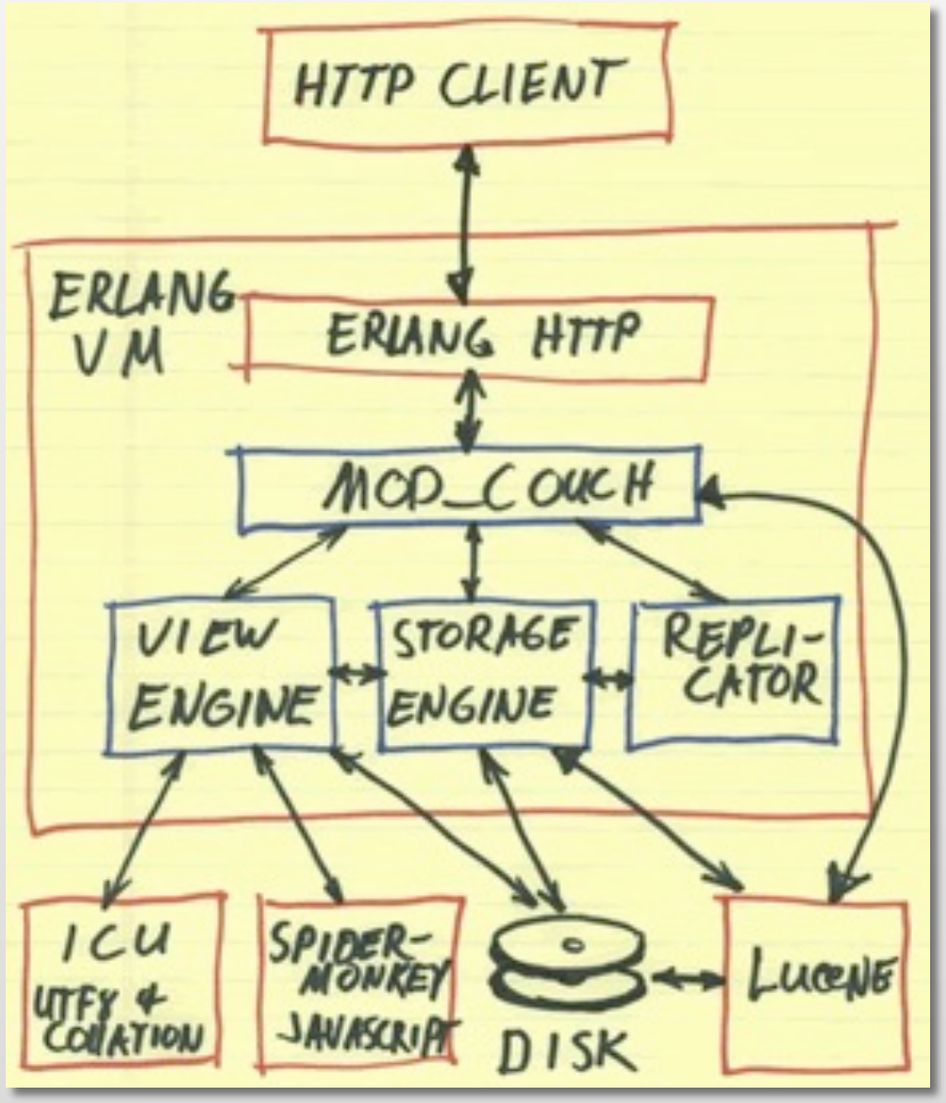
```
function(head, row, req) {
  // !json templates.index
  // !json blog
  // !code helpers.couchapp
  // !code helpers.template
  // log(req.headers.Accept);
  var indexPath = listPath('index','recent-posts',{descending:true, limit:8});
  var feedPath = listPath('index','recent-posts',{descending:true, limit:8, format:"atom"});
  return respondWith(req, {
    html : function() {
      if (head) {
        return template(templates.index.head, {
          title : blog.title,
          newPostPath : showPath("edit"),
          index : indexPath,
          assets : assetPath()
        });
      } else if (row) {
        var post = row.value;
        return template(templates.index.row, {
          title : post.title,
          summary : post.summary,
          date : post.created_at,
          link : showPath('post', row.id),
          assets : assetPath()
        });
      } else {
        return template(templates.index.tail, {
          assets : assetPath()
        });
      }
    },
    atom : function() {
      // with first row in head you can do updated.
      if (head) {
        var f = <feed xmlns="http://www.w3.org/2005/Atom"/>;
        f.title = blog.title;
        f.id = makeAbsolute(req, indexPath);
        f.link.@href = makeAbsolute(req, feedPath);
        f.link.@rel = "self";
        f.generator = 'Sofa on CouchDB';
        f.updated = new Date().rfc3339();
        return {body:f.toXMLString().replace(/\<\/feed\>/,'')};
      } else if (row) {
        var entry = <entry/>;
        entry.id = makeAbsolute(req, '/'+encodeURIComponent(req.info.db_name)+'/'+encodeURICom
        entry.title = row.value.title;
        entry.content = row.value.summary;
        entry.content.@type = 'html';
        entry.updated = new Date(row.value.created_at).rfc3339();
        entry.author = <author><name>{row.value.author}</name></author>;
        entry.link.@href = makeAbsolute(req, showPath('post', row.id));
        entry.link.@rel = "alternate";
        return {body:entry};
      } else {
        return {body : "</feed>"};
      }
    }
  })
};
```

/drl/_list/sofa/index/recent-posts?descending=true&limit=8

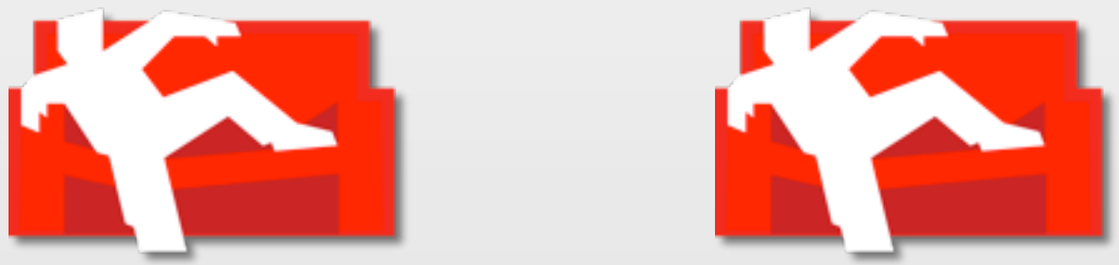


Features

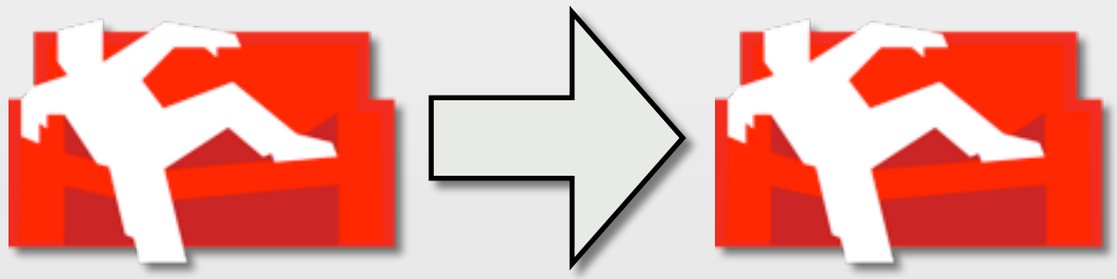
- Schema Free (JSON)
- Document Oriented,
  Not Relational
- Highly Concurrent
- RESTful HTTP API
- JavaScript Powered
  Map/Reduce
- **N-Master Replication**
- Robust Storage

