CouchDB, PHP & PHPillow

http://joind.in/1465

Kore Nordmann
<kore@php.net>
@koredn

February 26, 2010

About me

- Kore Nordmann, <kore@php.net>
- Long time PHP developer
- Regular speaker, author, etc.
- Studies computer science in Dortmund, currently writing thesis
- Active open source developer:
  - eZ Components (Graph, WebDav, Document), *Arbit*, PHPUnit, Torii, *PHPillow*, KaForkL, Image 3D, WCV, ...

Outline

Introduction

General

Structure

API

Views

Consistency

PHPillow

Applications

QA

CouchDB is paradigmn shift

CouchDB is paradigmn shift

- Structure

CouchDB is paradigmn shift

- Structure
- Consistency

CouchDB is paradigmn shift

- Structure
- Consistency
- API

CouchDB is paradigmn shift

- Structure
- Consistency
- API
- Applications

Outline

Introduction

General

Structure

API

Views

Consistency

PHPillow

Applications

QA


CouchDB
relax

- Apache top-level project

- Apache top-level project
- Build in Erlang / on Erlang/OTP

Outline

Introduction

General

Structure

API

Views

Consistency

PHPillow

Applications

QA

Structure

- Document based database

Structure

- Document based database
- No pre-defined structure (tables)

Structure

- Document based database
- No pre-defined structure (tables)
- Put in any JSON object you want

Structure

- Document based database
- No pre-defined structure (tables)
- Put in any JSON object you want
  - Even deep structures (arrays of objects)

Structure

- Document based database
- No pre-defined structure (tables)
- Put in any JSON object you want
  - Even deep structures (arrays of objects)
  - You may attach any number of files to documents

Structure

- Example wiki document

```
{ "title":     "PHPUK 2010",
  "text":      "Welcome to the PHPUK...",
  "creator":   "user-bar",
  "edited":    2935678239,
  "revisions": [
    { "title":   "PHPUK 2009",
      "text":    "Welcome to the PHPUK...",
      "creator": "user-foo",
      "edited":  2935678183,
      ]
    },
    ...
  ]
}
```

Attachements

- Example wiki document

```
{ "title":       "PHPUK 2010",
  "creator":     "user-bar",
  "text":        "<h1>Welcome to the PHPUK</h1>

                  <img src=\"phpuk_2010/logo.png\" alt=\"
                      PHPUK logo\"/>
                  ...",
  "_attachments": {
    "logo.png": {
      "content_type": "image\/png",
      "stub":         true,
      "length":       42,
    }
  }
}
```

Migration / Refactoring

- Change document structure at any time

Migration / Refactoring

- Change document structure at any time
- No need for non-transaction-safe Data Definition Language (DDL)

Migration / Refactoring

- Change document structure at any time
- No need for non-transaction-safe Data Definition Language (DDL)
- Fits rapid development approaches with common customer requested changes to the data structure

Migration / Refactoring

- Change document structure at any time
- No need for non-transaction-safe Data Definition Language (DDL)
- Fits rapid development approaches with common customer requested changes to the data structure
  - You need to handle this in your application properly, of course:
    - Incrementally update structure on modification
    - Liberal validation on read

Outline

Introduction

General

Structure

API

Views

Consistency

PHPillow

Applications

QA

HTTP API

- RESTful HTTP access

HTTP API

- RESTful HTTP access
- HTTP is available on “all” platforms natively
  - No PHP extension required
  - Just use PHPs HTTP stream wrapper, pecl/http or curl

HTTP API

- RESTful HTTP access
- HTTP is available on “all” platforms natively
  - No PHP extension required
  - Just use PHPs HTTP stream wrapper, pecl/http or curl
- You can use all your known HTTP middleware
  - Reverse proxies for scaling reads (Varnish, Squid)
  - Simple custom proxy configuration for direct “AJAX” access

HTTP API

- GET / POST / PUT / DELETE
- `<METHOD> http://<host>/<database>/<document>`

Create a database

```
$ curl -i -X PUT http://localhost:5984/phpuk_wiki

HTTP/1.1 201 Created
Server: CouchDB/0.10.0 (Erlang OTP/R13B)
Location: http://localhost:5984/phpuk_wiki
Date: Fri, 13 Jan 2009 14:07:57 GMT
Content-Type: text/plain;charset=utf-8
Content-Length: 12
Cache-Control: must-revalidate

{"ok":true}
```

Create a wiki document


```
$ curl -i -X PUT http://localhost:5984/phpuk_wiki/Start
    --data '{"name": "Start", "text": "Hello World!"}'

HTTP/1.1 201 Created
Server: CouchDB/0.10.0 (Erlang OTP/R13B)
Location: http://localhost:5984/phpuk_wiki/Start
Etag: "1-6bfd4885b6c62bb5169a19d5a81927e3"
Date: Fri, 13 Jan 2009 14:14:55 GMT
Content-Type: text/plain;charset=utf-8
Content-Length: 68
Cache-Control: must-revalidate

{"ok":true,"id":"Start","rev":"1-6
    bfd4885b6c62bb5169a19d5a81927e3"}
```

Get a wiki document back


```
$ curl -i -X GET http://localhost:5984/phpuk_wiki/Start

HTTP/1.1 200 OK
Server: CouchDB/0.10.0 (Erlang OTP/R13B)
Etag: "1-6bfd4885b6c62bb5169a19d5a81927e3"
Date: Fri, 13 Jan 2009 14:15:48 GMT
Content-Type: text/plain;charset=utf-8
Content-Length: 97
Cache-Control: must-revalidate

{"_id":"Start","_rev":"1-6
    bfd4885b6c62bb5169a19d5a81927e3","name":"Start","
    text":"Hello World!"}
```

Security

- Simple database based access restrictions

Security

- Simple database based access restrictions
- Using HTTP plain auth

Security

- Simple database based access restrictions
- Using HTTP plain auth
- More fine grained access control will be in next release
  - Define functions which decide if a request from a user will be accepted.

Outline

Introduction

General

Structure

API

Views

Consistency

PHPillow

Applications

QA

Data access

- How to query such a mess?

Data access

- How to query such a mess?
  - Views are small scripts, run for all documents in a database

Data access

- How to query such a mess?
  - Views are small scripts, run for all documents in a database
  - Views are built iteratively, results stored in BTrees
    - Once built, they are *fast*

Data access

- How to query such a mess?
  - Views are small scripts, run for all documents in a database
  - Views are built iteratively, results stored in BTrees
    - Once built, they are *fast*
  - Mostly JavaScript, but also PHP, Ruby, Perl, Erlang, ...

Data access

- How to query such a mess?
  - Views are small scripts, run for all documents in a database
  - Views are built iteratively, results stored in BTrees
    - Once built, they are *fast*
  - Mostly JavaScript, but also PHP, Ruby, Perl, Erlang, ...
  - A view may emit any number of key-value pairs for each document

Data access

- How to query such a mess?
  - Views are small scripts, run for all documents in a database
  - Views are built iteratively, results stored in BTrees
    - Once built, they are *fast*
  - Mostly JavaScript, but also PHP, Ruby, Perl, Erlang, ...
  - A view may emit any number of key-value pairs for each document
  - Key and value may be any JSON structure

Wiki

- Index all wiki documents by their title

```
1  function( doc )
2  {
3      if ( doc.title && doc.text )
4      {
5          emit( doc.title , doc._id );
6      }
7  }
```

Wiki

- Index all wiki documents by their title

```
function( doc )
{
    if ( doc.type == "wiki" )
    {
        emit( doc.title , doc._id );
    }
}
```

Wiki

- Index all documents by their title

```
"BuildModuleDesign"          => "wiki-buildmoduledesign"
"CodingGuidelines"           => "wiki-codingguidelines"
"DiscussionProtocols"        => "wiki-discussionprotocols"
"ModuleDesign"               => "wiki-moduledesign"
"Protocol_08_02_07"          => "wiki-protocol_08_02_07"
"VCSModuleDesign"            => "wiki-vcsmoduledesign"
...
```

Wiki

- Index all documents by their title

```
"BuildModuleDesign"          => "wiki-buildmoduledesign"
"CodingGuidelines"           => "wiki-codingguidelines"
"DiscussionProtocols"        => "wiki-discussionprotocols"
"ModuleDesign"               => "wiki-moduledesign"
"Protocol_08_02_07"          => "wiki-protocol_08_02_07"
"VCSModuleDesign"            => "wiki-vcsmoduledesign"
...
```

- Custom deteministic IDs can ensure uniqueness of documents
  - Just set the `_id` property on insert.