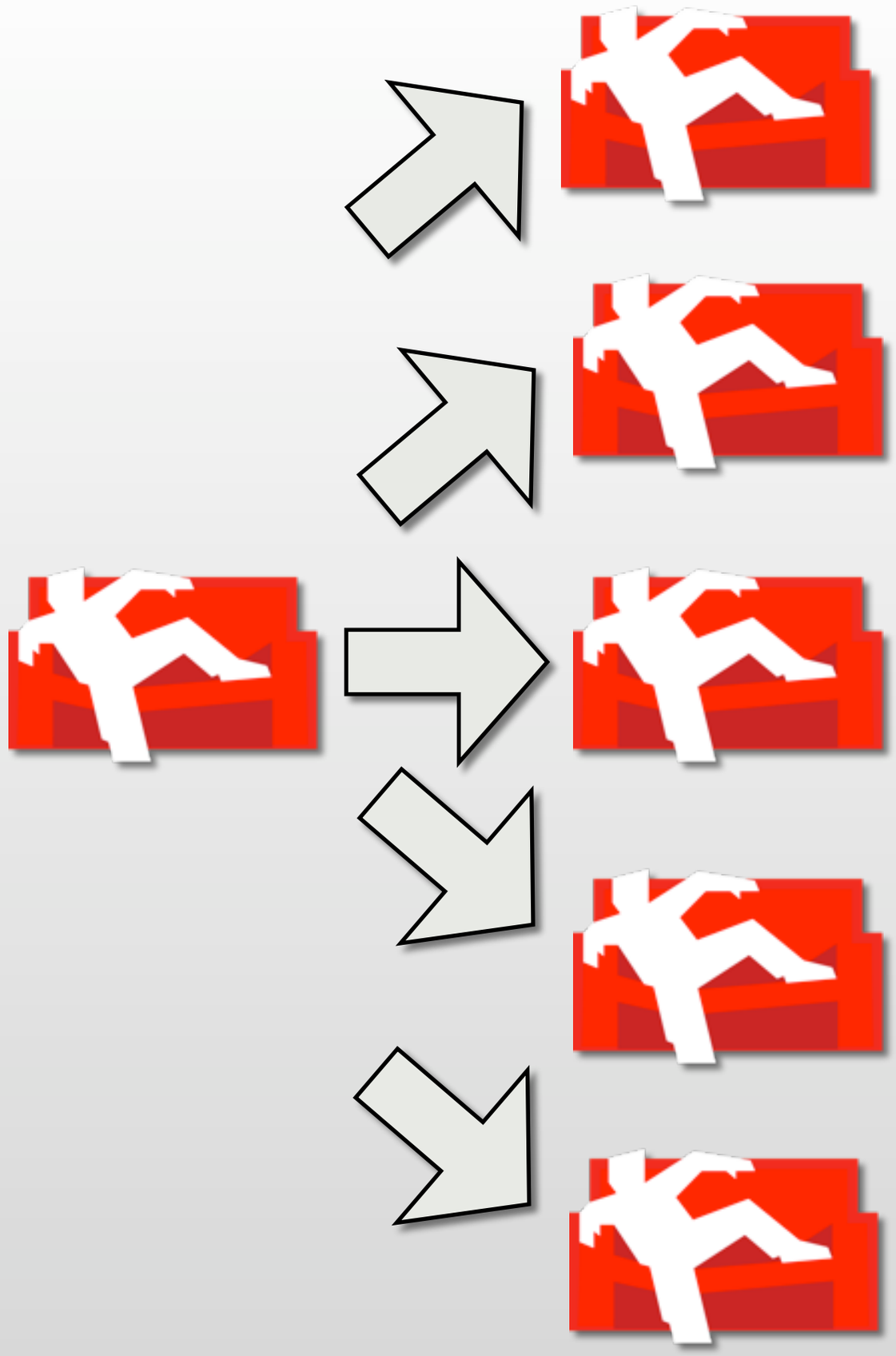


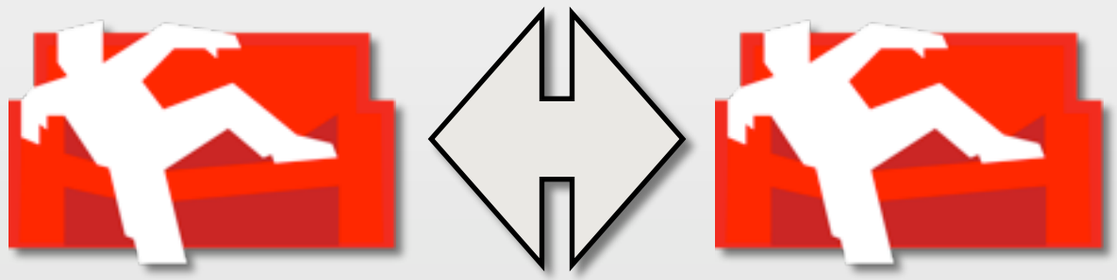
couch.io

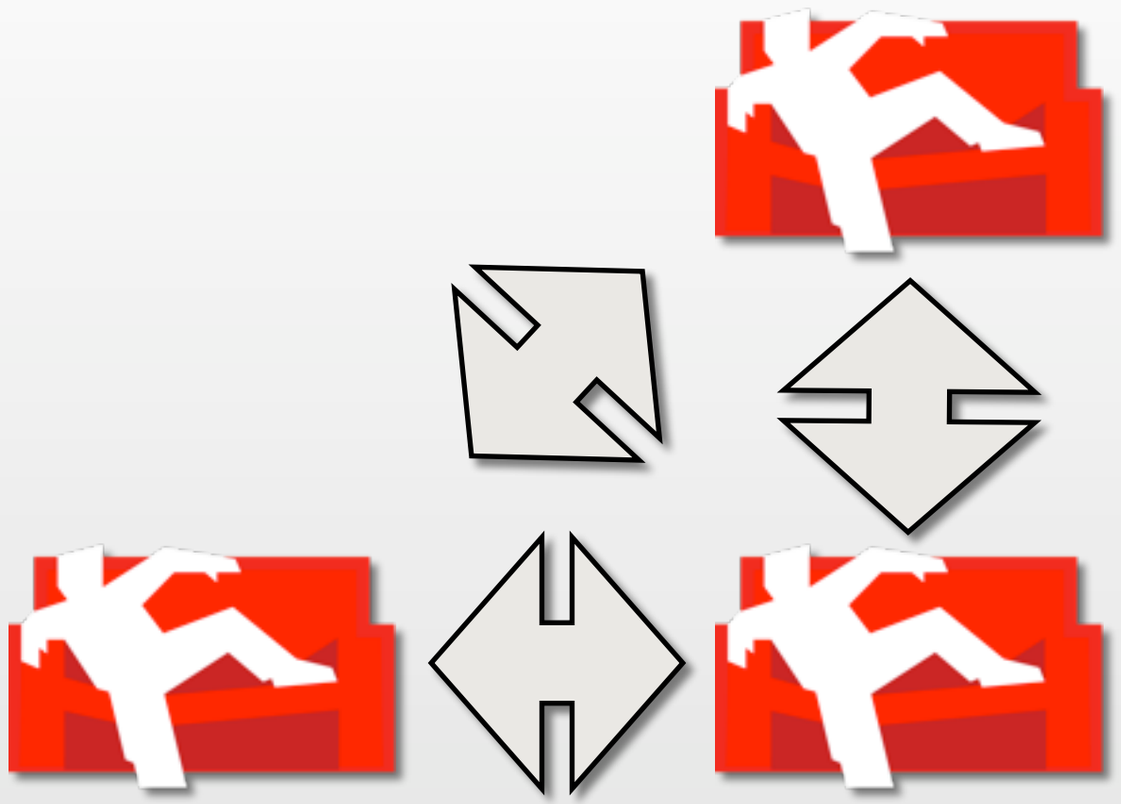


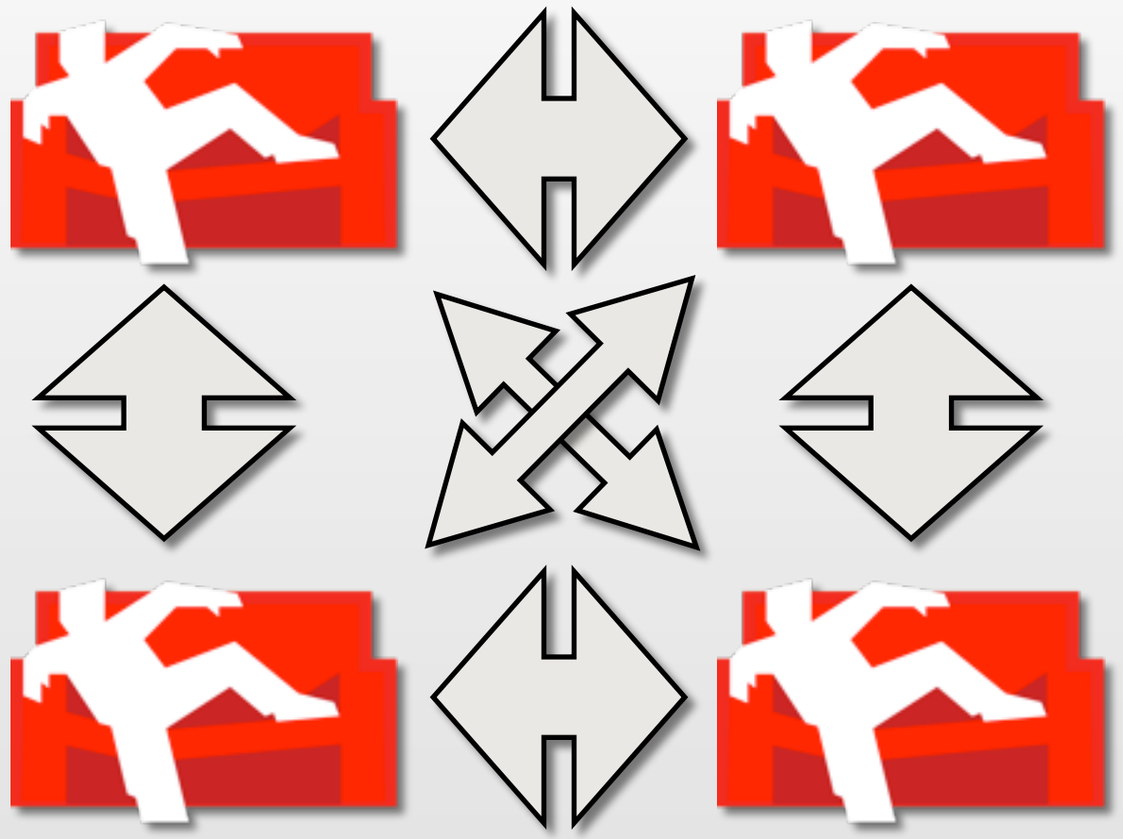
couch.io

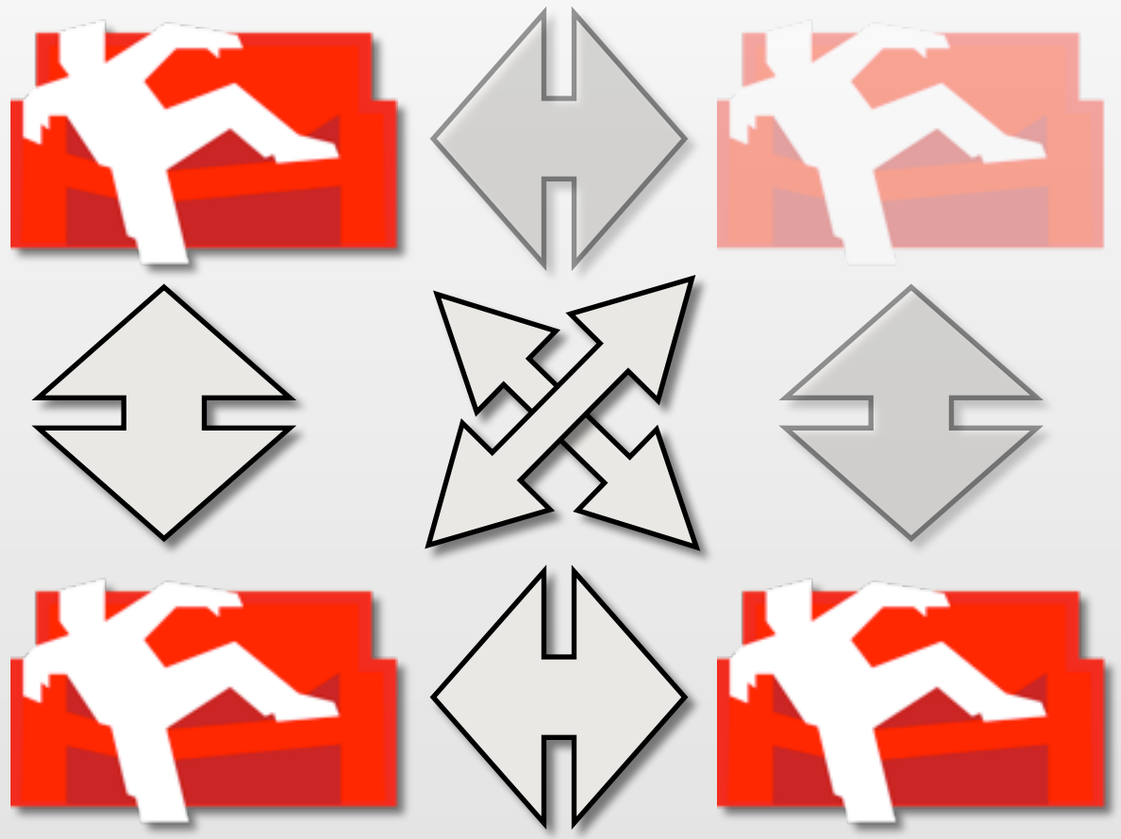


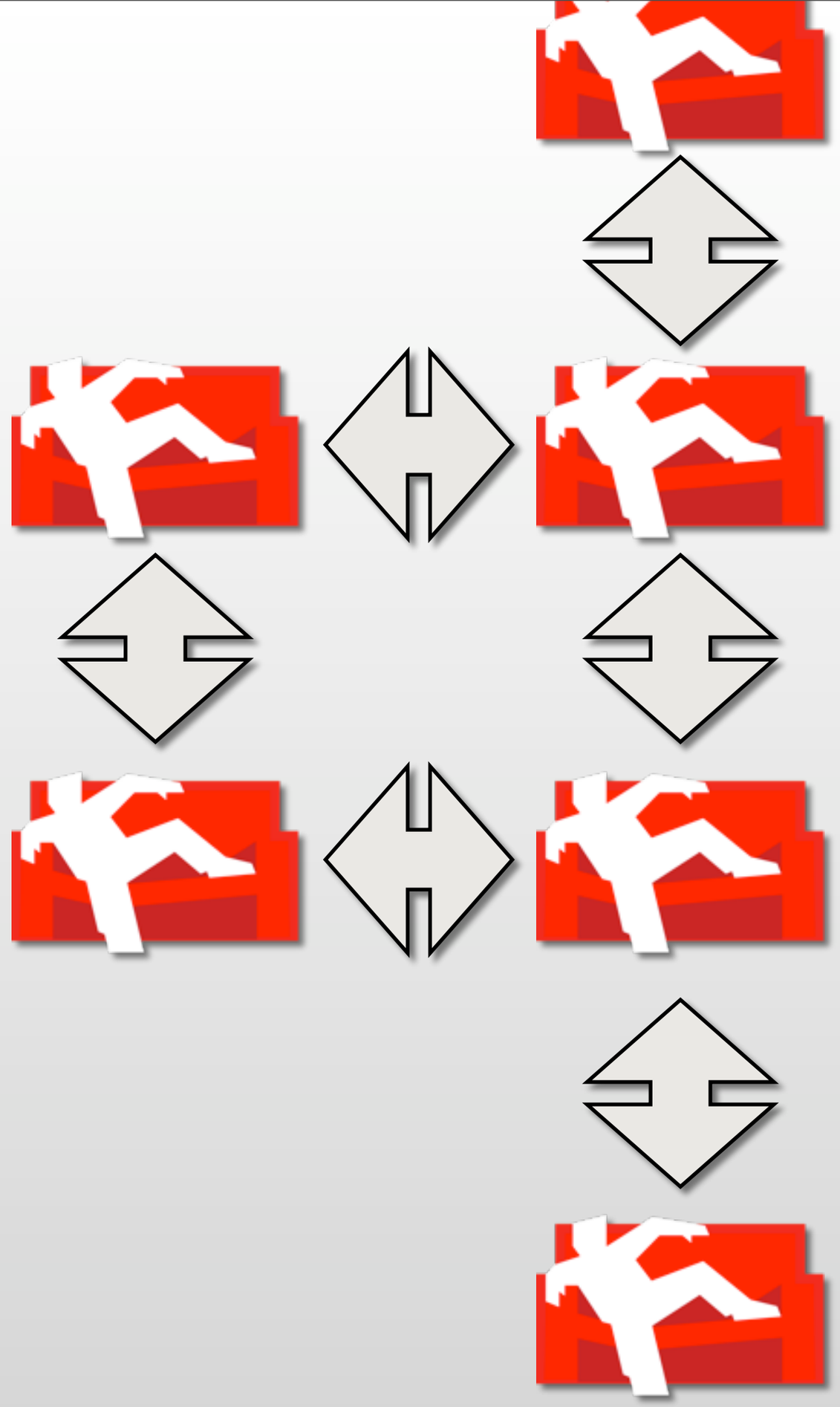


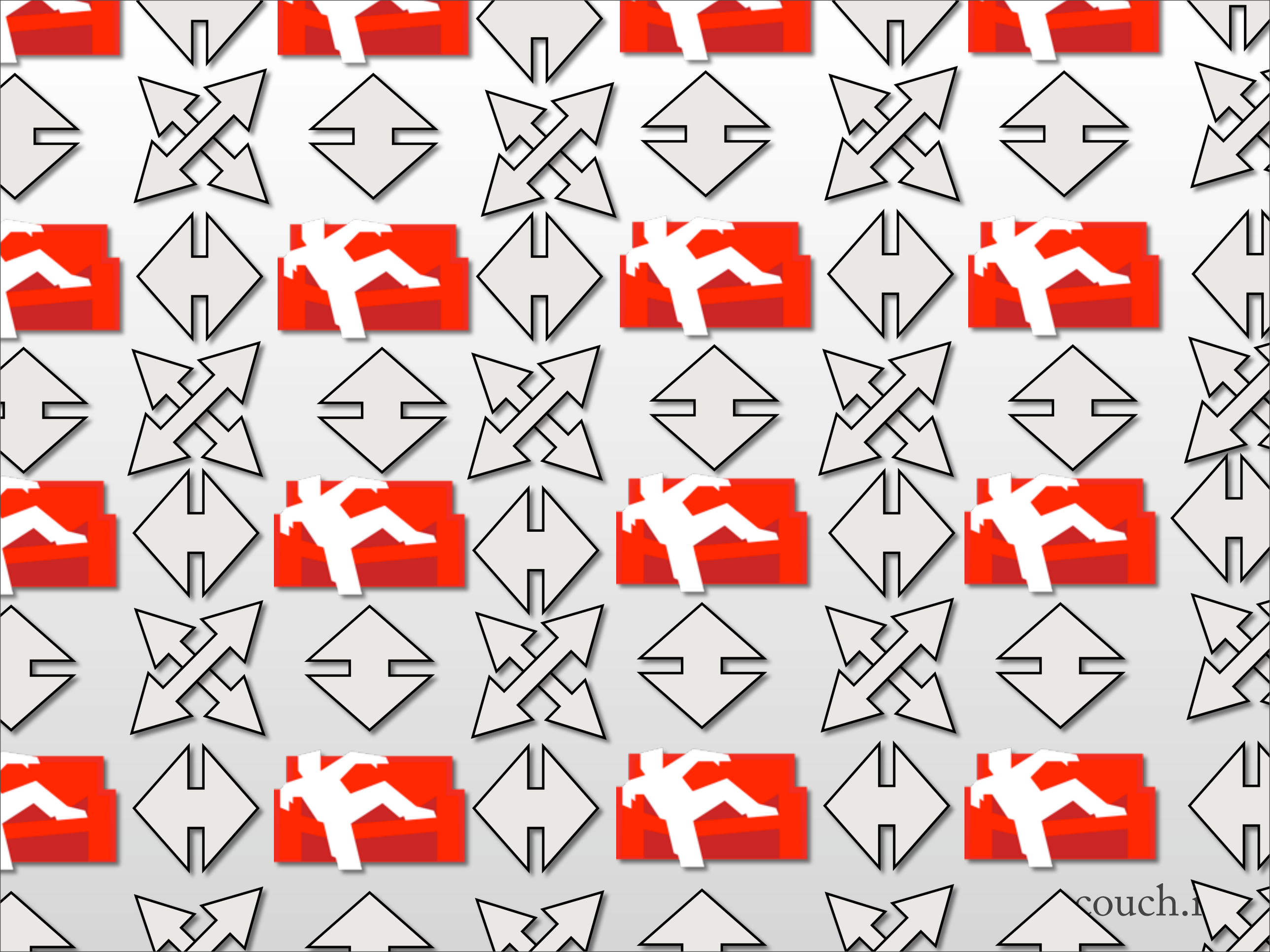






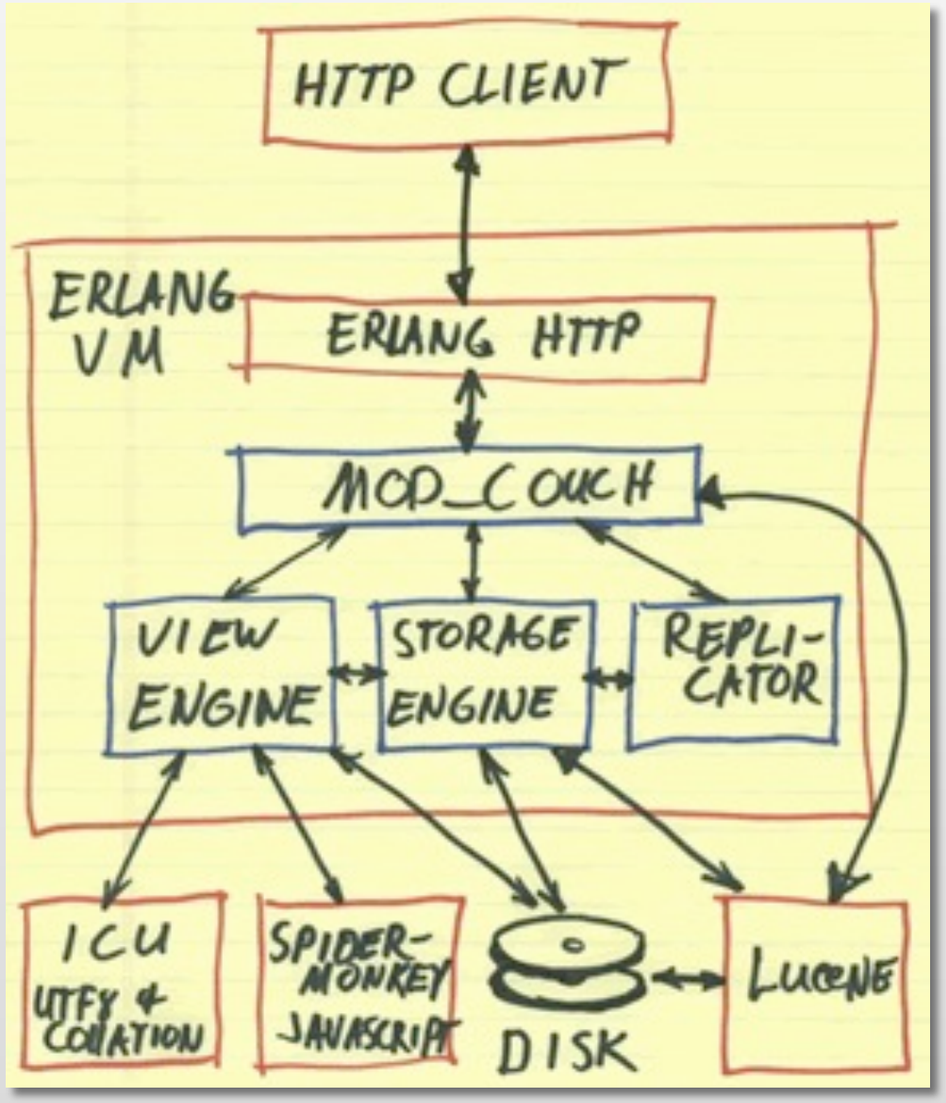





couch.

Features

- Schema Free (JSON)
- Document Oriented,
  Not Relational
- Highly Concurrent
- RESTful HTTP API
- JavaScript Powered
  Map/Reduce
- N-Master Replication
- **Robust Storage**

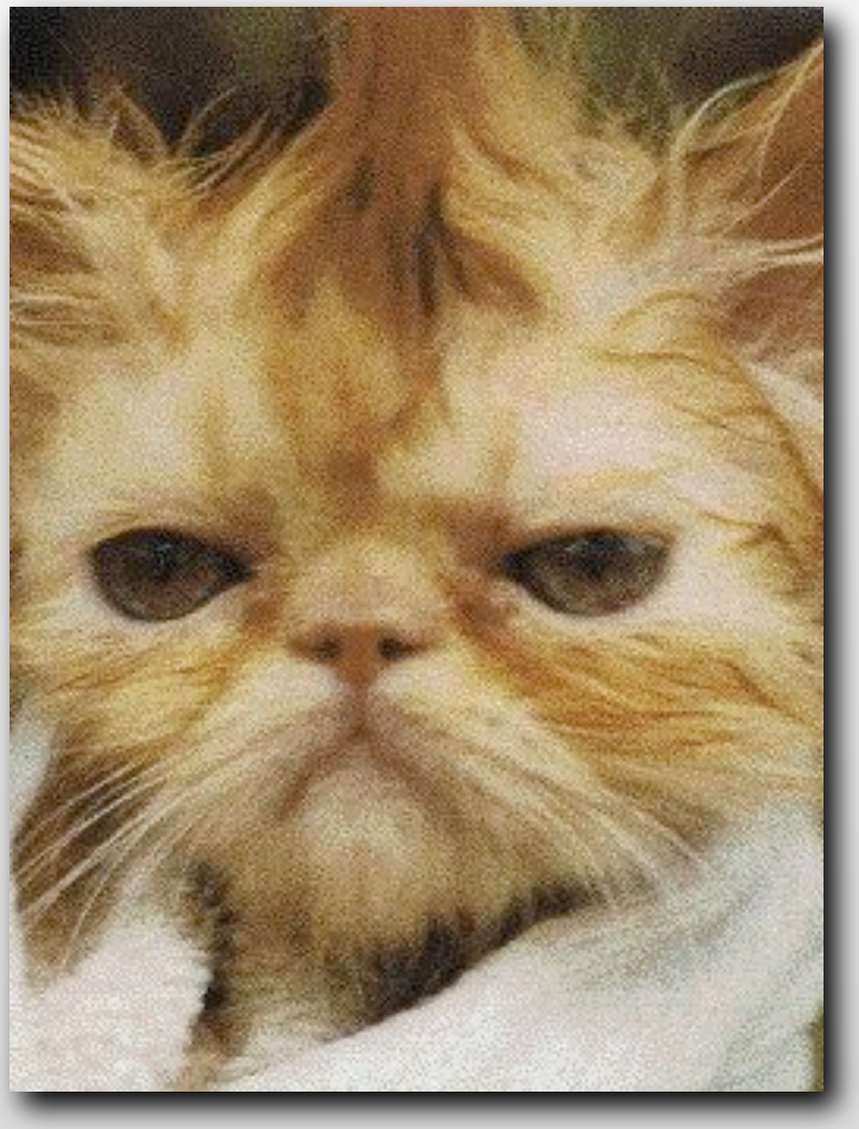


Robust Storage

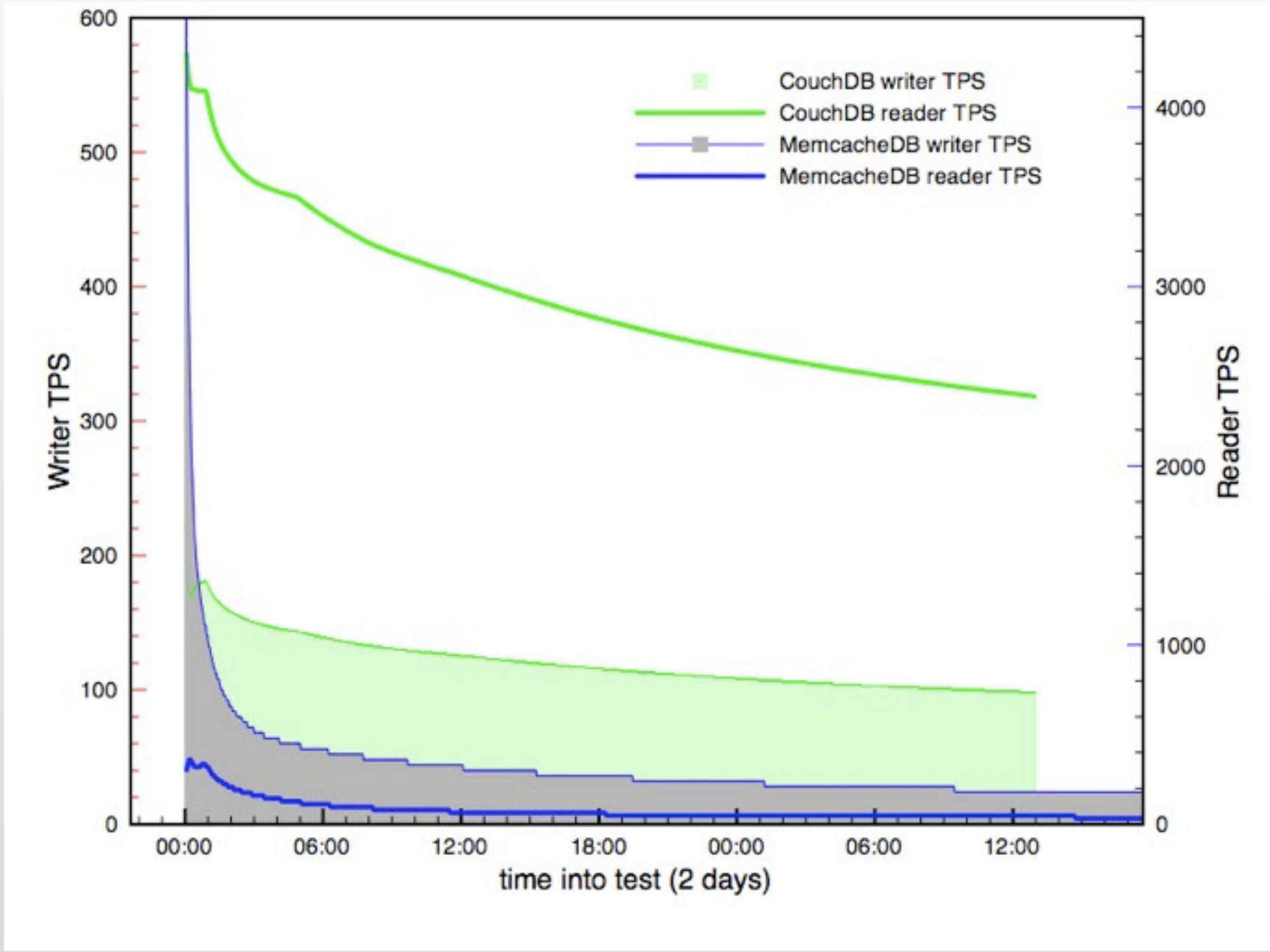
Append Only
File Structure

Designed to Crash

Instant-On
(no fixup phase)



Happy IO Patterns



Agenda

- Introduction
- **Installation**
- API by Example
- Design Documents
- CouchApps
- Advanced Views
- Replication
- Distributed Systems
- Deployment

Installation – Unix

- Dependencies
  - Erlang (duh), 5.6.4+ (R12B-4+)
  - libspidermonkey-dev, libjs-dev, libmozjs-dev, js-dev 1.6+
  - ICU 3.6+
- From Source
  - Autotools, autoconf, automake, make, libtool

Installation – Unix

- apt-get install automake autoconf libtool make help2man erlang libssl-dev libicu-dev libmozjs-dev libcurl4-openssl-dev libreadline5-dev gcc
- yum install gcc m4 autoconf213 curl-devel erlang libicu-devel make libtool readline-devel

Installation – Windows

- Sorry, no dice.
  - unless you're willing to hack it yourself.

Installation – Mac OS X

http://janl.github.com/couchdbx/

Ufeful Utilitief

- **easy_install -U couchapp**
- Firebug
- Safari 4
- tree
- gem install jsonpretty

Installation – Cloud

- Rent a couch: http://hosting.couch.io/
- Free Private Beta
  - Invite code: *booom-couch*

Apache CouchDB

Agenda

- Introduction
- Installation
- **API by Example**
- Client Libraries
- Design Documents
- CouchApps
- Advanced Views
- Replication
- Distributed Systems
- Deployment
- Internals
- Writing Extensions

API *by* example

- Server
- Databases
- Documents
- Replication
- Statistics
- Configuration
- Design Documents
  - Define Views
  - Query Views
  - Show & List
  - Validation

API *by* example: server

- **Server**
- Databases
- Documents
- Replication
- Statistics
- Configuration
- Design Documents
  - Define Views
  - Query Views
  - Show & List
  - Validation

API *by* example: Server

```
curl -vX GET http://127.0.0.1:5984/
> GET / HTTP/1.1
> Host: 127.0.0.1:5984
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: CouchDB/0.9.0 (Erlang OTP/R12B)
< Content-Length: 40
<
{"couchdb":"Welcome","version":"0.9.0"}
```

API *by* example: Server

```
curl -vX GET http://127.0.0.1:5984/
> GET / HTTP/1.1
> Host: 127.0.0.1:5984
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: CouchDB/0.9.0 (Erlang OTP/R12B)
< Content-Length: 40
<
{"couchdb":"Welcome","version":"0.9.0"}
```

API *by* example: Server

```
curl -vX GET http://127.0.0.1:5984/
> GET / HTTP/1.1
> Host: 127.0.0.1:5984
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: CouchDB/0.9.0 (Erlang OTP/R12B)
< Content-Length: 40
<
{"couchdb":"Welcome","version":"0.9.0"}
```

API *by* example: Server

```
curl -vX GET http://127.0.0.1:5984/
> GET / HTTP/1.1
> Host: 127.0.0.1:5984
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: CouchDB/0.9.0 (Erlang OTP/R12B)
< Content-Length: 40
<
{"couchdb":"Welcome","version":"0.9.0"}
```

API *by* example

- Server
- **Databases**
- Documents
- Replication
- Statistics
- Configuration

- Design Documents
  - Define Views
  - Query Views
  - Show & List
  - Validation

API *by* example: Databases

```
curl -vX GET http://127.0.0.1:5984/_all_dbs
>
< HTTP/1.1 200 OK
< Content-Type: text/plain;charset=utf-8
< Content-Length: 2
<
[]
```

API *by* example: Databases

```
curl -vX GET http://127.0.0.1:5984/_all_dbs
>
< HTTP/1.1 200 OK
< Content-Type: text/plain;charset=utf-8
< Content-Length: 2
<
[]
```

API *by* example: Databases

```
curl -vX GET http://127.0.0.1:5984/_all_dbs
>
< HTTP/1.1 200 OK
< Content-Type: text/plain;charset=utf-8
< Content-Length: 2
<
[]
```

API *by* example: Databases

```
curl -vX PUT http://127.0.0.1:5984/db
> PUT /db HTTP/1.1
> Host: 127.0.0.1:5984
>
< HTTP/1.1 201 Created
< Location: http://127.0.0.1:5984/db
< Content-Type: text/plain;charset=utf-8
< Content-Length: 12
<
{"ok":true}
```

API *by* example: Databases

```
curl -vX PUT http://127.0.0.1:5984/db
> PUT /db HTTP/1.1
> Host: 127.0.0.1:5984
>
< HTTP/1.1 201 Created
< Location: http://127.0.0.1:5984/db
< Content-Type: text/plain;charset=utf-8
< Content-Length: 12
<
{"ok":true}
```

API *by* example: Databases

```
curl -vX PUT http://127.0.0.1:5984/db
> PUT /db HTTP/1.1
> Host: 127.0.0.1:5984
>
< HTTP/1.1 201 Created
< Location: http://127.0.0.1:5984/db
< Content-Type: text/plain;charset=utf-8
< Content-Length: 12
<
{"ok":true}
```

API *by* example: Databases

```
curl -vX PUT http://127.0.0.1:5984/db
> PUT /db HTTP/1.1
> Host: 127.0.0.1:5984
>
< HTTP/1.1 201 Created
< Location: http://127.0.0.1:5984/db
< Content-Type: text/plain;charset=utf-8
< Content-Length: 12
<
{"ok":true}
```

API *by* example: Databases

```
curl -vX GET http://127.0.0.1:5984/_all_dbs
>
< HTTP/1.1 200 OK
< Content-Type: text/plain;charset=utf-8
< Content-Length: 6
<
["db"]
```

API *by* example: Databases

```
curl -vX DELETE http://127.0.0.1:5984/db
> DELETE /db HTTP/1.1
> Host: 127.0.0.1:5984
>
< HTTP/1.1 200 OK
< Content-Type: text/plain;charset=utf-8
< Content-Length: 12
<
{"ok":true}
```

API *by* example: Databases

```
curl -vX GET http://127.0.0.1:5984/_all_dbs
>
< HTTP/1.1 200 OK
< Content-Type: text/plain;charset=utf-8
< Content-Length: 2
<
[]
```

API *by* example

- Server
- Databases
- **Documents**
- Replication
- Statistics
- Configuration

- Design Documents
  - Define Views
  - Query Views
  - Show & List
  - Validation

API *by* example: Documents

```
curl -vX GET http://127.0.0.1:5984/db/_all_docs
> GET /db/_all_docs HTTP/1.1
> Host: 127.0.0.1:5984
>
< HTTP/1.1 200 OK
< Server: CouchDB/0.9.0 (Erlang OTP/R12B)
< Content-Length: 27
<
{"total_rows":0,"rows":[]}
```

API *by* example: Documents

```
curl -vX POST http://127.0.0.1:5984/db \
 -d '{"foo":1}'
> POST /db HTTP/1.1
> Host: 127.0.0.1:5984
> Content-Type: application/json
> Content-Length: 9
>
< HTTP/1.1 201 Created
< Location: http://127.0.0.1:5984/db/
f7963c91145f58dd5ec2fc744fdba199
< Content-Length: 73
<
{ "ok":true, \
  "id":"f7963c91145f58dd5ec2fc744fdba199", \
  "rev":"1-2183040919" }
```

API *by* example: Documents

```
curl -vX POST http://127.0.0.1:5984/db \
 -d '{"foo":1}'
> POST /db HTTP/1.1
> Host: 127.0.0.1:5984
> Content-Type: application/json
> Content-Length: 9
>
< HTTP/1.1 201 Created
< Location: http://127.0.0.1:5984/db/
f7963c91145f58dd5ec2fc744fdba199
< Content-Length: 73
<
{ "ok":true, \
  "id":"f7963c91145f58dd5ec2fc744fdba199", \
  "rev":"1-2183040919" }
```

API *by* example: Documents

```
curl -vX POST http://127.0.0.1:5984/db \
 -d '{"foo":1}'
> POST /db HTTP/1.1
> Host: 127.0.0.1:5984
> Content-Type: application/json
> Content-Length: 9
>
< HTTP/1.1 201 Created
< Location: http://127.0.0.1:5984/db/
f7963c91145f58dd5ec2fc744fdba199
< Content-Length: 73
<
{ "ok":true, \
  "id":"f7963c91145f58dd5ec2fc744fdba199", \
  "rev":"1-2183040919" }
```

API *by* example: Documents

```
curl -vX POST http://127.0.0.1:5984/db \
 -d '{"foo":1}'
> POST /db HTTP/1.1
> Host: 127.0.0.1:5984
> Content-Type: application/json
> Content-Length: 9
>
< HTTP/1.1 201 Created
< Location: http://127.0.0.1:5984/db/
f7963c91145f58dd5ec2fc744fdba199
< Content-Length: 73
<
{ "ok":true, \
  "id":"f7963c91145f58dd5ec2fc744fdba199", \
  "rev":"1-2183040919" }
```

API *by* example: Documents

```
curl -vX PUT http://127.0.0.1:5984/db/my-id \
 -d '{"bar":2}'
> PUT /db/my-id HTTP/1.1
> Host: 127.0.0.1:5984
> Content-Type: application/json
> Content-Length: 9
>
< HTTP/1.1 201 Created
< Location: http://127.0.0.1:5984/db/my-id
< Etag: "1-3205258393"
< Content-Length: 46
<
{ "ok":true, \
  "id":"my-id", \
  "rev":"1-3205258393" }
```

API *by* example: Documents

```
curl -X PUT http://127.0.0.1:5984/db/my-id2 \
 -d '{"bar":3}'
{"ok":true,"id":"my-id2","rev":"1-3358466818"}
```

API *by* example: Documents

```
curl -X GET http://127.0.0.1:5984/db/_all_docs
{"total_rows":3,"offset":0,"rows":[
  {"id":"f7963c91145f58dd5ec2fc744fdba199",
    "key":"f7963c91145f58dd5ec2fc744fdba199",
    "value":{"rev":"1-2183040919"}},

  {"id":"my-id",
    "key":"my-id",
     "value":{"rev":"1-3205258393"}},

  {"id":"my-id2",
    "key":"my-id2",
    "value":{"rev":"1-3358466818"}}
]}
```

API *by* example: Documents

```
curl -X PUT http://127.0.0.1:5984/db/my-id \
 -d '{"bar":4}'
{"error":"conflict","reason":"Document update conflict."}
```

API *by* example: Documents

```
curl -X PUT http://127.0.0.1:5984/db/my-id \
 -d '{"bar":4, "_rev":"1-3205258393"}'
{"ok":true,"id":"my-id","rev":"2-570602750"}
```

API *by* example: Documents

curl -X GET http://127.0.0.1:5984/_uuids
{"uuids":["07e6719d31c9e89352eddedc10bfe4cc"]}

curl -X GET http://127.0.0.1:5984/_uuids?count=2
{"uuids":["2ecb573d4141a359a5c1ba7a204ca0be", \
"6bf11020930d344693bfb6c004121629"]}

API *by* example

- Server
- Databases
- Documents
- **Replication**
- Statistics
- Configuration

- Design Documents
- Define Views
- Query Views
- Show & List
- Validation

API *by* example: Replication

curl -X PUT http://127.0.0.1:5984/db-replica
{"ok":true}

API *by* example: Replication

```
curl -vX POST http://127.0.0.1:5984/_replicate \
 -H 'Content-Type: application/json' \
 -d '{"source":"db", \
       "target":"db-replica"}'
> POST /_replicate HTTP/1.1
> Host: 127.0.0.1:5984
> Content-Type: application/json
> Content-Length: 37
>
< HTTP/1.1 200 OK
< Server: CouchDB/0.9.0 (Erlang OTP/R12B)
< Content-Length: 310
```

API *by* example: Replication

```
curl -vX POST http://127.0.0.1:5984/_replicate \
 -H 'Content-Type: application/json' \
 -d '{"source":"db", \
       "target":"db-replica"}'
> POST /_replicate HTTP/1.1
> Host: 127.0.0.1:5984
> Content-Type: application/json
> Content-Length: 37
>
< HTTP/1.1 200 OK
< Server: CouchDB/0.9.0 (Erlang OTP/R12B)
< Content-Length: 310
```

API *by* example: Replication


```
{"ok":true,
 "session_id":"566b128214cf86c5b09d7bb89aa15c86",
 "source_last_seq":4,
 "history":[
  {"start_time":"Sun, 12 Apr 2009 20:17:09 GMT",
   "end_time":"Sun, 12 Apr 2009 20:17:10 GMT",
   "start_last_seq":0,
   "end_last_seq":4,
   "missing_checked":4,
   "missing_found":4,
   "docs_read":4,
   "docs_written":4,
   "doc_write_failures":0
  }]
}
```

API *by* example: Replication

```
curl -vX POST http://127.0.0.1:5984/_replicate \
 -H 'Content-Type: application/json' \
 -d '{"source":"db", \
       "target":"db-replica"}'
> POST /_replicate HTTP/1.1
> Host: 127.0.0.1:5984
> Content-Type: application/json
> Content-Length: 37
>
< HTTP/1.1 200 OK
< Server: CouchDB/0.9.0 (Erlang OTP/R12B)
< Content-Length: 310
```

API *by* example: Replication

Local

```
-d '{"source":"db", \
      "target":"db-replica"}'
```

API *by* example: Replication

Pull

```
-d '{"source":"http://server/db", \
      "target":"db-replica"}'
```

API *by* example: Replication

Push

```
-d '{"source":"db-replica", \
      "target":"http://server/db"}'
```

API *by* example: Replication

Remote

```
-d '{"source":"http://server-one/db", \
      "target":"http://server-two/db"}'
```

API *by* example

- Server
- Databases
- Documents
- Replication
- **Statistics**
- Configuration

- Design Documents
- Define Views
- Query Views
- Show & List
- Validation

API *by* example: Statistics

curl -X GET http://127.0.0.1:5984/_stats/

API *by* example: Statistics


```
{"httpd_status_codes":
  {"200":
    {"current":24,
      "count":30333,
      "mean":0.0007912174859064414,
      "min":0, "max":2,
      "stddev":0.030372043753768774,
      "description":"number of HTTP 200 OK responses"},

  "201":
   {"current":8,
     "count":30259,
     "mean":0.0002643841501701965,
     "min":0, "max":1,
     "stddev":0.016257744345121602,
     "description":"number of HTTP 201 Created responses"},
```

API *by* example: Statistics


```
{"httpd_status_codes":
  {"200":
    {"current":24,
      "count":30333,
      "mean":0.0007912174859064414,
      "min":0, "max":2,
      "stddev":0.030372043753768774,
      "description":"number of HTTP 200 OK responses"},

  "201":
   {"current":8,
     "count":30259,
     "mean":0.0002643841501701965,
     "min":0, "max":1,
     "stddev":0.016257744345121602,
     "description":"number of HTTP 201 Created responses"},
```

API *by* example: Statistics


```
{"httpd_status_codes":
  {"200":
    {"current":24,
     "count":30333,
     "mean":0.0007912174859064414,
     "min":0, "max":2,
     "stddev":0.030372043753768774,
     "description":"number of HTTP 200 OK responses"},

  "201":
   {"current":8,
    "count":30259,
    "mean":0.0002643841501701965,
    "min":0, "max":1,
    "stddev":0.016257744345121602,
    "description":"number of HTTP 201 Created responses"}
```

API *by* example: Statistics


```
{"httpd_status_codes":
  {"200":
    {"current":24,
      "count":30333,
      "mean":0.0007912174859064414,
      "min":0, "max":2,
      "stddev":0.030372043753768774,
      "description":"number of HTTP 200 OK responses"},

  "201":
   {"current":8,
     "count":30259,
     "mean":0.00026438415017019650,
     "min":0, "max":1,
     "stddev":0.016257744345121602,
     "description":"number of HTTP 201 Created responses"}
```

API *by* example: Statistics


```
"httpd_request_methods":
  {"DELETE":
    {"current":1,
      "count":23690,
      "mean":0.00004221190375685942,
      "min":0,"max":1,
      "stddev":0.006496931730597203 5,
      "description":"number of HTTP DELETE requests"},

  "GET":
    {"current":27,
      "count":30333,
      "mean":0.0008901196716447457,
      "min":0,"max":3,
      "stddev":0.03395683319427332 5,
      "description":"number of HTTP GET requests"},
```

API *by* example: Statistics


```
"httpd_request_methods":
  {"DELETE":
    {"current":1,
     "count":23690,
     "mean":0.00004221190375685942,
     "min":0,"max":1,
     "stddev":0.006496931730597203 5,
     "description":"number of HTTP DELETE requests"},

  "GET":
   {"current":27,
    "count":30333,
    "mean":0.0008901196716447457,
    "min":0,"max":3,
    "stddev":0.03395683319427332 5,
    "description":"number of HTTP GET requests"},
```

API *by* example: Statistics


```
"httpd":
  {"requests":
    {"current":43,
     "count":30333,
     "mean":0.0014175979955823685,
     "min":0,"max":3,
     "stddev":0.04177633736897358,
     "description":"number of HTTP requests"},
```

API *by* example: Statistics


```
"couchdb":
  {"database_reads":
    {"current":26,
      "count":30332,
      "mean":0.00085718053540815 54,
      "min":0,"max":9,
      "stddev":0.065461193989801 8,
      "description":"number of times a document was read
   from a database"},
"database_writes":
  {"current":8,
    "count":30259,
    "mean":0.00026438415017019 81,
    "min":0,"max":3,
    "stddev":0.021508192946386385,
    "description":"number of times a database was
   changed"},
```

API *by* example: Statistics

/_stats/<module>/<counter>

/_stats/httpd_status_codes/200
/_stats/httpd/requests
/_stats/couchdb/database_writes

API *by* example

- Server
- Databases
- Documents
- Replication
- Statistics
- **Configuration**
- Design Documents
- Define Views
- Query Views
- Show & List
- Validation

API *by* example: Configuration

curl -X GET http://127.0.0.1:5984/_config

API *by* example: Configuration


```
  "couchdb": {
    "os_process_timeout": "5000",
    "util_driver_dir": "\/usr\/local\/lib\/couchdb\/erlang\/lib
\/  \
       couch-0.9.0\/priv\/lib",
    "max_document_size": "4294967296",
    "max_dbs_open": "100",
    "max_attachment_chunk_size": "4294967296",
    "database_dir": "\/usr\/local\/var\/lib\/couchdb",
    "view_index_dir": "\/usr\/local\/var\/lib\/couchdb"
  },
```

API *by* example: Configuration

```
"httpd": {
  "bind_address": "127.0.0.1",
  "WWW-Authenticate": "Basic realm=\"administrator\"",
  "port": "5984",
  "authentication_handler": \
     "{couch_httpd, default_authentication_handler}"
}
```

API *by* example: Configuration

```
"log": {
  "level": "debug",
  "file": "\/usr\/local\/var\/log\/couchdb\/couch.log"
},
```

API *by* example: Configuration
at runtime

/_config/couchdb/max_dbs_open
/_config/httpd/port
/_config/log/level

- Unlike Statistics:
  - Accept GET & POST

```
> GET /_config/couchdb/max_dbs_open
"100"
> PUT /_config/couchdb/max_dbs_open  -d '"50"'
"100"
> GET /_config/couchdb/max_dbs_open
"50"
```

Excercise

- Create a database **albums**
- Create five document with your favourite music albums
- {"title":"The Color and the Shape",
  "artist":"Foo Fighters",
  "year":1997,
  "author":"jan"
  …
  }

Add as much info as you know

Excercise

- Create second database **album-backup**
- Use Futon *Replication* to replicate **albums** to **albums-backup**
- Replicate with your left & right neighbours
- Add album info to your peer's albums you happen to know
- Discover conflicts

API *by* example

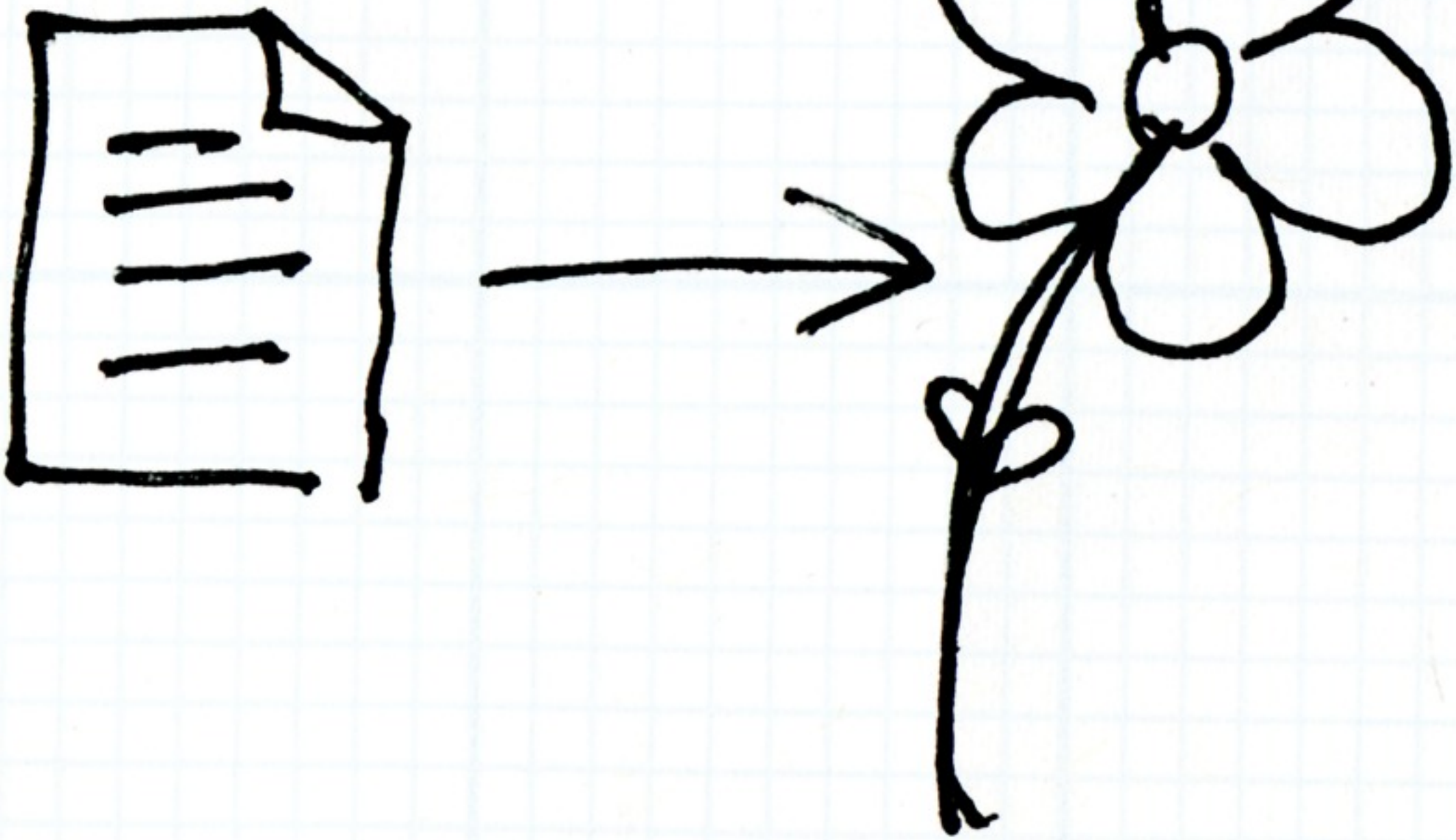
- Server
- Databases
- Documents
- Replication
- Statistics
- Configuration

- **Design Documents**
  - Define **Views**
    - View Query Path
  - **Show & List**
  - **Validation**

Applications

are

Documents.



_design/

Documents

- map/reduce
- validations
- render html

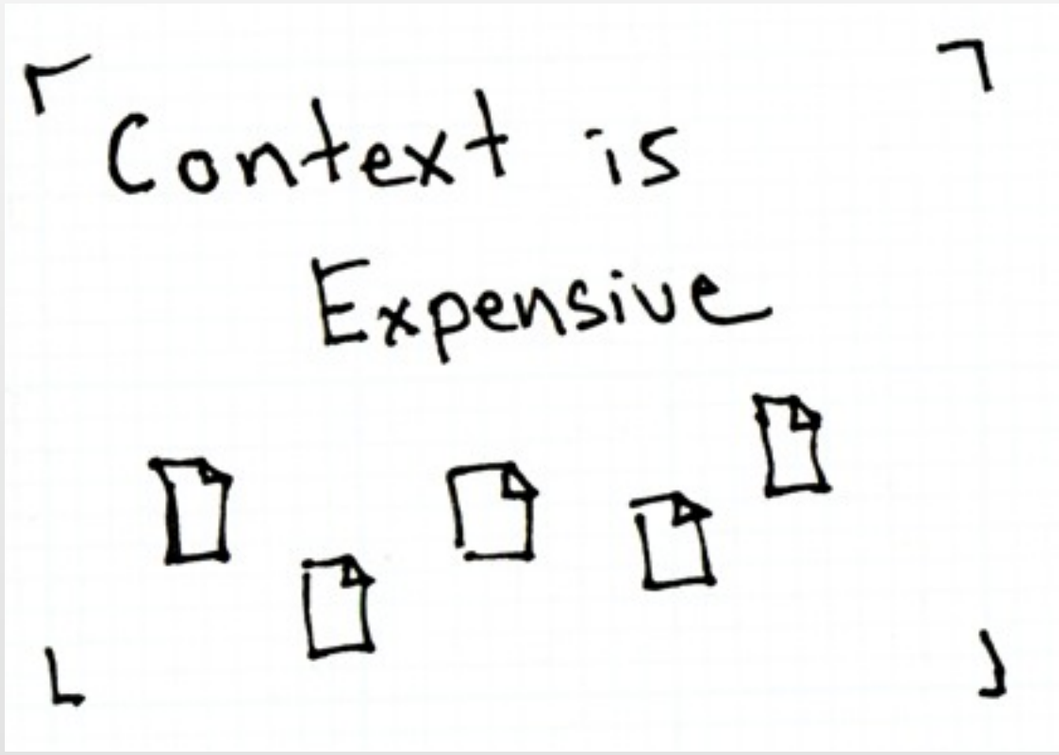
Context is Expensive

Render requests have no side effects for cacheablity.

Functions limited to individual database events.

Use updates to trigger asynchronous processing.

Validate individual updates.