Wiki

- Index all documents by their title

```
"BuildModuleDesign"          => "wiki-buildmoduledesign"
"CodingGuidelines"           => "wiki-codingguidelines"
"DiscussionProtocols"        => "wiki-discussionprotocols"
"ModuleDesign"               => "wiki-moduledesign"
"Protocol_08_02_07"          => "wiki-protocol_08_02_07"
"VCSModuleDesign"            => "wiki-vcsmoduledesign"
...
```

- Custom deteministic IDs can ensure uniqueness of documents
  - Just set the `_id` property on insert.
- CouchDB can also generate IDs for you

Map-reduce-views

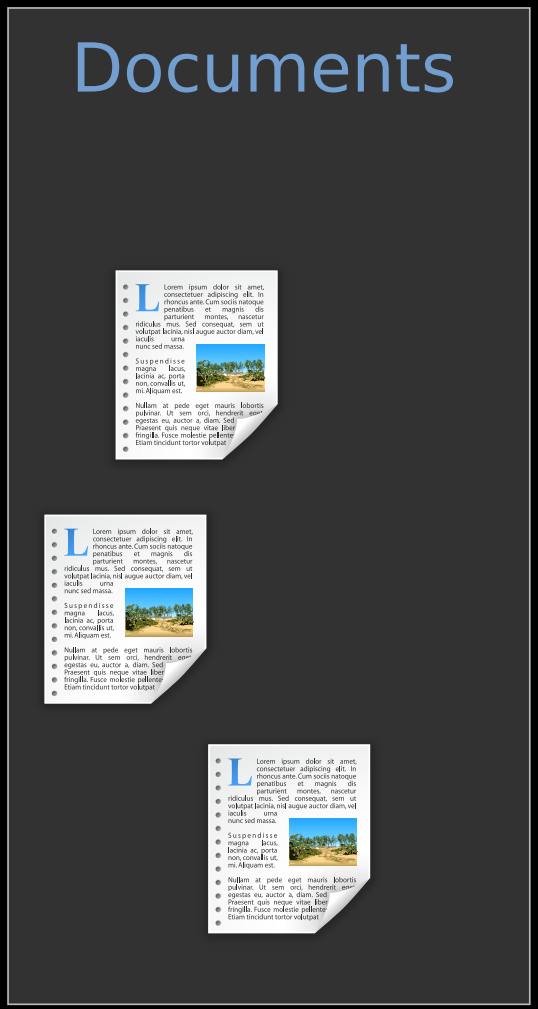
Map-reduce-views

- “MapReduce is a software framework introduced by Google to support distributed computing on large data sets on clusters of computers.” [Wik09]
- Used by CouchDB to implement views