Design Documents

- Map Views
- Update Validations
- Show & List
- Other User Code

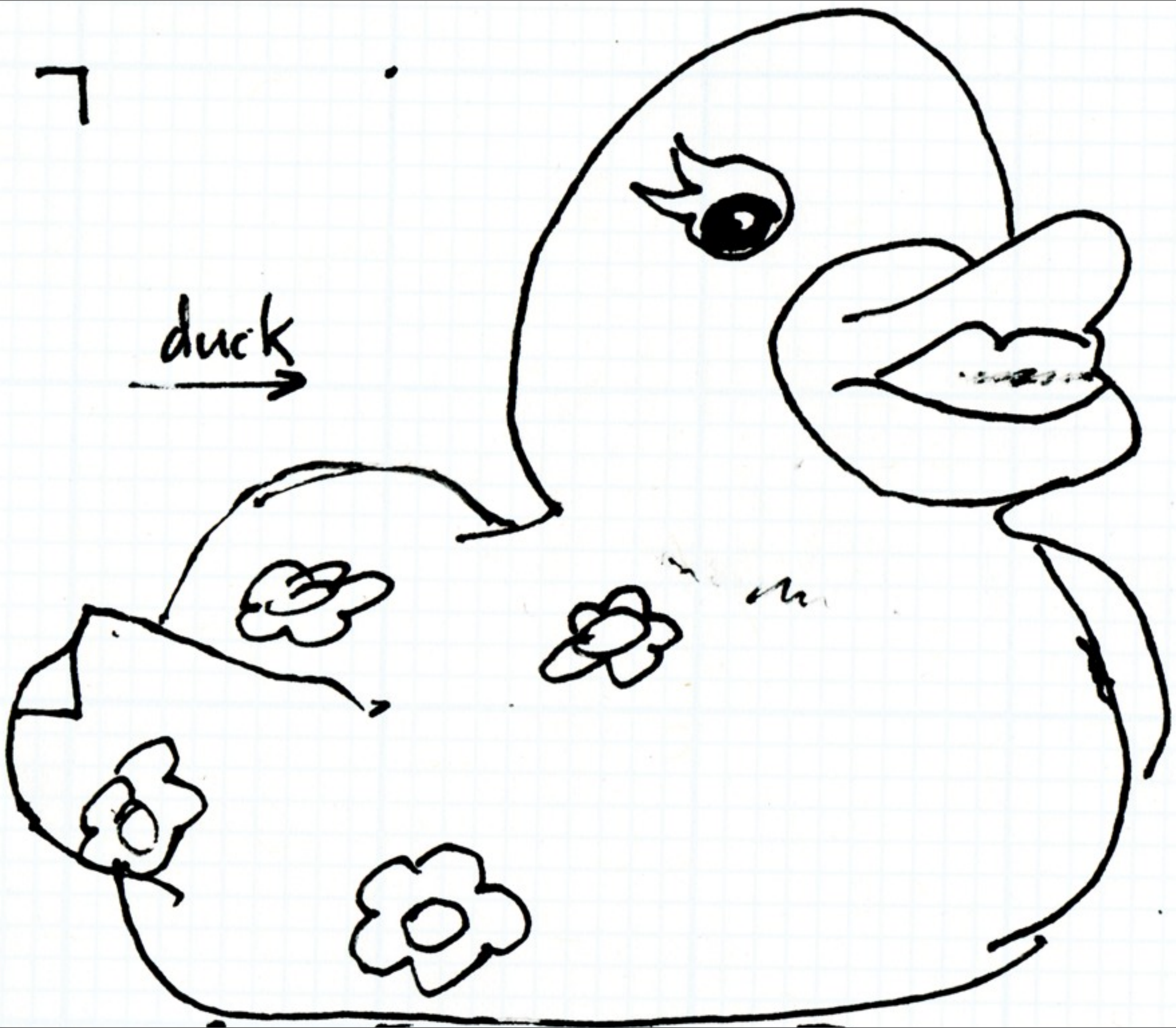
View Definitions

```
{
 "views": {
    "foobar" : {
    "map": "function(doc){emit(doc.date, 1)}",
    "reduce":
        "function(keys, values, rereduce) {
           return sum(values)
        }"
}
```

Duck

TYPING

duck



View Query

/my-db/_design/ddoc-name/_view/foobar
?startkey=[1990,2]&endkey=[1995,2]

Update Validations

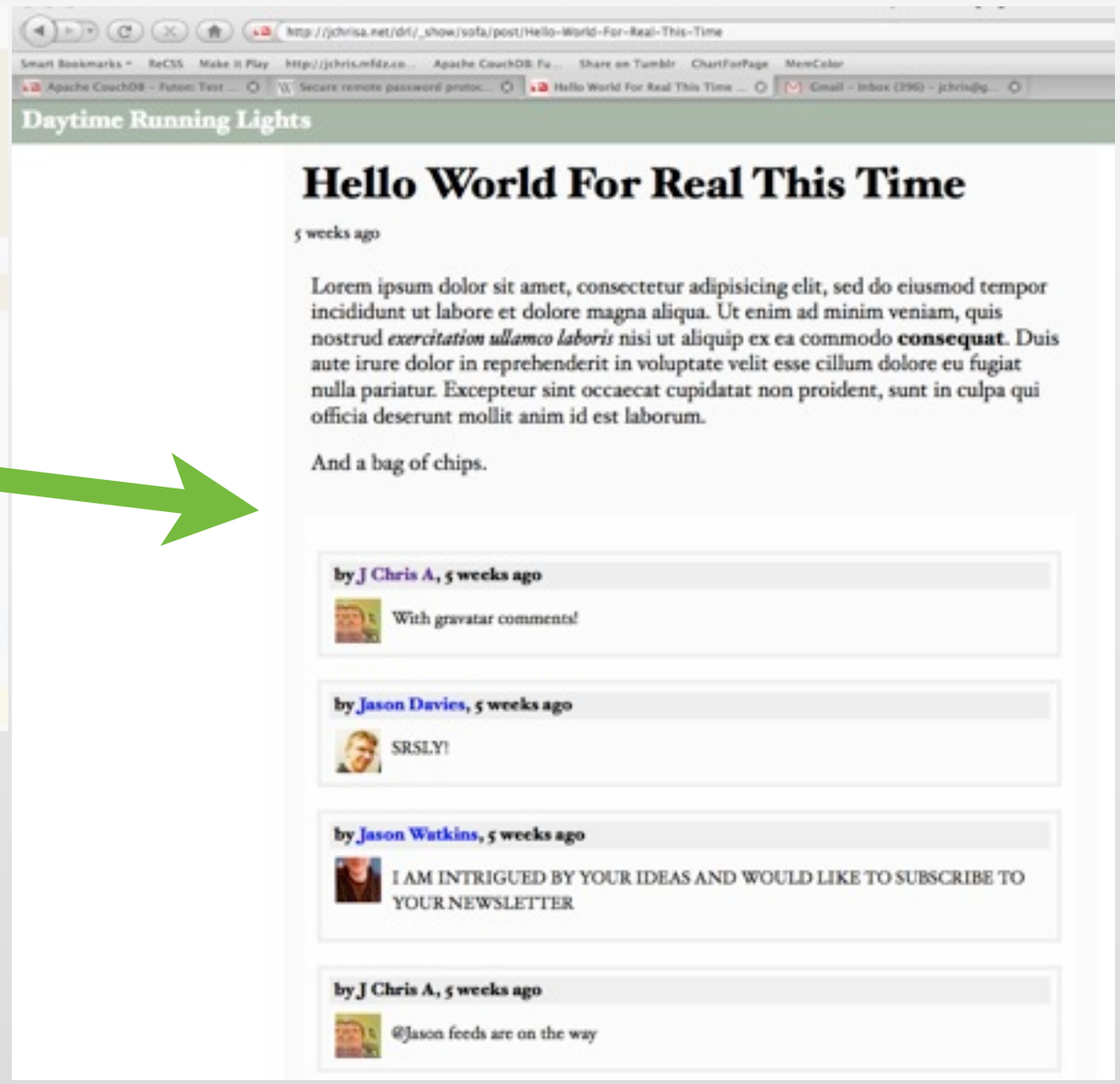
```
{
  "validate_doc_update" :
      "function (newDoc, oldDoc, userCtx) {
          if(!doc.title) {
              throw({bad_request : "must have title"});
          }
      }"
  }
}
```

Render JSON Docs as HTML

shows/post.js

```
function(doc, req) {
  // !json templates.post
  // !json blog
  // !code helpers.template
  // !code helpers.couchapp
  // log(req.headers.Accept);

  // we only show html
  return template(templates.post, {
    title : doc.title,
    blogName : blog.title,
    post : doc.html,
    date : doc.created_at,
    author : doc.author,
    assets : assetPath(),
    editPostPath : showPath('edit', doc._id),
    index : listPath('index','recent-posts',{descending:true, limit:8})
  });
}
```

/drl/_show/sofa/post/Hello-World-For-Real-This-Time

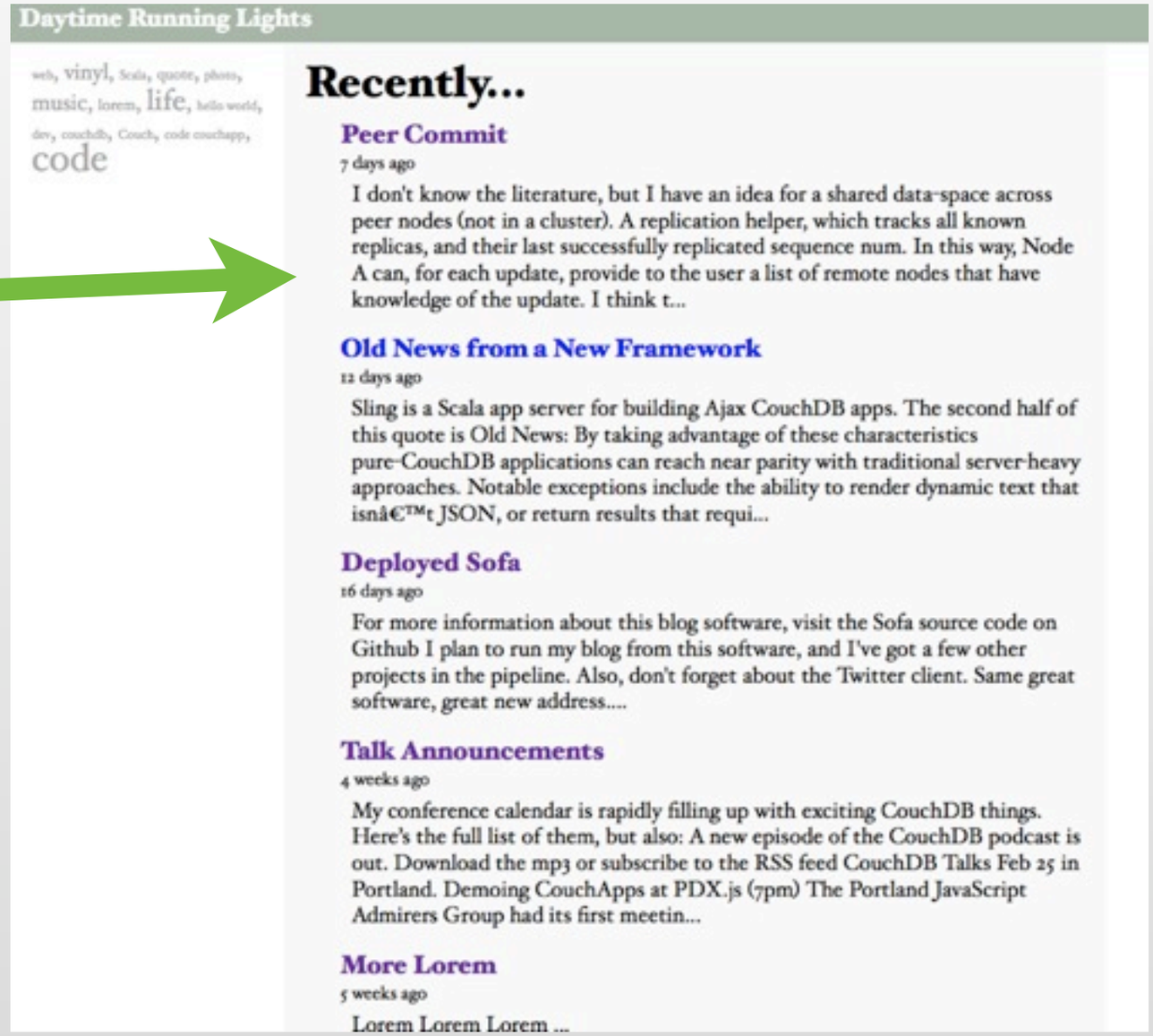


Render View Rows as HTML

lists/index.js

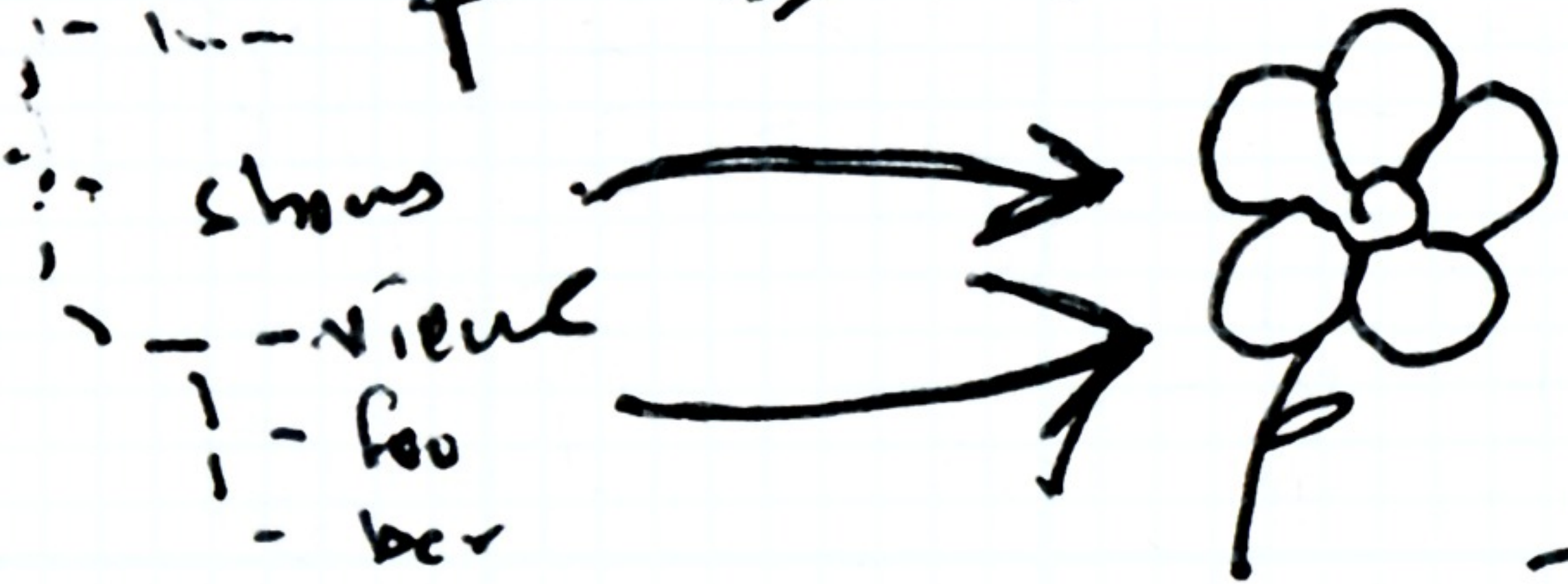
```
function(head, row, req) {
  // !json templates.index
  // !json blog
  // !code helpers.couchapp
  // !code helpers.template
  // log(req.headers.Accept);
  var indexPath = listPath('index','recent-posts',{descending:true, limit:8});
  var feedPath = listPath('index','recent-posts',{descending:true, limit:8, format:"atom"});
  return respondWith(req, {
    html : function() {
      if (head) {
        return template(templates.index.head, {
          title : blog.title,
          newPostPath : showPath("edit"),
          index : indexPath,
          assets : assetPath()
        });
      } else if (row) {
        var post = row.value;
        return template(templates.index.row, {
          title : post.title,
          summary : post.summary,
          date : post.created_at,
          link : showPath('post', row.id),
          assets : assetPath()
        });
      } else {
        return template(templates.index.tail, {
          assets : assetPath()
        });
      }
    },
    atom : function() {
      // with first row in head you can do updated.
      if (head) {
        var f = <feed xmlns="http://www.w3.org/2005/Atom"/>;
        f.title = blog.title;
        f.id = makeAbsolute(req, indexPath);
        f.link.@href = makeAbsolute(req, feedPath);
        f.link.@rel = "self";
        f.generator = 'Sofa on CouchDB';
        f.updated = new Date().rfc3339();
        return {body:f.toXMLString().replace(/\<\/feed\>/,'')};
      } else if (row) {
        var entry = <entry/>;
        entry.id = makeAbsolute(req, '/'+encodeURIComponent(req.info.db_name)+'/'+encodeURICom
        entry.title = row.value.title;
        entry.content = row.value.summary;
        entry.content.@type = 'html';
        entry.updated = new Date(row.value.created_at).rfc3339();
        entry.author = <author><name>{row.value.author}</name></author>;
        entry.link.@href = makeAbsolute(req, showPath('post', row.id));
        entry.link.@rel = "alternate";
        return {body:entry};
      } else {
        return {body : "</feed>"};
      }
    }
  })
};
```

/drl/_list/sofa/index/recent-posts?descending=true&limit=8



couchapp

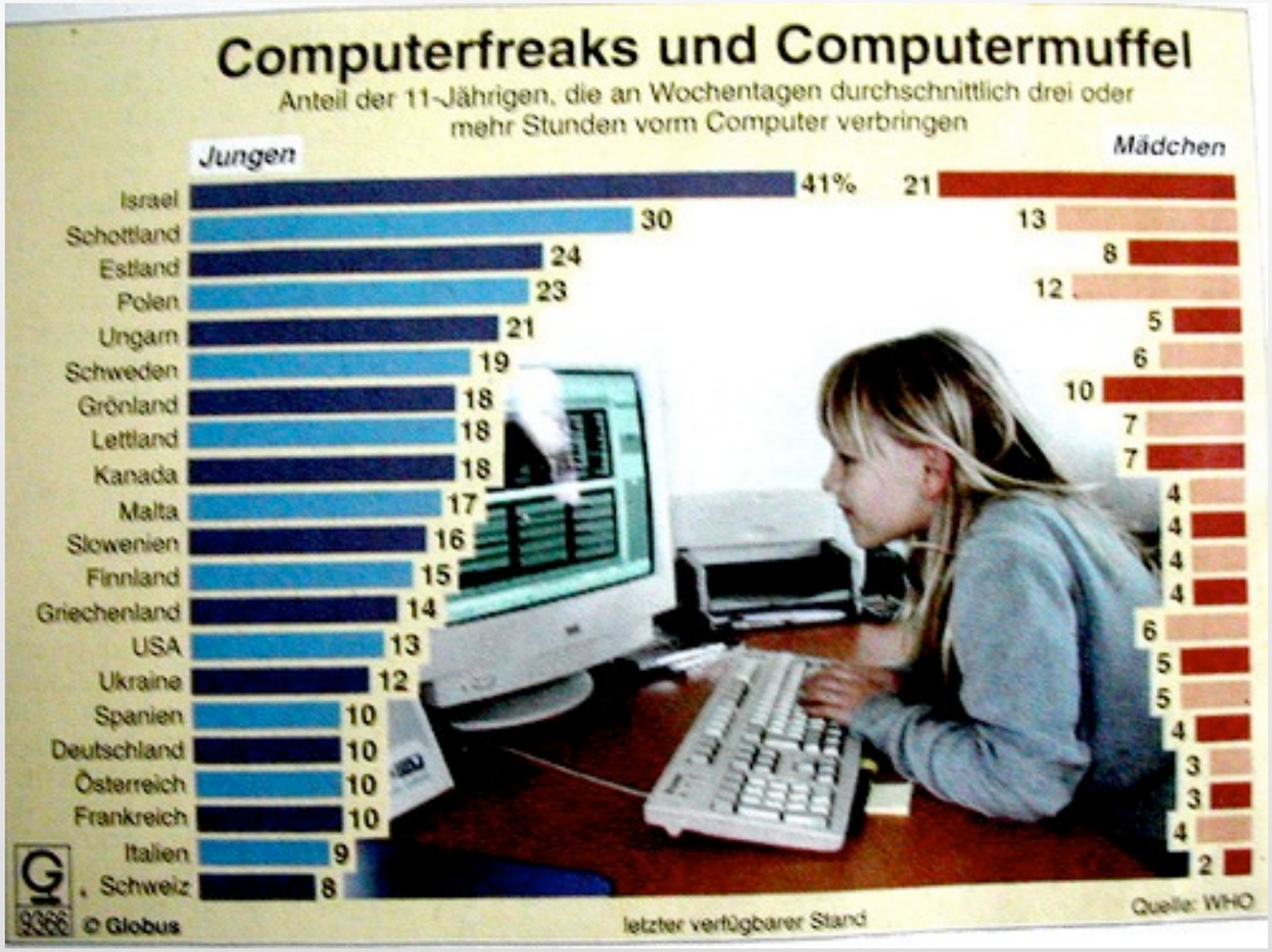
push



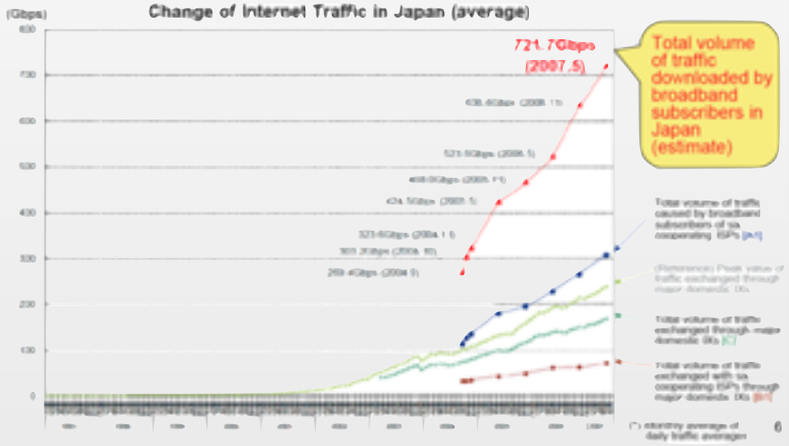
VIEW SOURCE

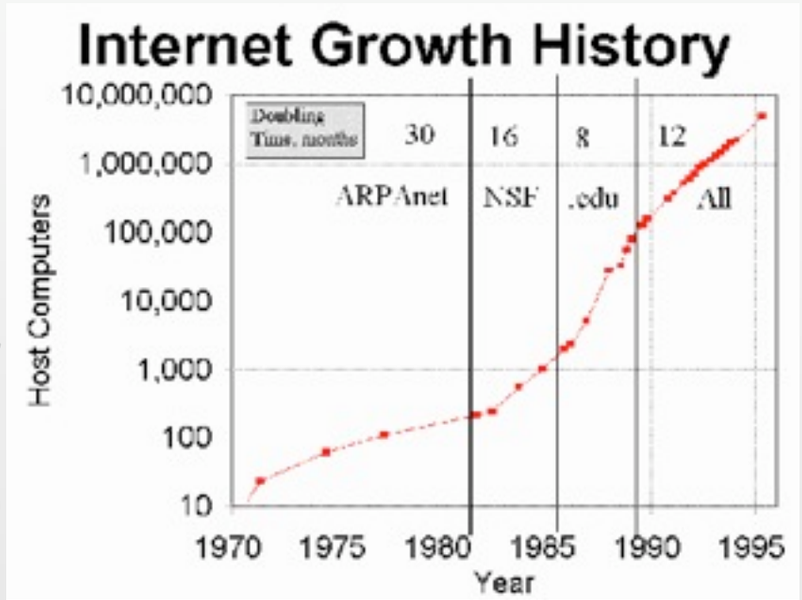
OPEN SOURCE

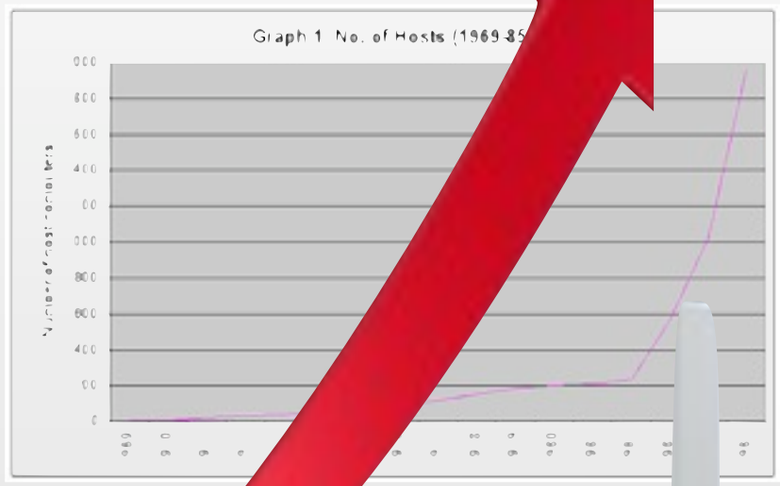
Gives Control to Users

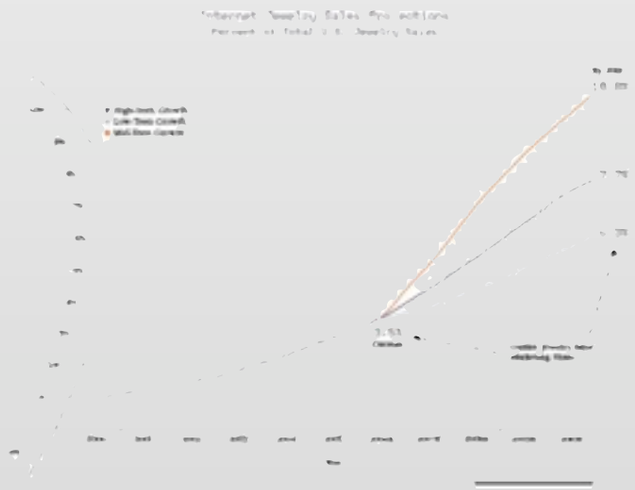


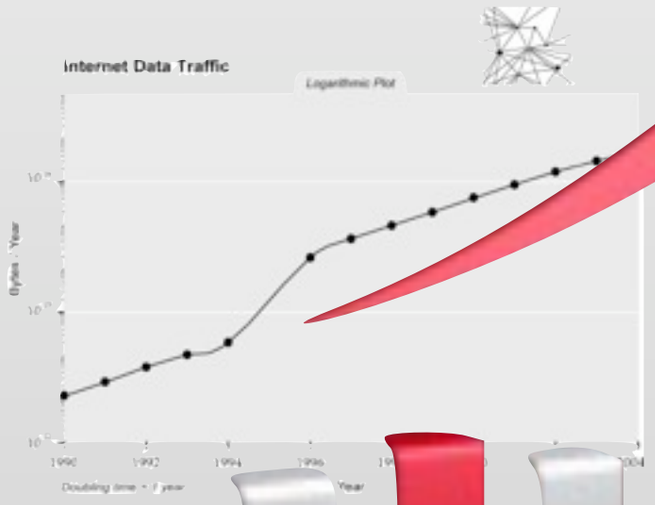
Bandwidth Explosion



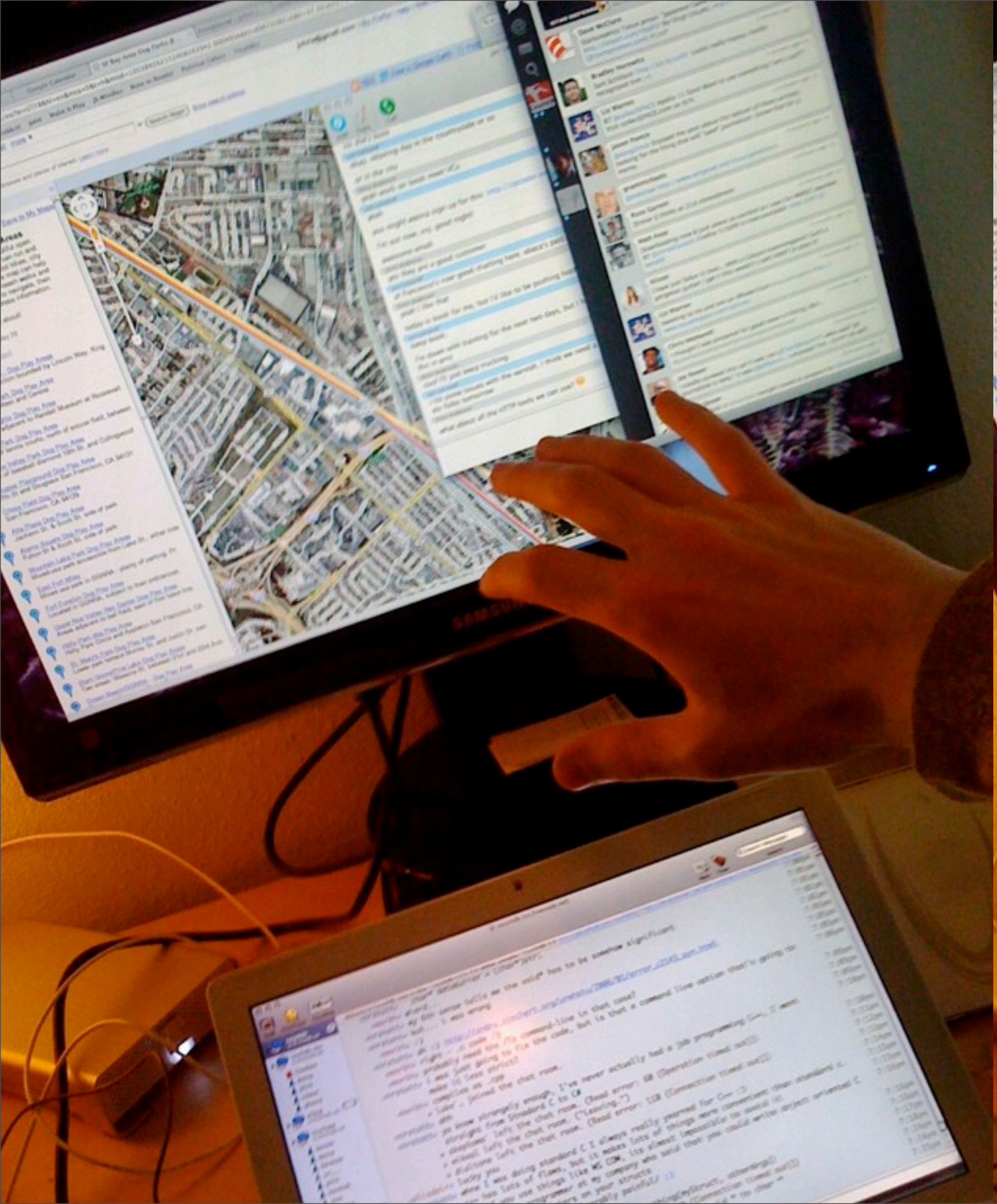






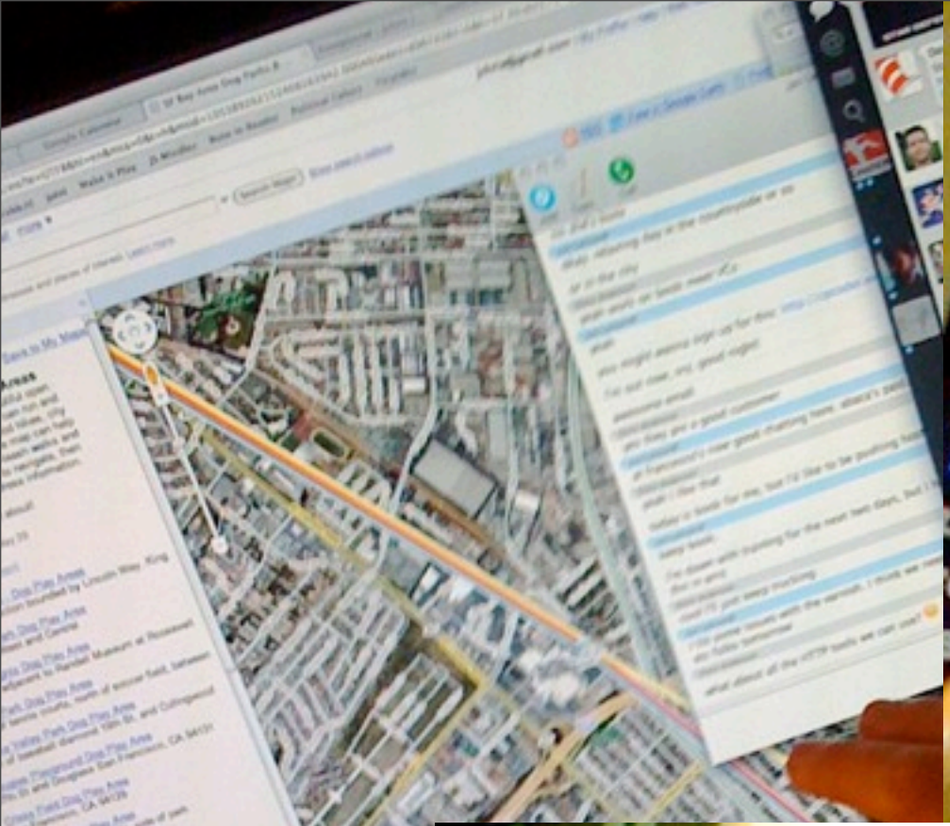


couch.io




$ g c halorgium/
warning: refna
Switched to
brindle 19
$ g pull
remote:
emot
er' is ambiguous.
aster"
(halorgium/master) jc
8, done.
cs: 100% (17/17), done.
13), reused 0 (delta 0)
(17/17), done.
halorgium/couchdb
master
1f3ede -> halorgium/master
ews.js
ouch_db_update_notifier.erl | 1 +
ouch_os_process.erl | 4 +++
mochijson2.erl | 4 ++
ed, 34 insertions(+), 0 deletions(-)
09 ~/code/couchdb-git(halorgium/master) jchris


$ g c halorgium/
warning: refn
Switched to
brindle 19
$ g pull
remote:
emot
er' is ambiguous.
aster"
halorgium/master) j
8, done.
cs: 100% (17/17), done.
13), reused 0 (delta 0)
(17/17), done.
halorgium/couchdb
master
1f3ede
-> halorgium/ma
.js
db_upda notifier.erl
os_proce
son2.erl
insertions(+)
de/couchdb-gi




SUNTORY
SUNTORY


AC OK

“no bars”

http://www.flickr.com/photos/hercwad/2290378571/

Latency Sucks

$$rt^b = rt^a + \sum_{i \in \text{Links}} (l_i^b - l_i^a) \cdot \vec{\nabla} \bar{rt}(l_i)$$

“Ground Computing”

@jhuggins

http://www.flickr.com/photos/mcpig/872293700/

Scaling Down

4 MB RAM

couch.io

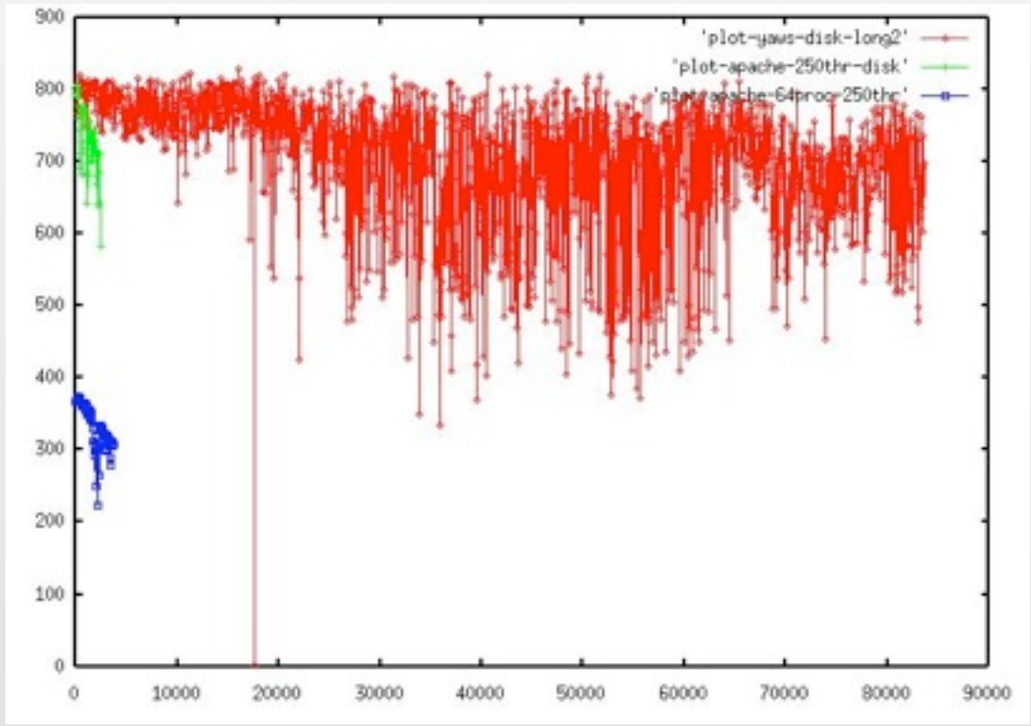
Erlang

Parallel

Fault tolerant

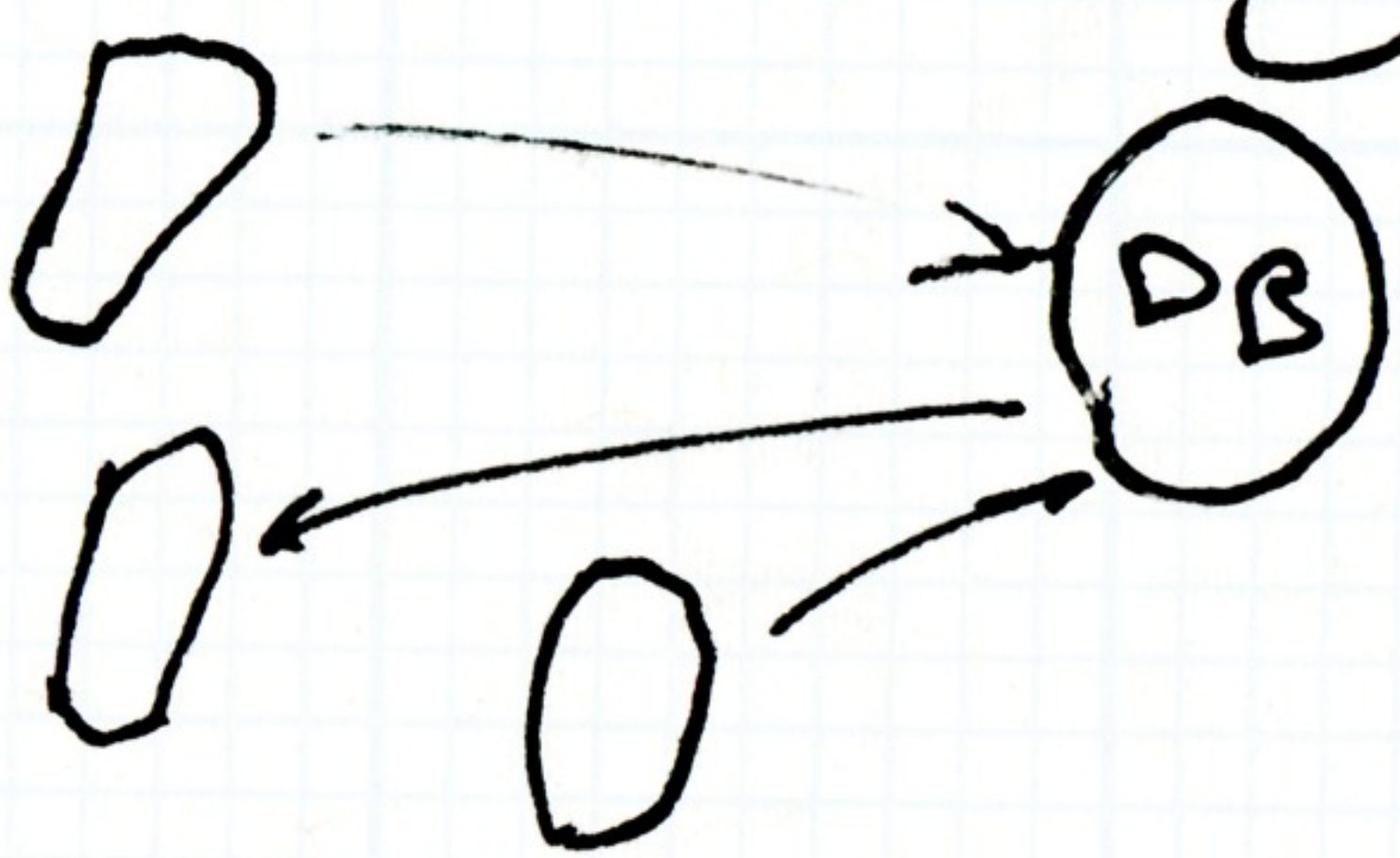
Addictive

Ninja Syntax



http://www.sics.se/~joe/apachevsyaws.html

Concurrency



"smart phones"

Browser Couch

JavaScript port

Uses HTML5 storage

Replicates with CouchDB

http://hg.toolness.com/browser-couch/

couch.io

BrowserCouch Test Suite

This file contains a simple test runner and suite for BrowserCouch.

The suite can be run here.

The Runner

The `Testing` namespace is a simple test framework that supports the skipping of tests (for when the host system doesn't support required functionality for a test's execution), as well as asynchronous tests.

It doesn't currently have the ability to detect when an asynchronous test has failed, however.

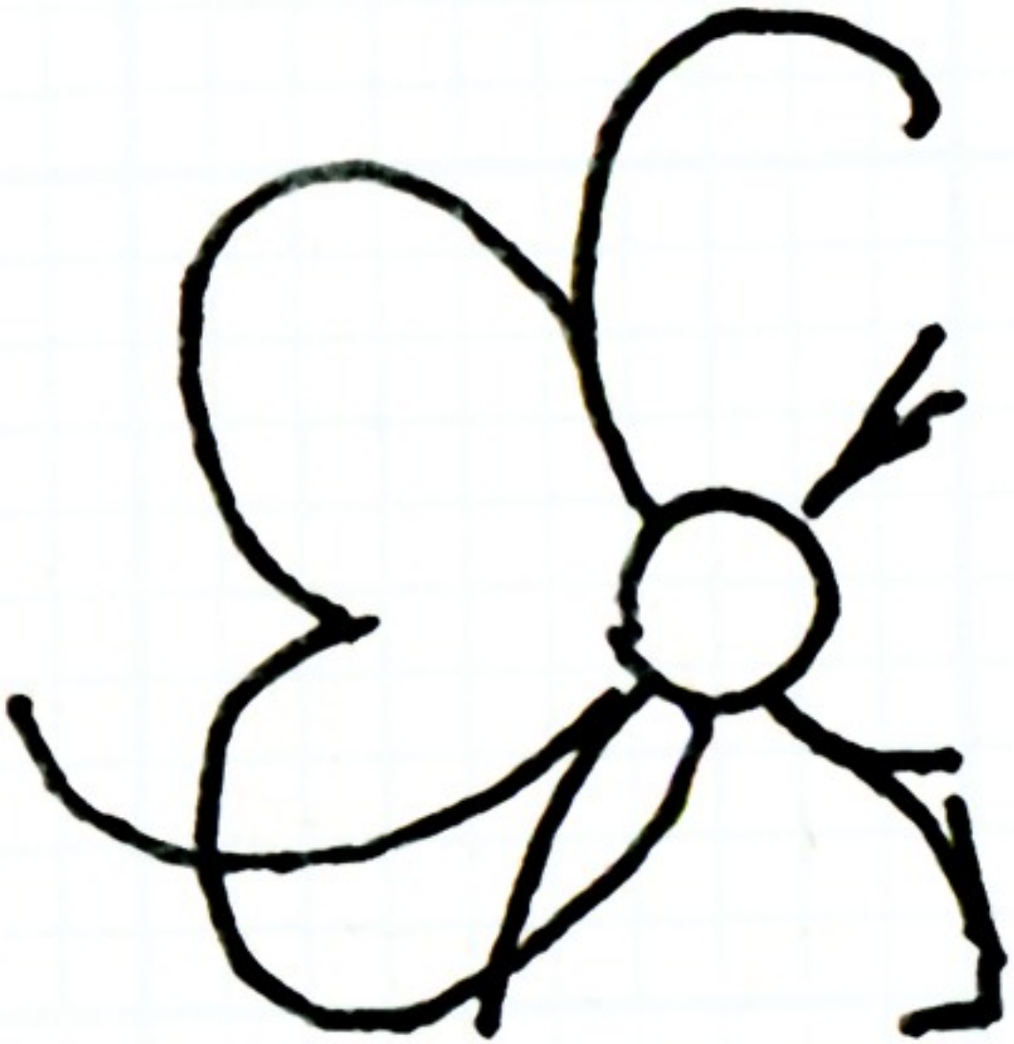

```
var Testing = {
  run: function(listener, container, setTimeout) {
    if (!setTimeout)
      setTimeout = window.setTimeout;

    var tests = [];

    for (name in container)
      if (name.indexOf("test") == "0") {
        var test = {
          name: name,
          func: container[name],
          isAsync: name.indexOf("_async") != -1,
          id: tests.length,
          assertEqual: function assertEqual(a, b) {
            if (a != b)
              throw new Error(a + " != " + b);
          }
        };
        tests.push(test);
      }

    listener.onReady(tests);
    var nextTest = 0;

    function runNextTest() {
      if (nextTest < tests.length) {
        var test = tests[nextTest];
        listener.onRun(test);
        test.skip = function() {
          listener.onSkip(this);
          setTimeout(runNextTest, 0);
        };
        test.done = function() {
          listener.onFinish(this);
          setTimeout(runNextTest, 0);
        };
        test.func.call(container, test);
        if (!test.isAsync)
          test.done();
        nextTest++;
      }
    }

    runNextTest();
  }
};
```

The Suite

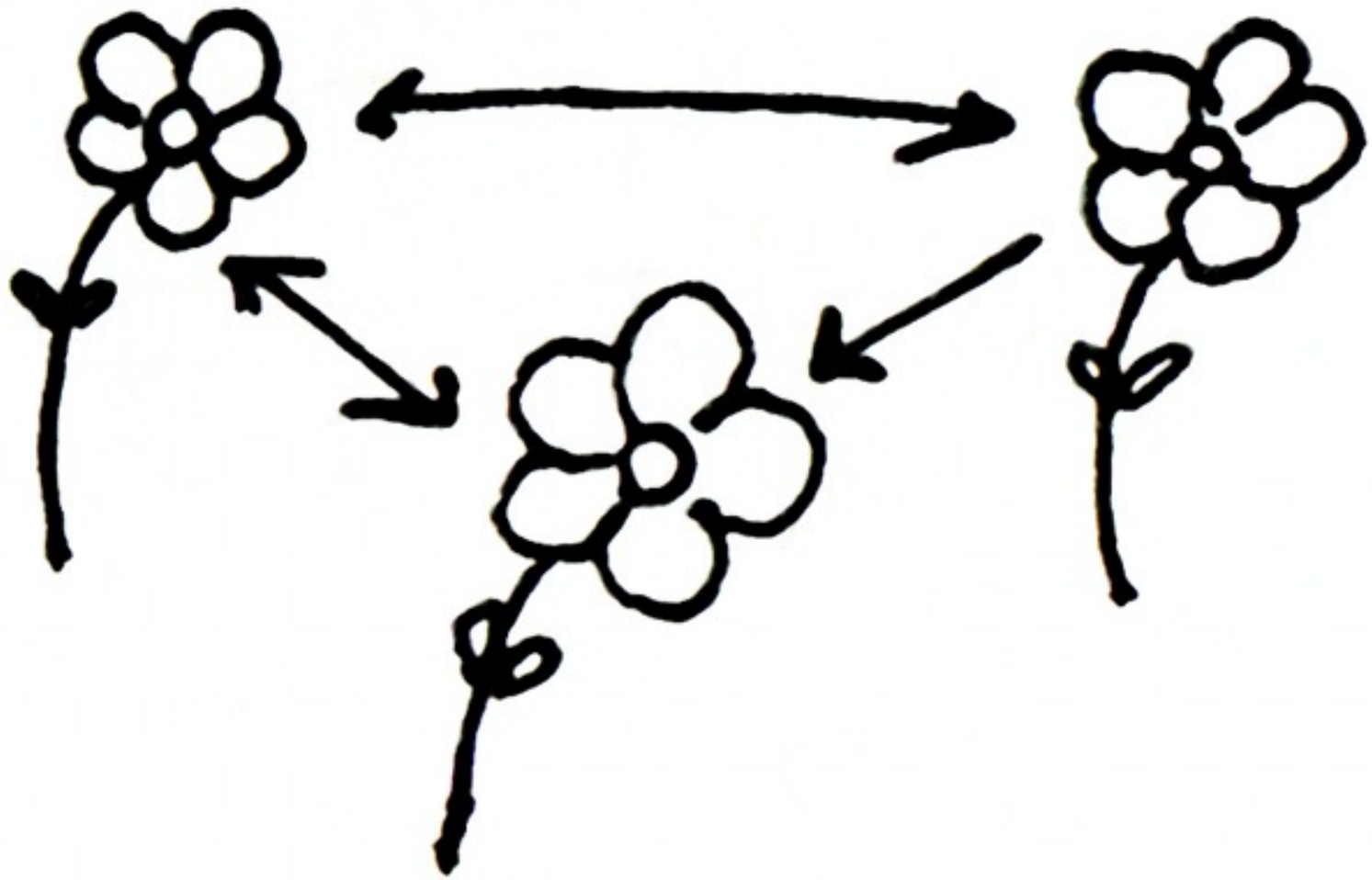
The `Tests` namespace contains the actual testing suite for BrowserCouch.

```
var Tests = {
  testDictionary: function(self) {
    var dict = new BrowserCouch._Dictionary();
    dict.set('foo', {a: 'hello'});
    dict.set('bar', {b: 'goodbye'});
    self.assertEqual(dict.get('foo').a, 'hello');
    self.assertEqual(dict.get('bar').b, 'goodbye');
    self.assertEqual(dict.getKeys().length, 2);
    self.assertEqual(dict.has('foo'), true);
    self.assertEqual(dict.has('bar'), true);
    self.assertEqual(dict.has('spatula'), false);
    dict.remove('bar');
    self.assertEqual(dict.getKeys().length, 1);
    self.assertEqual(dict.has('foo'), true);
  },
  _setupTestDb: function(cb) {
    var documents = this._testDbContents;
    BrowserCouch.get(
      "blarg",
      function(db) {
        db.wipe(
          function() {
            db.put(
              documents,
              function() {
                ModuleLoader.require(
                  "JSON",
                  function() { cb(db); }
                );
              }
            );
          });
      },
      new FakeStorage()
    );
  },
  _testDbContents: [{id: "monkey",
                     content: "hello there dude"},
```