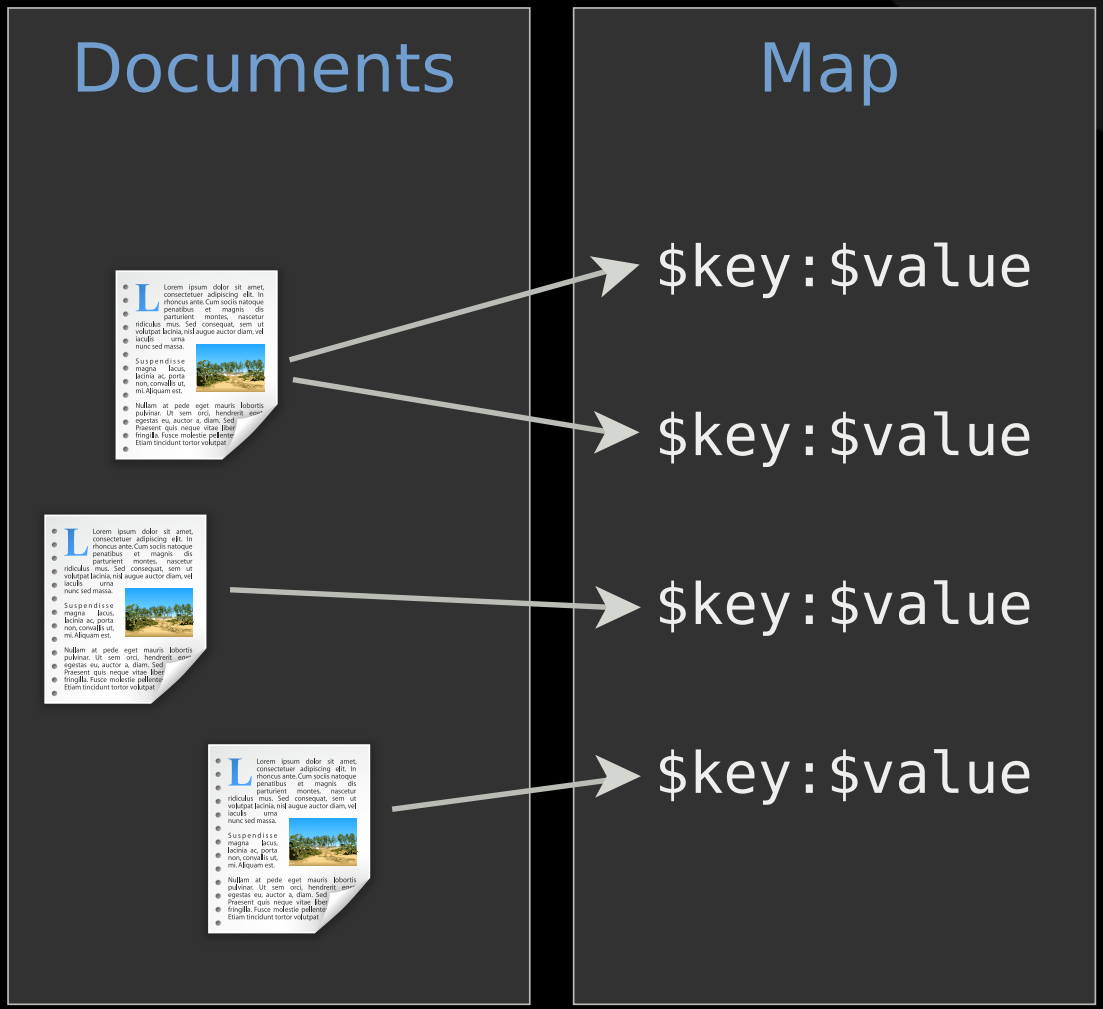
Map-reduce-views

- “MapReduce is a software framework introduced by Google to support distributed computing on large data sets on clusters of computers.” [Wik09]
- Used by CouchDB to implement views
- Just a framework / pattern: You can implement “any” algorithm using map-reduce.