Programming at the Edge



Replication



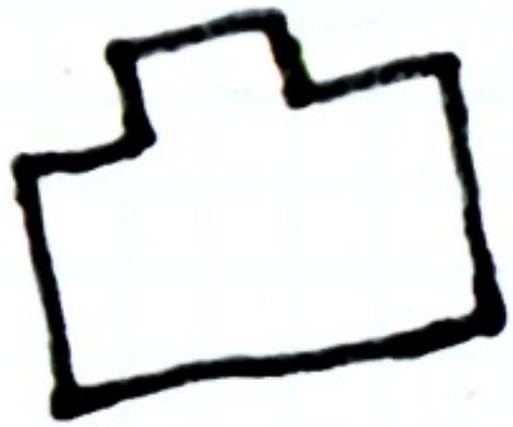
"normal"

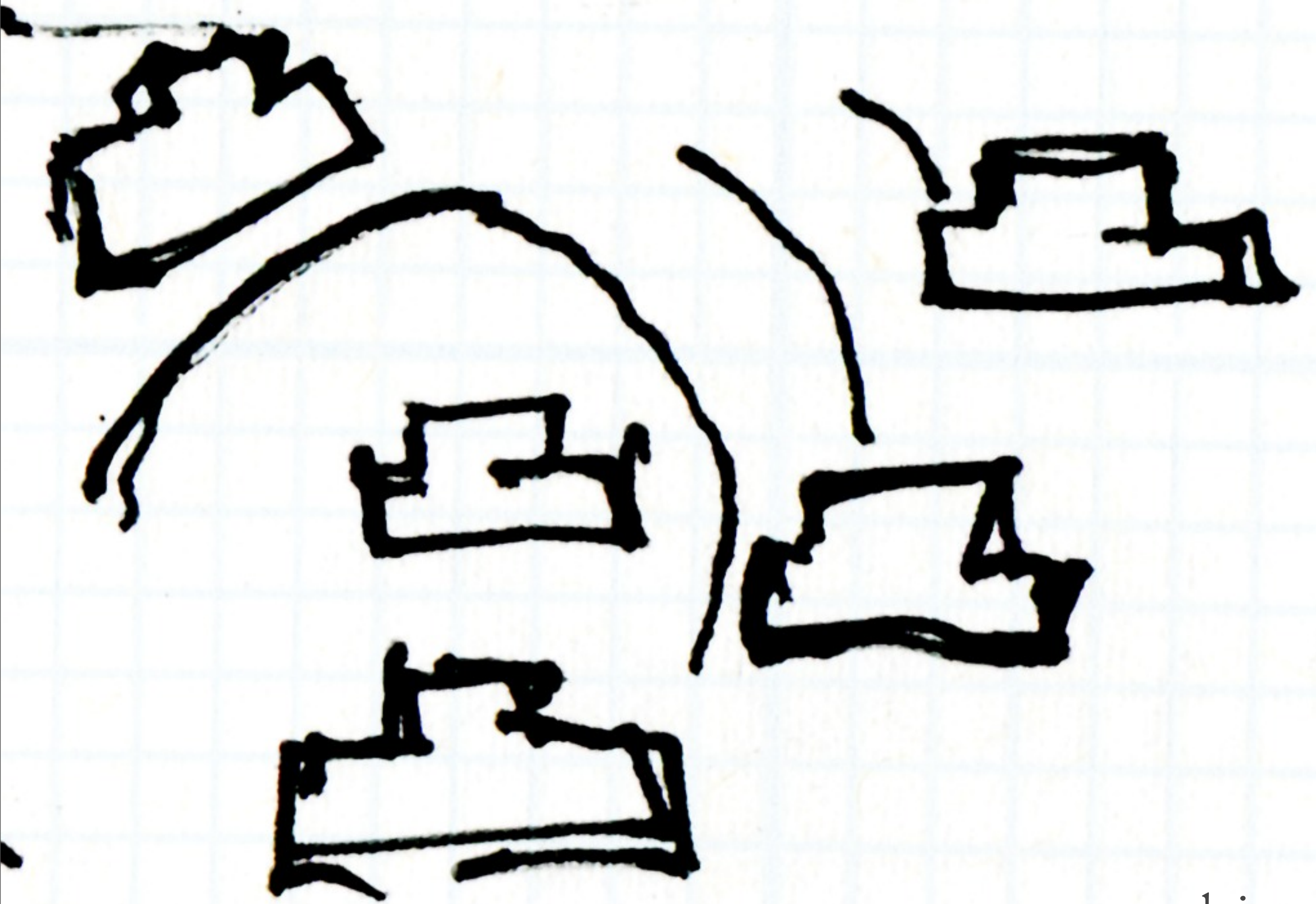
server

offline mode

the p2p

web





Apache CouchDB

Agenda

- Introduction
- Installation
- API by Example
- Client Libraries
- Design Documents
- CouchApps
- **Advanced Views**
- Replication
- Distributed Systems
- Deployment
- Internals
- Writing Extensions

Views are secondary indexes for your data

- single dimensional indexes
- (just a sorted list)
- derived from documents
- can be used to “join” documents

filtering & sorting documents

```
// all the playlists with more than 10 tracks, by title
function(doc) {
  if (doc.type == "playlist" && doc.tracks.length > 10) {
    emit(doc.title, doc.tracks);
  }
};
```

by field, by date, by X

```
// all blog posts, sorted by date
function(doc) {
  if (doc.type == "post") {
    emit(doc.created_at, doc.body);
  }
};
```

view query parameters

```
key
startkey
startkey_docid
endkey
endkey_docid
limit
stale
descending
skip
group
group_level
reduce
include_docs
```

key=JSON

- /db/_design/name/_view/vname?key=JSON
- key=″a string″
- key=[1,2,3]
- key={“a″:true}
- key=42

startkey=JSON

- /db/_design/name/_view/vname?startkey=JSON
- startkey=”a string”
- startkey=[1,2,3]
- startkey={“a”:true}
- startkey=42

startkey_docid=ID

- /db/_design/name/_view/vname?startkey=JSON
  &startkey_docid=ID
- startkey="a string"
  &startkey_docid=22f9fd9a8c693137
- startkey=[1,2,3]
  &startkey_docid=22f9fd9a8c693137
- startkey={"a":true}
  &startkey_docid=22f9fd9a8c693137

endkey=JSON

- /db/_design/name/_view/vname?endkey=JSON
- endkey="a string"
- endkey=[1,2,3]
- endkey={"a":true}
- endkey=42

endkey_docid=ID

- Just like startkey_docid, but for the end of the key range.
- (These are both used for more granular control of rows)
- Useful when there are more rows with a particular key than you want to show on a single page.

limit=NUMBER

- /db/_design/name/_view/vname?limit=10
- stops view output after N rows
- useful for pagination

skip=NUMBER

- /db/_design/name/_view/vname?skip=2
- skips N rows before beginning view output
- skip is expensive (like a table scan)
- should not be used for pagination
- used to fine tune query boundaries

stale=ok

- /db/_design/name/_view/vname?stale=ok
- Gives the current available view index
- Does not update the view
- Results in low-latency responses and less server load
- **NOTE**: behaves just like the default if there is not a view index in local process cache.

descending=true

- /db/_design/name/_view/vname?descending=true
- **NOTE**: requires reversing startkey and endkey
- ?descending=true&startkey="A"&endkey="Z" (returns no rows)
- ?descending=true&startkey="Z"&endkey="A"

group_level=NUMBER

- /db/_design/name/_view/vname?group_level=2
- **NOTE**: Only applies to reduce views

Map Keys	Group Level 1	Group Level 2
["a",1,1]	["a"]	["a",1]
["a",3,4]		["a",3]
["a",3,8]		
["b",2,6]	["b"]	["b",2]
["b",2,6]		
["c",1,5]	["c"]	["c",1]
["c",4,2]		["c",4]

group=true

- /db/_design/name/_view/vname?group=true
- Equivalent to group_level=Infinity
- One output row for each unique key

reduce=false

- /db/_design/name/_view/vname?reduce=false
- Gives access to the raw map rows on a view with a reduce function

include_docs=true

- /db/_design/name/_view/vname?include_docs=true
- includes the original document in the view row
- **NOTE**: only works with map views (or reduce=false)

multi-key

- POST /db/_design/name/_view/vname

```
{"keys": [
   "key1",
   "key2",
   ...
]}
```

- NOTE: can be used with include_docs=true and the _all_docs view to load many documents in one request

Collation Tricks by example

- Views merely sort rows by key
- Using the correct tricks, it's easy to get the right documents to sort next to each other in the same view.
- Useful for "joins", path-lookups, transaction logs, etc

"JOIN"s

- Joining blog posts and comments so that they can be loaded in a single query
- http://www.cmlenz.net/archives/2007/10/couchdb-joins
- Posts have doc._id
- Comments have doc.post_id
- Collate so each post is followed by its comments

"JOIN"s

```
/db/_design/name/_view/vname?startkey=["erlang-factory"]
&endkey=["erlang-factory",{}]
```

trick to sort last

```
function(doc) {
  if (doc.type == "post") {
    // posts sort first
    emit([doc._id,0], null);
  } else if (doc.type == "comment") {
    // comments are collated after their posts,
    // in chronological order
    emit([doc.post_id, doc.created_at], null);
  }
};
```

Read-mostly trees

- Store full path to each node on document

```
{
   "planet" : "Earth",
   "country" : "USA",
   "state" : "Texas",
   "county" : "Travis",
   "city" : "Austin",
   "street" : "Guadalupe"
   "shop" : "Amy's Ice Cream"
}
```

Read-mostly trees

- Emit full path to each node

```
function(doc) {
  var pathToShop = ["planet", "country", "state",
    "county", "city", "street"].map(function(field) {
    return doc[field];
  });
  emit(pathToShop, doc.shop);
};
```

- All nodes contained within a given parent are found within the parent key-range

Banking

- Transaction-log (one document per each transaction)

```
{
  "withdraw" : "9A724DF6234F4",
  "deposit" : "2B45E79D6C134",
  "amount" : 3500
}
```

Banking

- Map to sort by account

```
function(transaction) {
  emit(transaction["withdraw"], (-1 * transaction["amount"]));
  emit(transaction["deposit"], transaction["amount"]);
};
```

- Reduce to determine balance

```
function(keys, values, rereduce) {
  return sum(values);
};
```

View Performance tricks

CC-BY http://www.flickr.com/photos/theancientbrit/297520575/

design doc strategies

- Convenient to develop with independent design docs
- Each document in the database is converted to JSON and sent to the view server once per each design doc.
- Consolidate design documents for production

small values

- Large values cause B-tree overflow
- “Tall” vs “Wide” lists
- (Don’t make wide lists)
- Avoid putting extra data in your emits
- Non-string keys are collated faster

alternate view servers (erlang)

- Parallelizes view computation
- Avoids IPC and JSON conversion
- Another 5-10x speedup

eep0018

- Native (C-based) JSON parsing for Erlang
- Between 2 and 10 x faster
- Needs review

Apache CouchDB

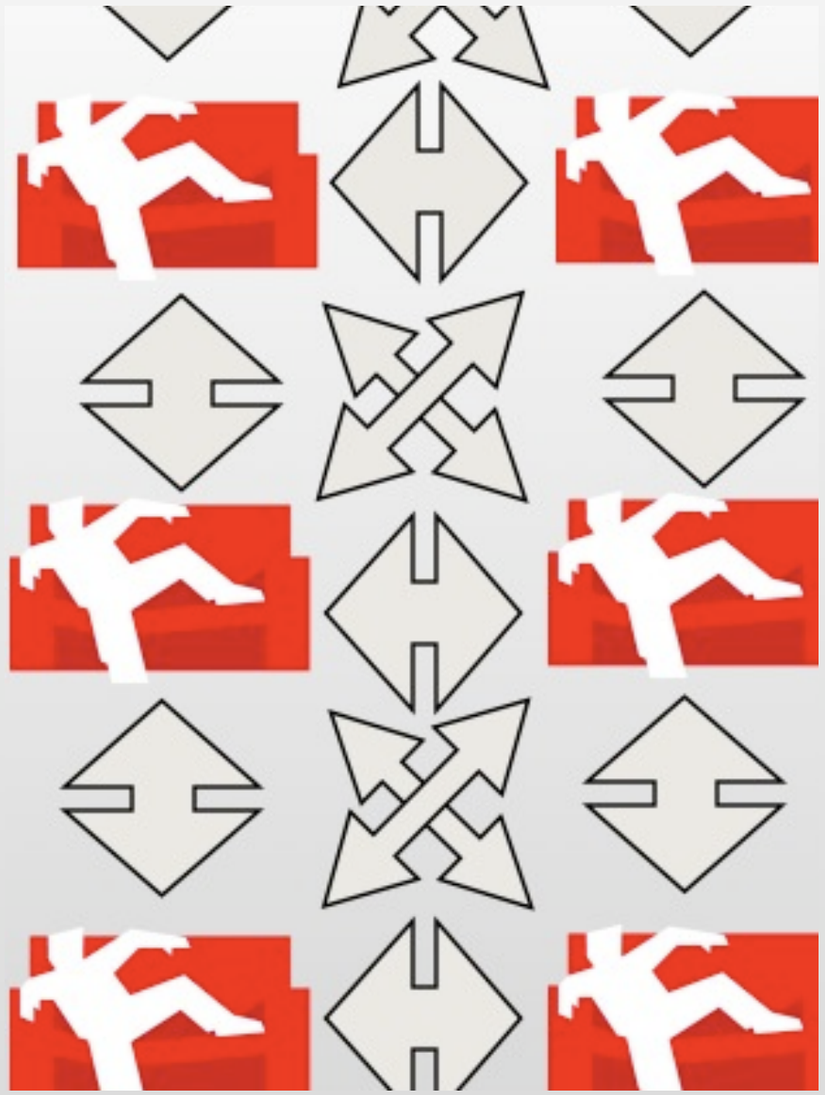
couch.io

Agenda

- Introduction
- Installation
- API by Example
- Client Libraries
- Design Documents
- CouchApps
- Advanced Views
- **Replication**
- Distributed Systems
- Deployment
- Internals
- Writing Extensions

Replication Basics Recap

- Event-based / triggered
- Unidirectional
- Conflict
  - detection
  - resolution mechanism



Triggering Replication

Local

POST http://couch.io/_replicate

```
{"source":"db",
  "target":"db-replica"}
```

- Good for…
  - backup
  - testing
  - development

Triggering Replication

Pull

POST http://couch.io/_replicate

{"source":"http://sofa.net/db",
"target":"db-replica"}

- Good for…
  - node-synchronization
  - offline-work
  - highly available clusters
  - data distribution

Triggering Replication

Push

POST http://couch.io/_replicate

{"source":"db",
"target":"http://sofa.net/db-replica"}

- Good for…
  - node-synchronization
  - offline-work-*sync*
  - highly available clusters
  - data distribution

Triggering Replication

Remote

POST http://couch.io/_replicate

{"source":"http://couch.io/db",
"target":"http://sofa.net/db-replica"}

- Good for…
  - node-synchronization
  - custom architectures
  - cluster management

Bi-directional replication

POST http://couch.io/_replicate

{"source":"http://sofa.net/db",
"target":"db-replica"}

POST http://couch.io/_replicate

{"source":"db-replica",
"target":"http://sofa.net/db"}

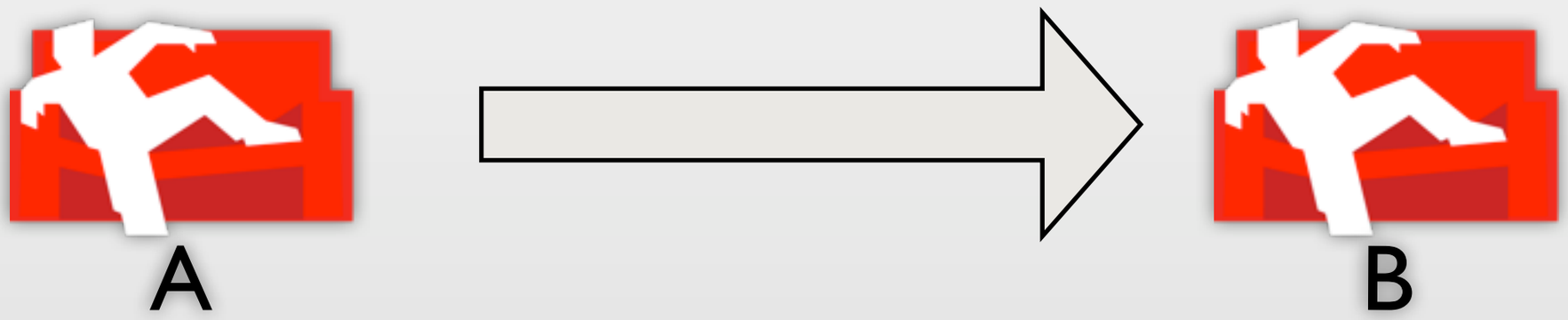
Bi-directional replication

Slide from the future

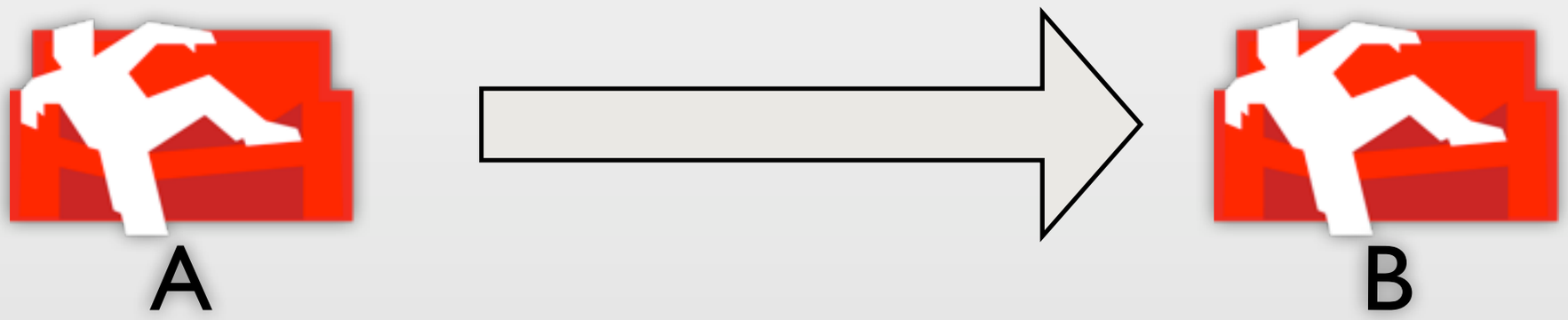
POST http://couch.io/_replicate?bidirectional=true

{"source":"http://sofa.net/db",
"target":"db-replica"}

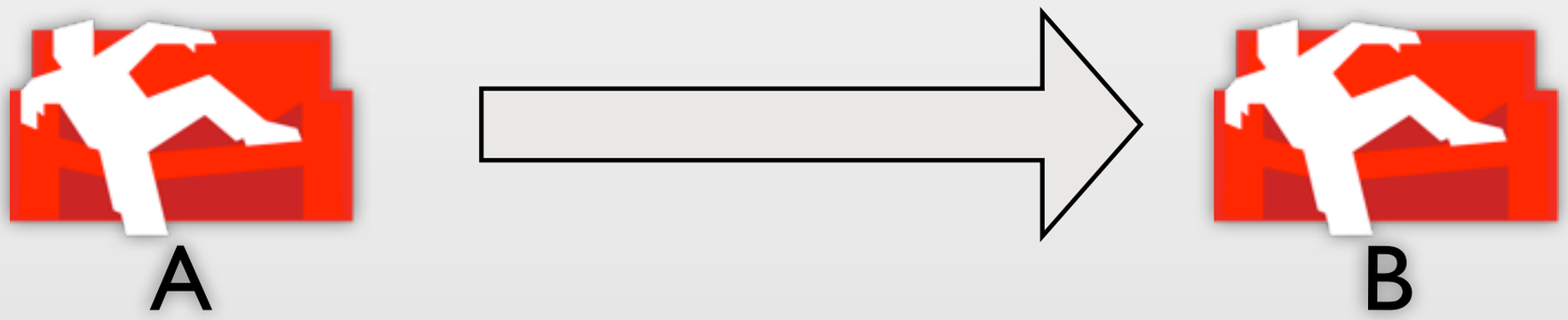
Conflict resolution by example



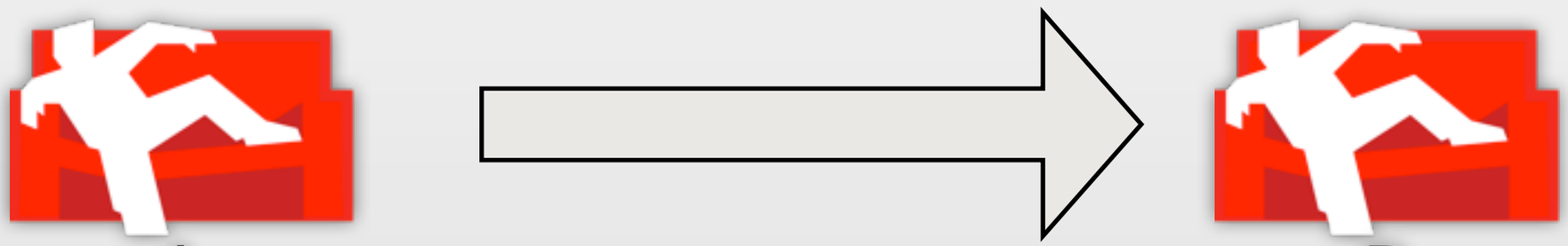
Conflict resolution by example



Conflict resolution by example



Conflict resolution by example



Conflict resolution by example

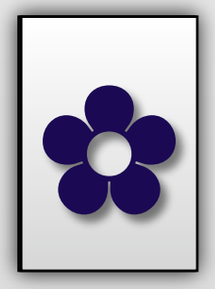
A

B

Conflict resolution by example

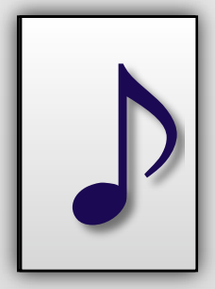
A

B

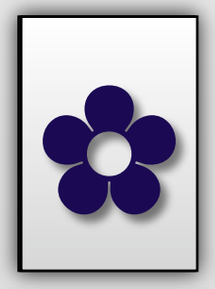


Conflict resolution by example

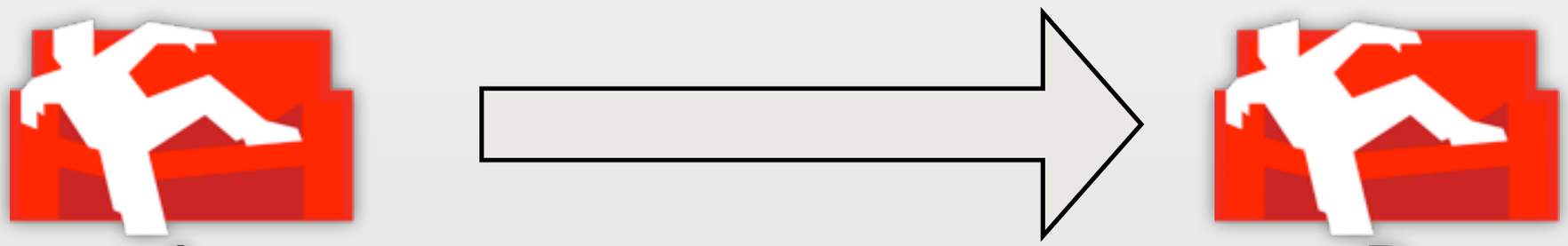
A

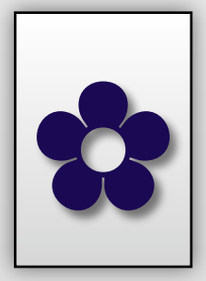


B

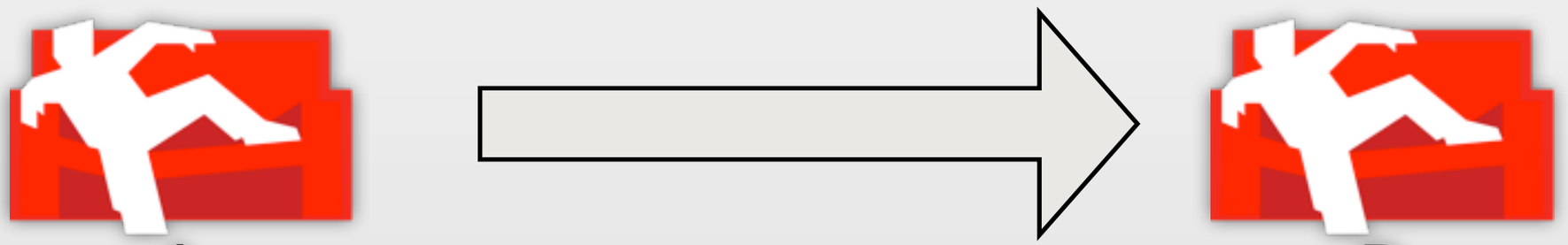


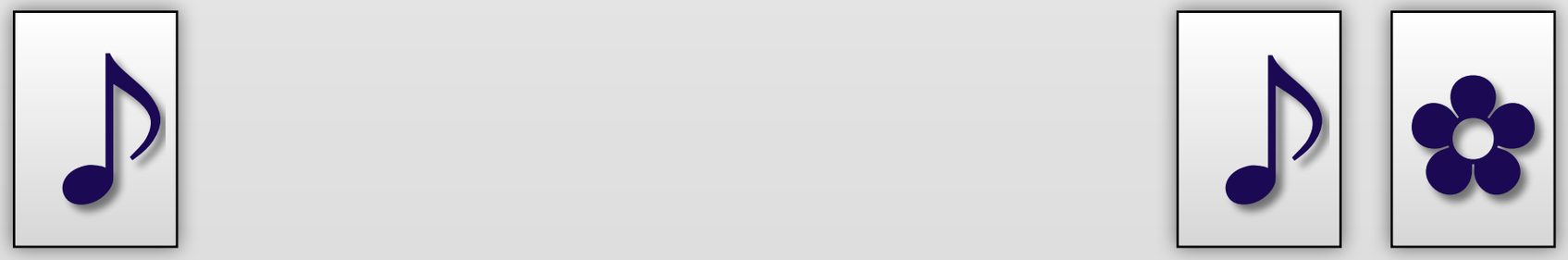
Conflict resolution by example



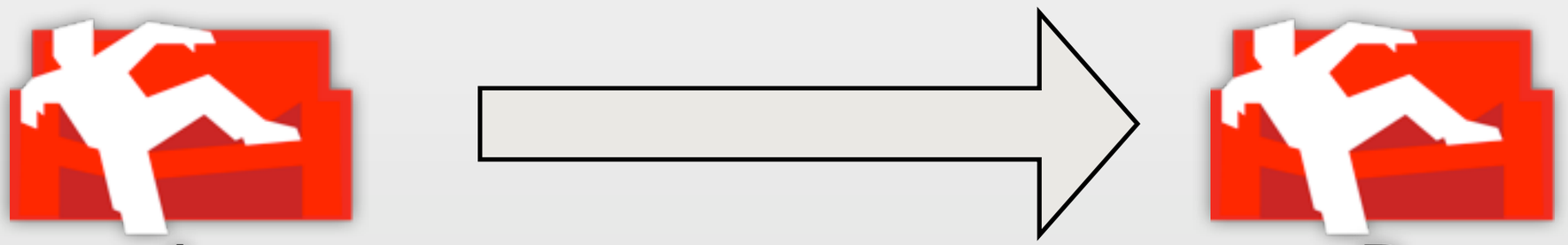


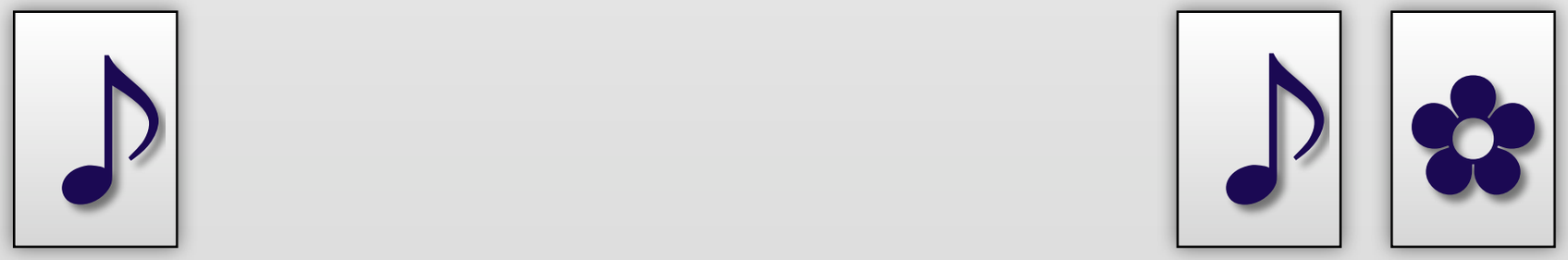
Conflict resolution by example



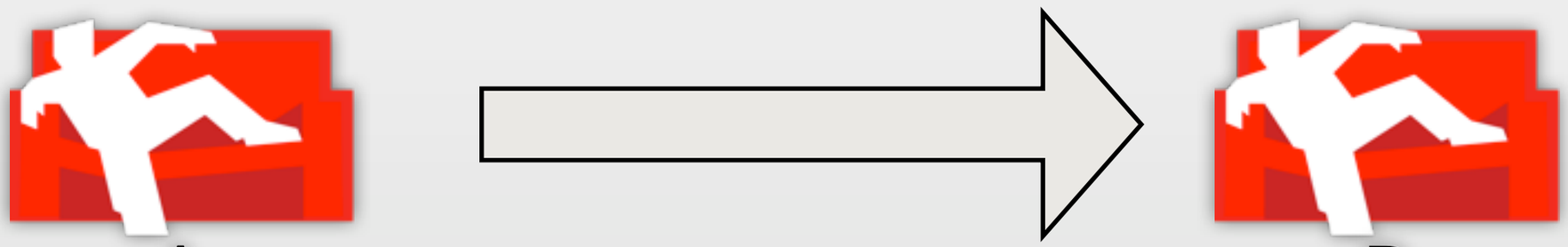


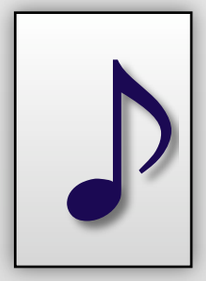
Conflict resolution by example





Conflict resolution by example





Conflict resolution by example

```
> curl -X PUT \
  http://localhost:5984/db/doc \
  -d '{"image":"❦"}'
{"ok":true,"id":"doc","rev":"1-3974263740"}
```

Conflict resolution by example

```
> curl -X POST \
  http://localhost:5984/_replicate \
  -d '{"source":"db", \
       "target":"db-replica"}'

{"ok":true,"session_id":"4187b9c0f21cf54b1
66037fd0d8c9d6c","docs_written":1...
```

Conflict resolution by example

```
> curl -X GET \
  http://localhost:5984/db/doc
```

```
{"_id":"doc",
 "_rev":"1-3974263740",
 "image":"\u2766"}
```

Conflict resolution by example

```
> curl -X GET \
  http://localhost:5984/db-replica/doc
```

```
{"_id":"doc",
 "_rev":"1-3974263740",
 "image":"\u2766"}
```

Conflict resolution by example

```
> curl -X PUT \
  http://localhost:5984/db-replica/doc
  -d '{"image":"♪", \
       "_rev":"1-3974263740"}'

{"ok":true,
 "id":"doc",
 "rev":"2-2709835302"}
```

Conflict resolution by example

```
> curl -X PUT \
  http://localhost:5984/db-replica/doc
    -d '{"image":"✿", \
        "_rev":"1-3974263740"}'

{"ok":true,
 "id":"doc",
 "rev":"2-53319531"}
```

Conflict resolution by example

```
> curl -X POST \
  http://localhost:5984/_replicate \
  -d '{"source":"db", \
       "target":"db-replica"}'

{"ok":true,"session_id":"f62b4b134aab24a7e
a060091c2009e62","docs_written":1...
```

Conflict resolution by example

```
function(doc) {
  if(doc._conflicts) {
    for(var idx in doc._conflicts) {
      emit(doc._conflicts[idx], null);
    }
  }
}
```

Conflict resolution by example


```
{
  "total_rows":1,
  "offset":0,
  "rows":[
    {
      "id":"doc",
      "key":"2-2709835302",
      "value":null
    }
  ]
}
```

Conflict resolution by example

```
> curl -X GET \
  http://localhost:5984/db-replica/doc?
conflicts=true

{
 "_id":"doc",
 "_rev":"2-53319531",
 "image":"\u273f",
 "_conflicts":["2-2709835302"]
}
```

Conflict resolution by example

```
> curl -X DELETE \
  http://localhost:5984/ \
  db-replica/doc?rev="2-2709835302"

{
 "ok":true,
 "id":"doc",
 "rev":"3-313142706"
}
```

Conflict resolution by example

```
> curl -X GET \
  http://localhost:5984/db-replica/doc?
conflicts=true

{
 "_id":"doc",
 "_rev":"3-313142706",
 "image":"\u273f",
}
```

Conflict resolution by example

```
> curl -X POST \
  http://localhost:5984/_replicate \
  -d '{"source":"db-replica", \
       "target":"db"}'

{"ok":true,"session_id":"f47226e71783a22f5
3cbddf9380a3ebd","docs_written":1...
```

Conflict resolution by example

```
> curl -X GET \
  http://localhost:5984/db/doc?conflicts=true

{
 "_id":"doc",
 "_rev":"3-313142706",
 "image":"\u273f",
}
```

Architectures

couch.io

Client-Server

- LAMP -> LACP
- LYME -> LYCE (DO'H!)
- Well understood, widely deployed
- Boooring

Offline mode or Gears done right

- CouchDB local to the user
- Replication to sync data
- Zero latency
- SQL in HTML?!
  - WHATWG / HTML 5 WGs need a little ~~spanking~~pushing

The Peer-to-Peer Web

- "Ground Computing"
- Build your own cloud
  - Google / Amazon / Apple don't have to know everything
    - Legal issue make cloud services a non-option