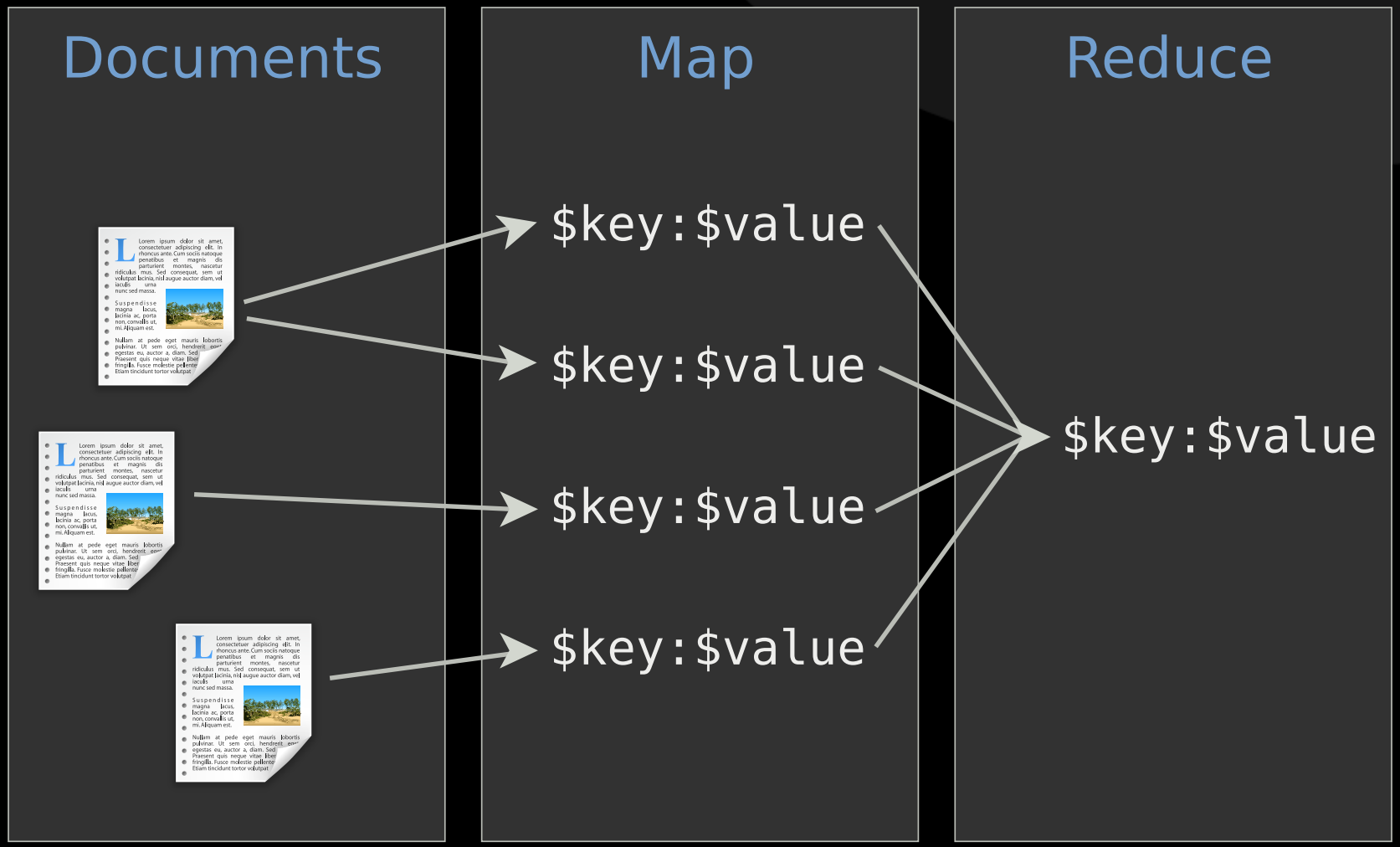
Map-Reduce



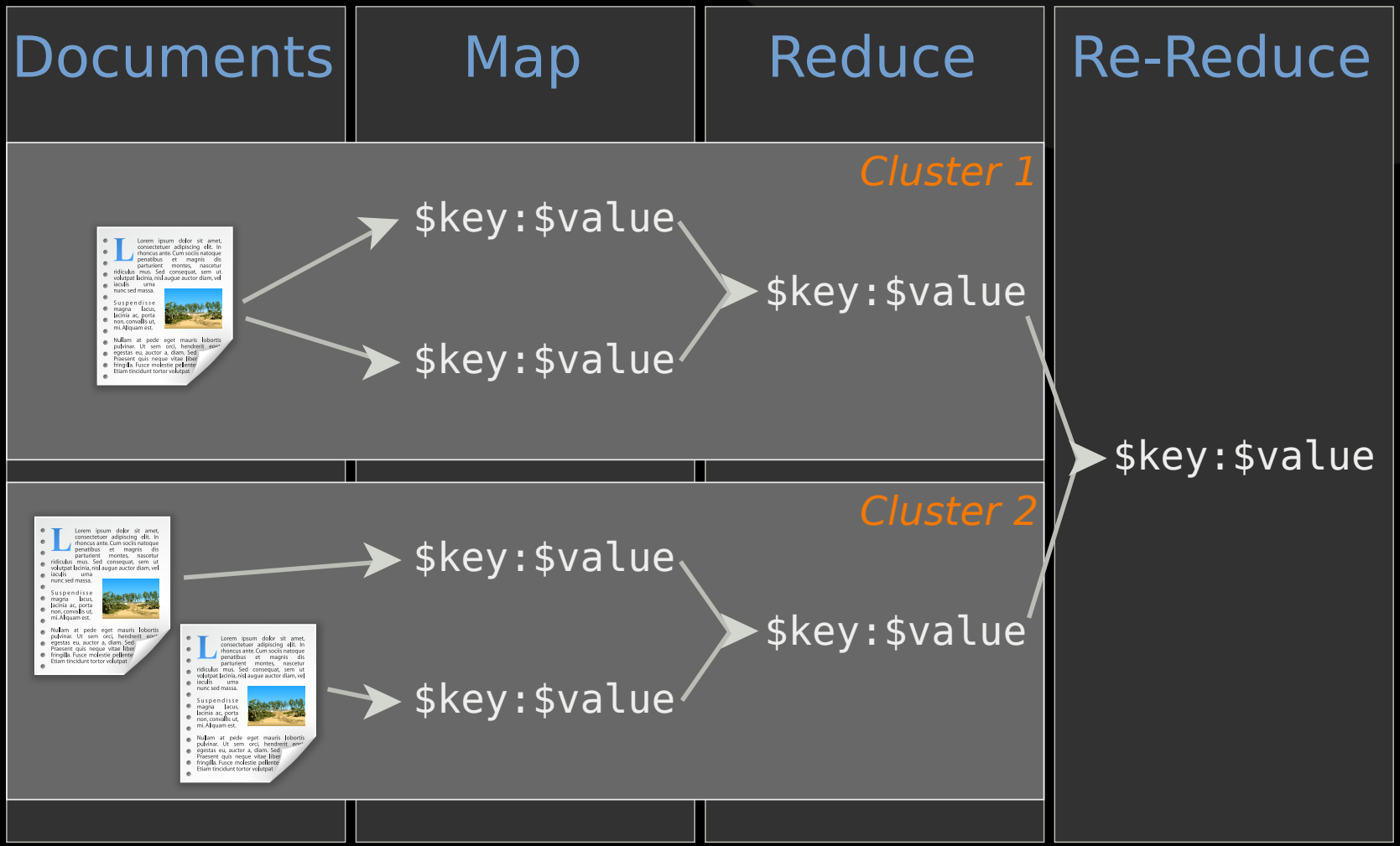
Map-Reduce



Map-Reduce


Documents
Map
Reduce
$key:$value
$key:$value
$key:$value
$key:$value
$key:$value

Map-Reduce


Documents
Map
Reduce
Re-Reduce
Cluster 1
$key:$value
$key:$value
$key:$value
Cluster 2
$key:$value
$key:$value
$key:$value
$key:$value

Map-Reduce

- Map and reduce functions are custom

Map-Reduce

- Map and reduce functions are custom
- Reduce is optional, plain view serves as a document index

Map-Reduce

- Map and reduce functions are custom
- Reduce is optional, plain view serves as a document index
- Reduce may be applied to subsets of the documents

Map-Reduce

- Map and reduce functions are custom
- Reduce is optional, plain view serves as a document index
- Reduce may be applied to subsets of the documents
- Reduce may be grouped

Map-Reduce example

- The simplest reduce function is just `count()`
  - Often used for statistics

```
1 function( keys, values, combine )
2 {
3     return sum( values );
4 }
```

Map-Reduce example

- The reduce result

```
1  null                                    => 42
```

More...

- What should be covered in more depth?
  - Map-reduce views in CouchDB
  - Scalability / consistency in CouchDB
  - The PHPillow API

Map-Reduce example

- The map function

```
function( doc )
{
    if ( doc.type == "wiki" )
    {
        date = new Date();
        date.setTime( doc.edited * 1000 );
        emit( [
                date.getUTCFullYear(),
                date.getUTCMonth() + 1,
                date.getUTCDate(),
                date.getUTCHours(),
                date.getUTCMinutes(),
                date.getUTCSeconds(),
            ], 1 );
        // You could also emit the whole doc as value
    }
}
```

Map-Reduce example

- The mapping result

```
[2008, 10, 11, 9,  11, 12] => 1
[2008, 10, 11, 9,  11, 12] => 1
[2008, 10, 11, 9,  11, 12] => 1
[2008, 10, 11, 9,  13, 8]  => 1
[2008, 10, 11, 9,  13, 44] => 1
[2008, 10, 11, 9,  14, 2]  => 1
[2008, 10, 12, 17, 46, 15] => 1
[2008, 10, 12, 17, 57, 52] => 1
[2008, 10, 12, 18, 0,  45] => 1
[2008, 10, 14, 8,  36, 29] => 1
[2008, 10, 14, 19, 33, 21] => 1
[2008, 10, 14, 19, 33, 35] => 1
```

Map-Reduce example

- The reduce function

```
1  function( keys, values, combine )
2  {
3      return sum( values );
4  }
```

Map-Reduce example

- The reduce result

```
1  null                                    => 12
```

Map-Reduce example

- The grouped reduce result

```
[2008, 10, 11, 9,  11, 12] => 3
[2008, 10, 11, 9,  13, 8]  => 1
[2008, 10, 11, 9,  13, 44] => 1
[2008, 10, 11, 9,  14, 2]  => 1
[2008, 10, 12, 17, 46, 15] => 1
[2008, 10, 12, 17, 57, 52] => 1
[2008, 10, 12, 18, 0,  45] => 1
[2008, 10, 14, 8,  36, 29] => 1
[2008, 10, 14, 19, 33, 21] => 1
[2008, 10, 14, 19, 33, 35] => 1
```