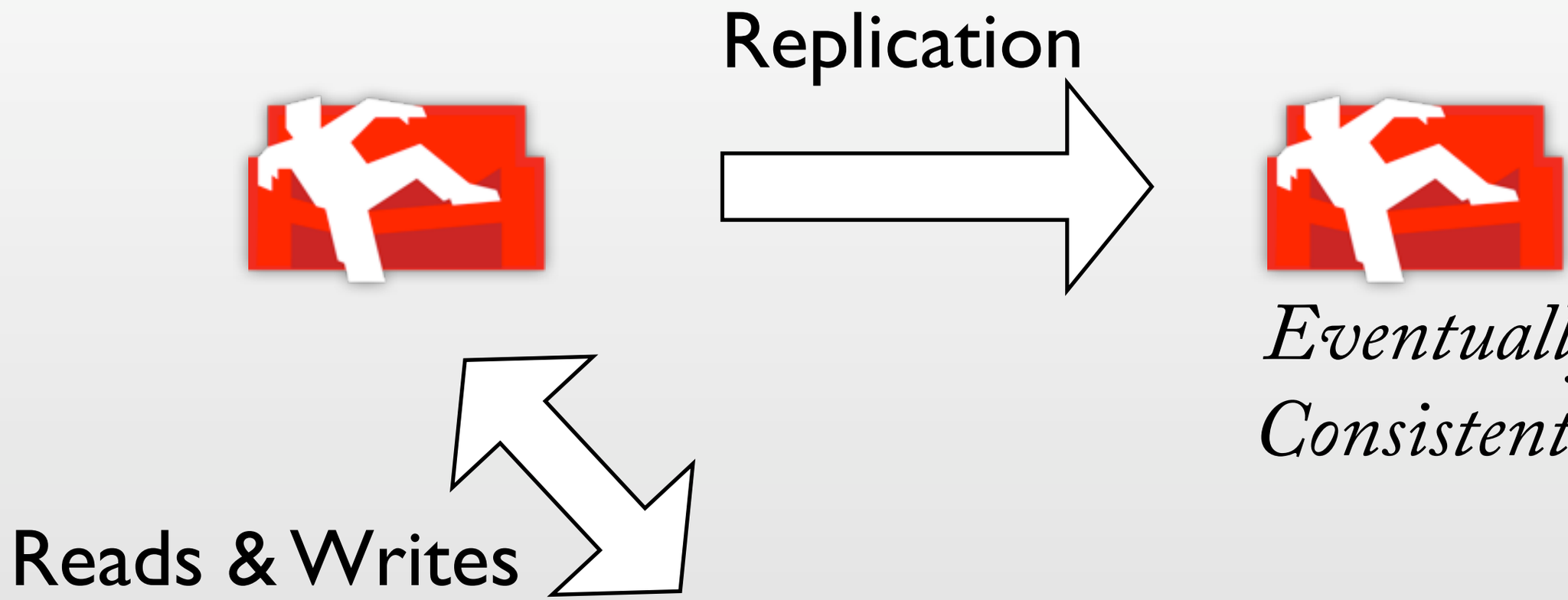
Growing Up

- Using replication to allow more
  - reads
  - writes
  - data

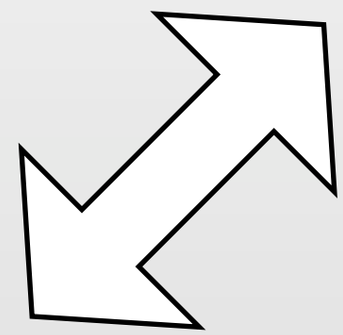
Master / Slave

couch.io

Master / Slave — Setup

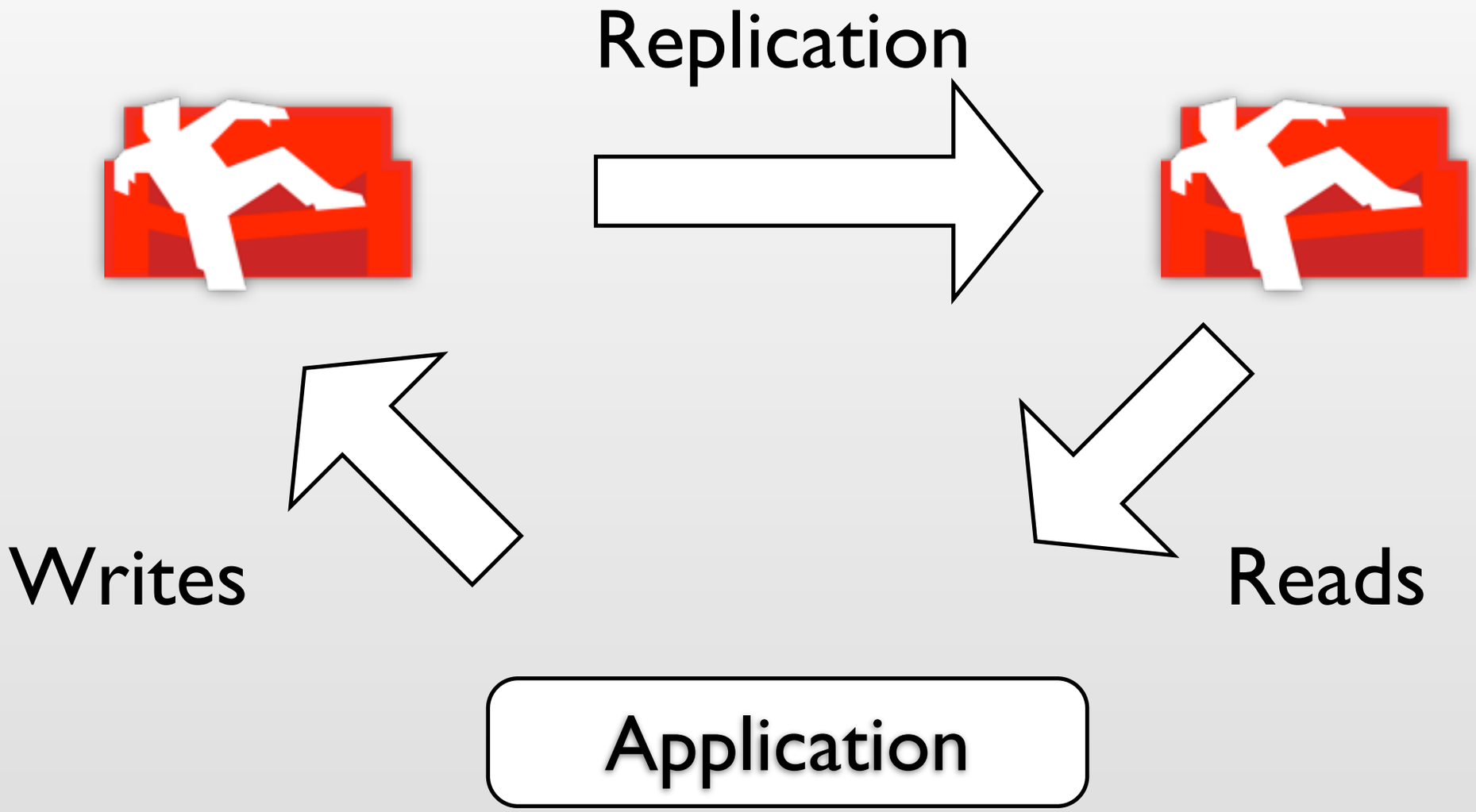


Master / Slave — Setup

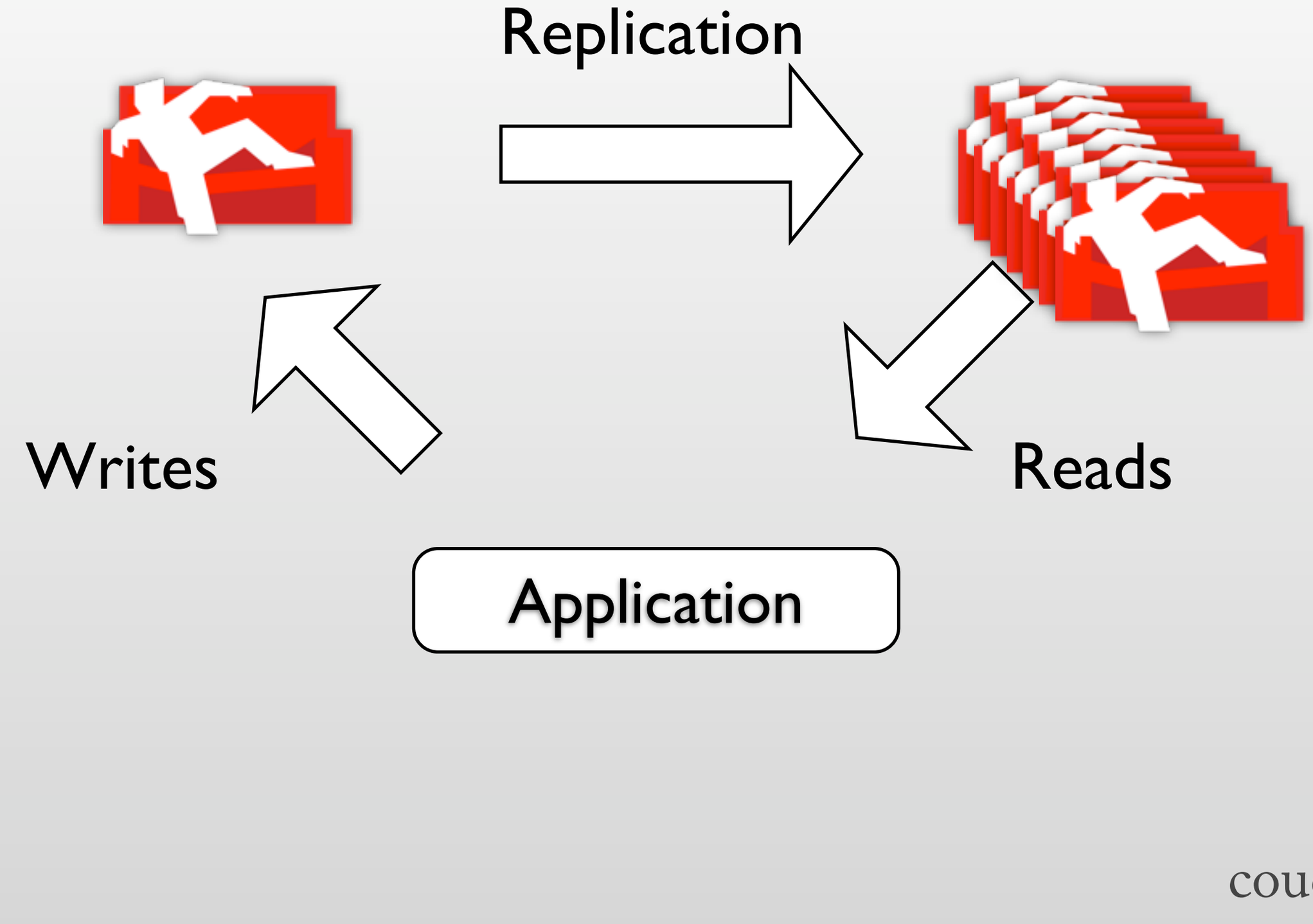


Application

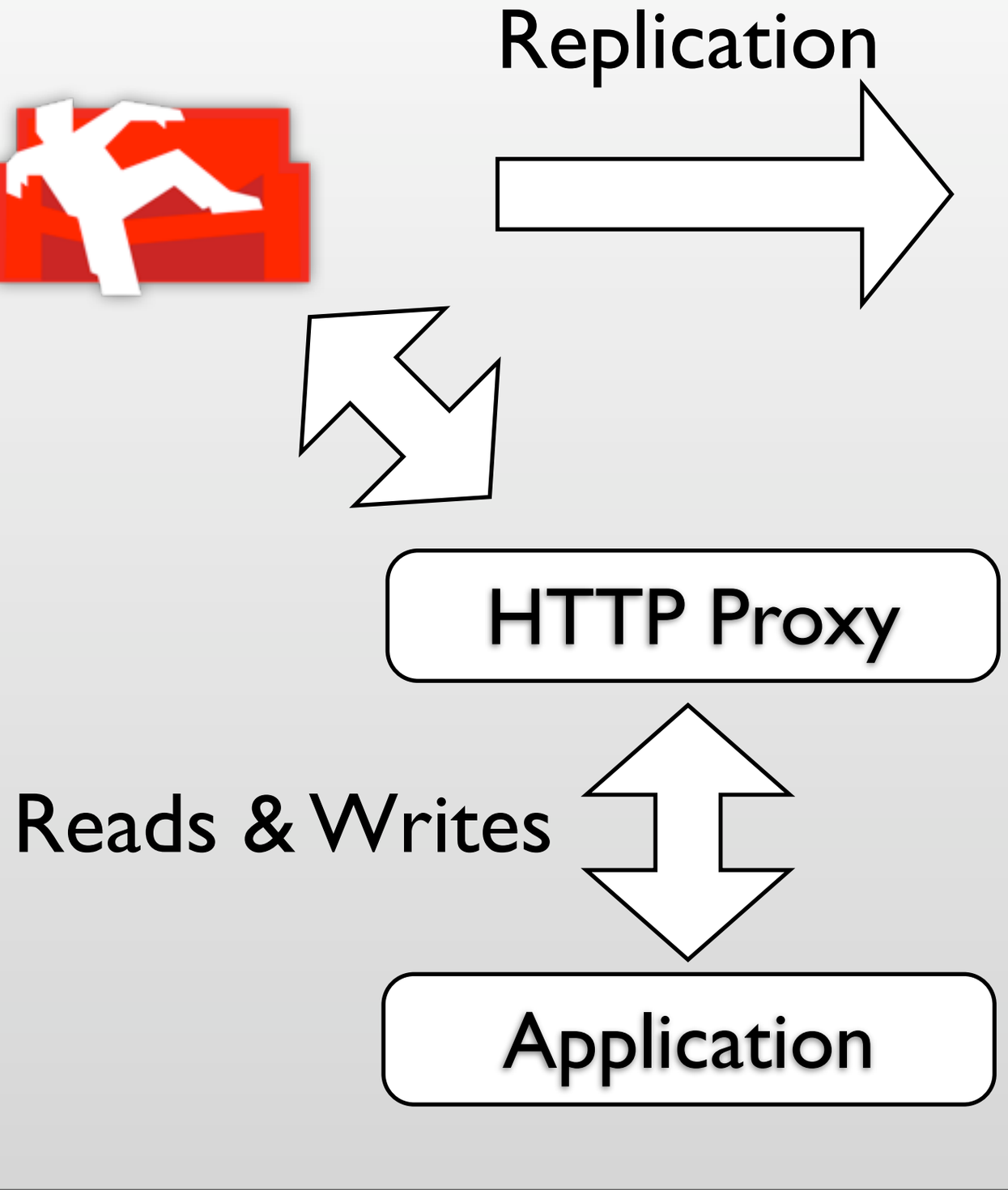
Master / Slave — Setup



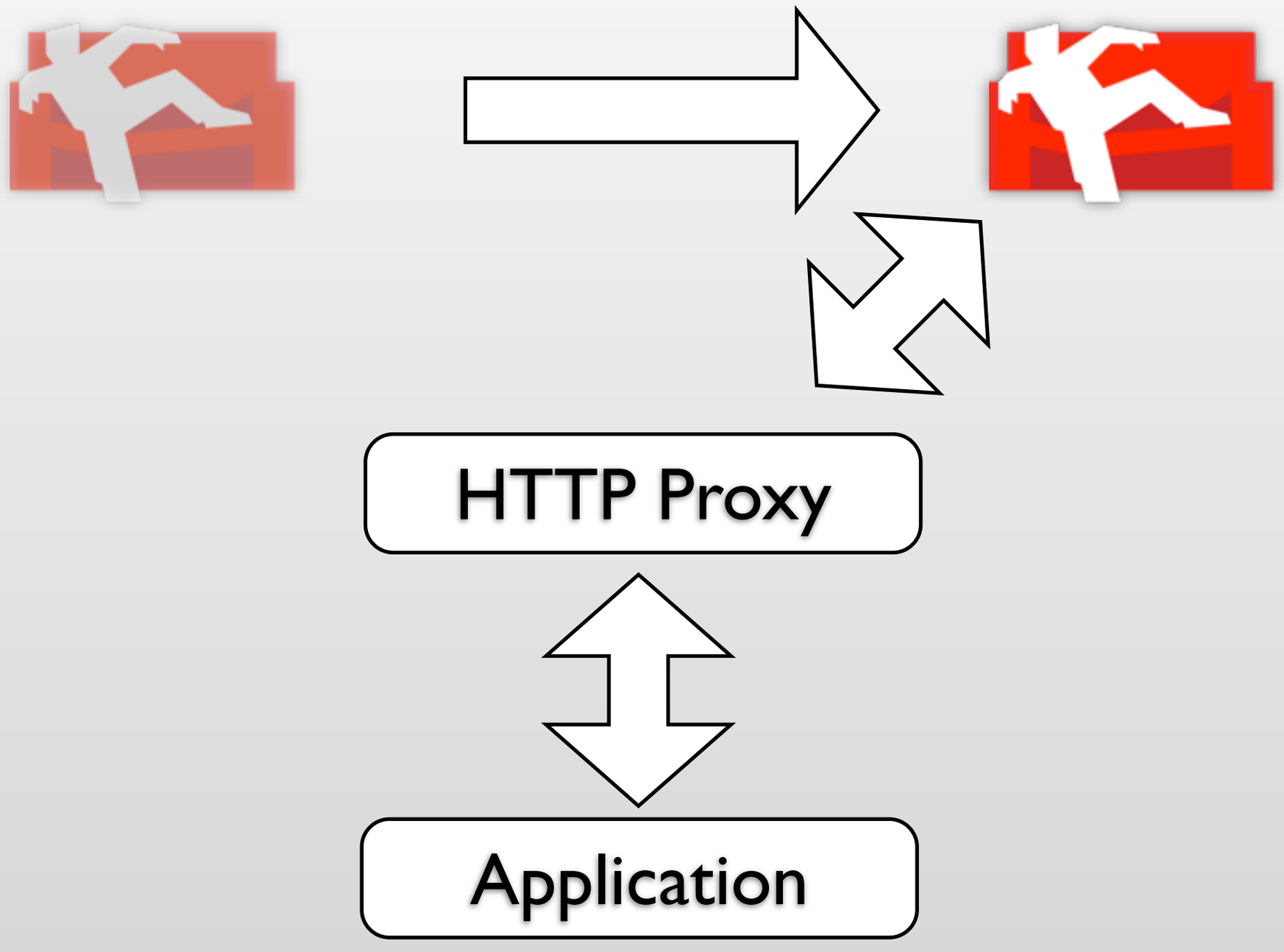
Master / Slave — Setup



Master / Slave — Setup



Master / Slave — Setup



Master / Slave — Setup

- HTTP Proxy Options
  - Apache 2.x & mod_proxy
    - AllowEncodedSlashes On(!)
  - Varnish
  - ~~nginx,~~ can't disable %2F -> /

Master / Slave — Setup

```
<VirtualHost *:80>

  ServerName username.couch.io
  ProxyRequests Off
  ProxyPass / http://127.0.0.1:5984/ nocanon
  ProxyPassReverse / http://127.0.0.15984/
  AllowEncodedSlashes On

</VirtualHost>
```

Master / Slave Use Cases

- Hot failover
- Backup
- Asynchronous calculation (stats, roll-ups)
- Serve more read requests

Problems & Solutions

- Nodes Down
- Adding Nodes

Node Down

- Varnish & nginx “backend” systems detect dead nodes and re-route traffic out of the box
- Heatbeat & Wackamole / Spread can reassign IP addresses in failure situations
- The application could maintain a list of nodes and skip if they timeout

Proxy Method

- Pro
  - Easy to set up
  - Well known
  - Many options
- Con
  - Configuration is static
  - Single proxy not fault tolerant
  - Large “routing table” can be hard to maintain or slow things down

IP Failover

- Pro
  - Robust, fast
  - Fewer moving parts
  - Software agnostic
- Con
  - Configuration is static
  - “Wastes” IP addresses
  - Doesn’t work on EC2

Application Decides

- Pro
  - Very simple
- Con
  - Not application agnostic
  - Tight coupling
  - For multiple app servers, needs atomic updates

Adding Nodes

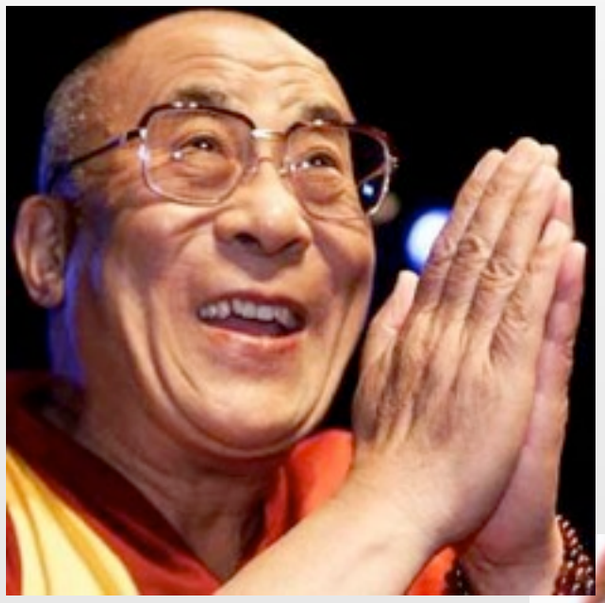
- Replication is not fastest solution
- “Priming” nodes with rsync / scp
  - Future Bullet Point: Initialize replication after rsync
- Use Replication to “catch up”

Multi Master

- Multiple distributed primary nodes
- Good for…
  - fault tolerance
  - geographical distribution
    - Local data is king

Multi Master Setup

- Same as slave / read cluster:
  - HTTP Proxy
  - IP Failover
  - Application option



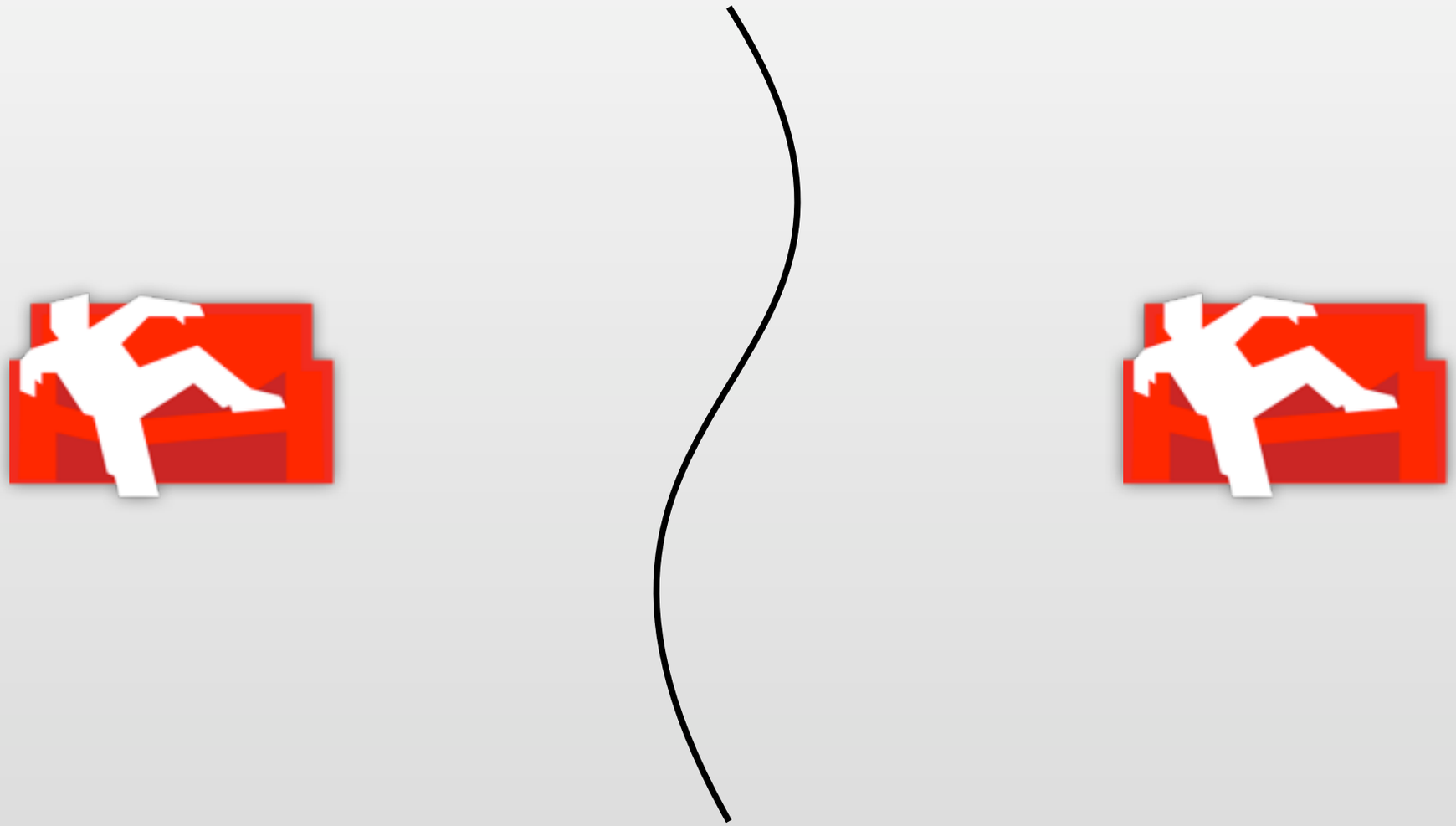
Problems & Solutions

- Master down
- Adding a master
- Split brain

Master Down / Up

- Same as slave up / down
- Needs 3rd part tier to reroute traffic

Split Brain



Setting up auto-replication

couch.io

Cron

- Just a curl call
- * * * * * /usr/local/bin/curl http://127.0.0…

Update Notifications

- Show replication_helper.py
  - in TextMate, stupid

Comet-style API

- Long polling GET
  - against docs
    - and views
- /_changes
- replication

Partitioned Cluster w/ CouchDB-Lounge

Consistent Hashing

HTTP proxy

Nginx / Twisted Python

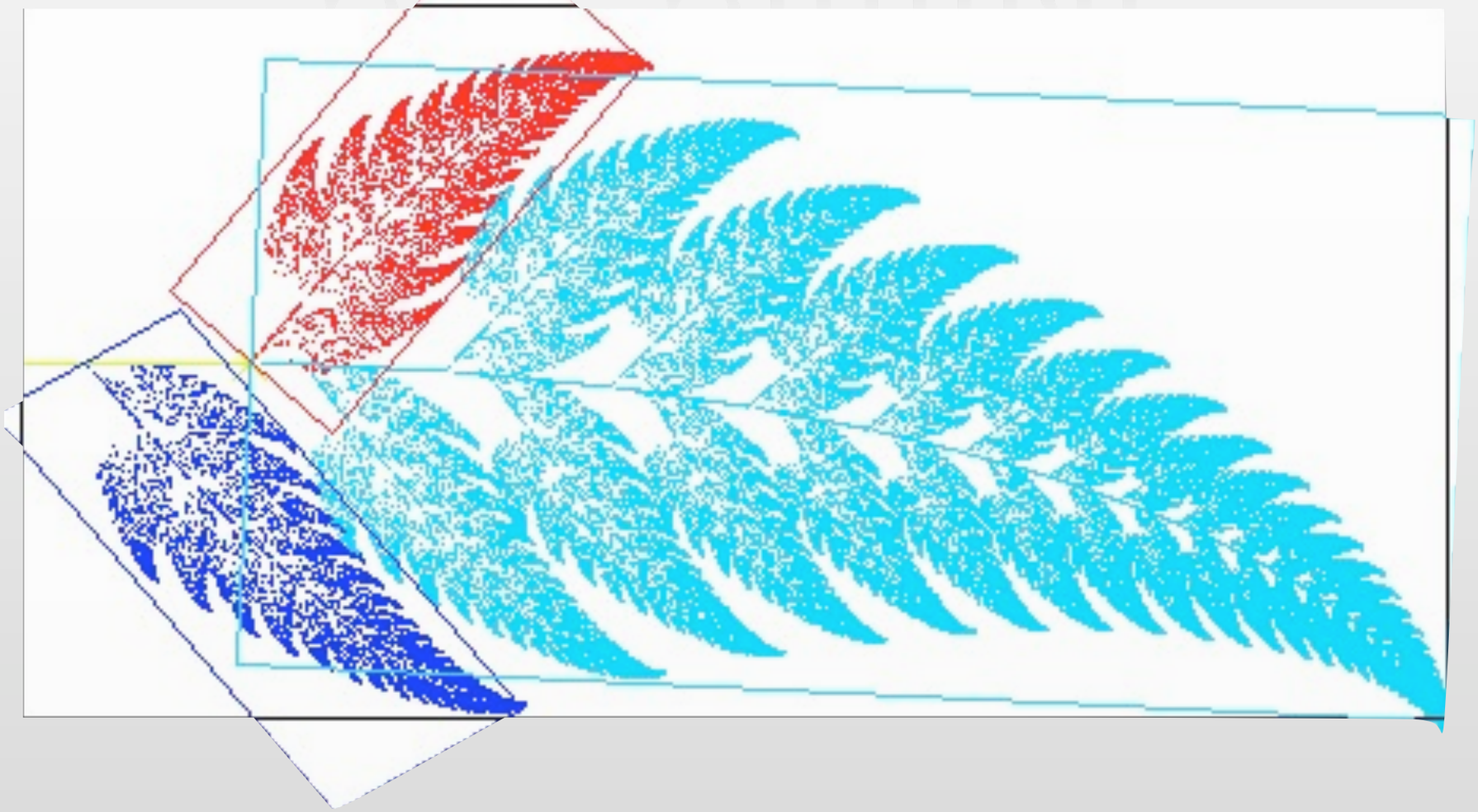
Many machines == 1 Couch

Replication for data scaling (preview)

- Filtered replication
- Replicates only a subset of data from source
- Allows to split a node in two (or N), by replicating 50% or 100/N% each to new nodes and repurpose the old node.

Self-Similar

Self-Similar
Self-Similar



Apache CouchDB

Agenda

- Introduction
- Installation
- API by Example
- Client Libraries
- Design Documents
- CouchApps
- Advanced Views
- Replication
- Distributed Systems
- **Deployment**
- Internals
- Writing Extensions

couchapp recap

- Not really, you should have gotten it by now

Importing Data

- bulk docs (say 1000, maybe)
- use sequential _ids
- a spindle per db
- update views periodically
  (say, every 10,000 inserts)

Backup / Archiving

- cp -r /var/lib/couchdb /mnt/backup/
- Replication can work, too

Dealing with Compaction

- Not too much of an issue if:
  - Write mostly data
  - bulk inserts
- Needs consideration when:
  - Frequent updates
- Needs free space
- Impacts disk I/O
  - at the moment, blows FS cache

Updating Views

- stale=ok
- put major new or changed views into new design docs, let them build and swap on the application level
- better to have fewer design documents. grouping views saves on IO and JSON parsing.

View-Update-Trick (preview)

- Deploy new views to a “staging” design doc
- Query the new view. When it returns, the index is built.
- COPY the staging design doc to the production URL, and the indexes will stay available for your users.

Monitoring CouchDB

- Monit
- SNMP
- rddtool
- Any HTTP-based tool

Runtime Configuration

- on-the-fly configuration / repurposing of a node

HTTP Tools

- curl
- Apache
- varnish
- nginx (+memcached)
- tinyproxy
- pound
- ab / httperf / tsung
- Firebug
- RFC 2616(!)

Shared Hosting

- A separate port or IP address per instance
- (Maybe) a system user per instance
- User quotas
- mod_proxy to federate
- HTTP auth

Security

- Regular UNIX daemon setup practices apply
- Admin accounts & design documents
- Separate databases for separate users
- A node per user