Map-Reduce example

- The filtered grouped reduce result
- `startkey=[2008,10,11]` and `endkey=[2008,10,12]`

```
1 [2008, 10, 11, 9,  11, 12] => 3
2 [2008, 10, 11, 9,  13, 8]  => 1
3 [2008, 10, 11, 9,  13, 44] => 1
4 [2008, 10, 11, 9,  14, 2]  => 1
```

Map-Reduce example

- The grouped reduce result, with group level
- `group-level=3`

```
1 [2008, 10, 11]          => 6
2 [2008, 10, 12]          => 3
3 [2008, 10, 14]          => 3
```

Full-Text-Search

- Index all documents by all their words

```
1  function( doc ) {
2    if ( doc.type == "wiki" ) {
3      // Simple word indexing, does not respect overall
           occurences of words,
4      // stopwords, different word seperation characters,
           or word variations.
5      var text  = doc.title.replace( /[\s:.,!?-]+/g, "␣"
           ) +
6                  doc.text.replace( /[\s:.,!?-]+/g, "␣" )
                       ;
7      var words = text.split( "␣" );
8      for ( var i = 0; i < words.length; ++i ) {
9        value = {};
10       value[doc._id] = 1;
11       emit( words[i].toLowerCase(), value );
12     }
13   }
14 }
```

Wiki

- Index all documents by all their words

```
...
"a"     => {wiki-8: 1}
"a"     => {wiki-8: 1}
"a"     => {wiki-8: 1}
"a"     => {wiki-8: 1}
"a"     => {wiki-81: 1}
"a"     => {wiki-83: 1}
"a"     => {wiki-83: 1}
"able"  => {wiki-39: 1}
"able"  => {wiki-56: 1}
"able"  => {wiki-73: 1}
"able"  => {wiki-80: 1}
"about" => {wiki-24: 1}
"about" => {wiki-43: 1}
"about" => {wiki-85: 1}
...
```

Full-Text-Search

- Reduce by word count

```
1  function( keys, values ) {
2    var count = {};
3    for ( var i in values ) {
4      for ( var id in values[i] ) {
5        if ( count[id] ) {
6          count[id] = values[i][id] + count[id];
7        } else {
8          count[id] = values[i][id];
9        }
10     }
11   }
12   return count;
13 }
```

Wiki

- Index all documents by all their words

```
...
"a"       => {
               wiki-68: 6,
               wiki-66: 6,
               wiki-22: 4,
               wiki-63: 3,
               wiki-60: 2,
               wiki-35: 2,
               wiki-34: 1,
               wiki-31: 1,
               ...
             }
"able"    => {wiki-86: 1, wiki-80: 1, wiki-73: 1, wiki
    -56: 1, wiki-39: 1}
"about"   => {wiki-85: 1, wiki-43: 1, wiki-24: 1}
...
```

Outline

Introduction

General

Structure

API

Views

Consistency

PHPillow

Applications

QA

Local conflicts

- Multi-Version Concurrency Control

Local conflicts

- Multi-Version Concurrency Control
- All documents in the database are versioned

Local conflicts

- Multi-Version Concurrency Control
- All documents in the database are versioned
  - Don't use it for application level document versioning

Local conflicts

- Multi-Version Concurrency Control
- All documents in the database are versioned
  - Don't use it for application level document versioning
- Updates and deletes need to specify the revision ID

Local conflicts

- Multi-Version Concurrency Control
- All documents in the database are versioned
  - Don't use it for application level document versioning
- Updates and deletes need to specify the revision ID
- Changing outdated documents result in conflicts

Inter document links

- There is no ensured inter document consistency in CouchDB

ents:

Inter document links

- There is no ensured inter document consistency in CouchDB
- Different possibilities of relating documents:

Inter document links

- There is no ensured inter document consistency in CouchDB
- Different possibilities of relating documents:
  - List IDs of related documents in document (n:m)

Inter document links

- There is no ensured inter document consistency in CouchDB
- Different possibilities of relating documents:
  - List IDs of related documents in document (n:m)
  - ... both directions are feasible

Inter document links

- There is no ensured inter document consistency in CouchDB
- Different possibilities of relating documents:
  - List IDs of related documents in document (n:m)
  - ... both directions are feasible
  - Embed the whole related document (1:n)

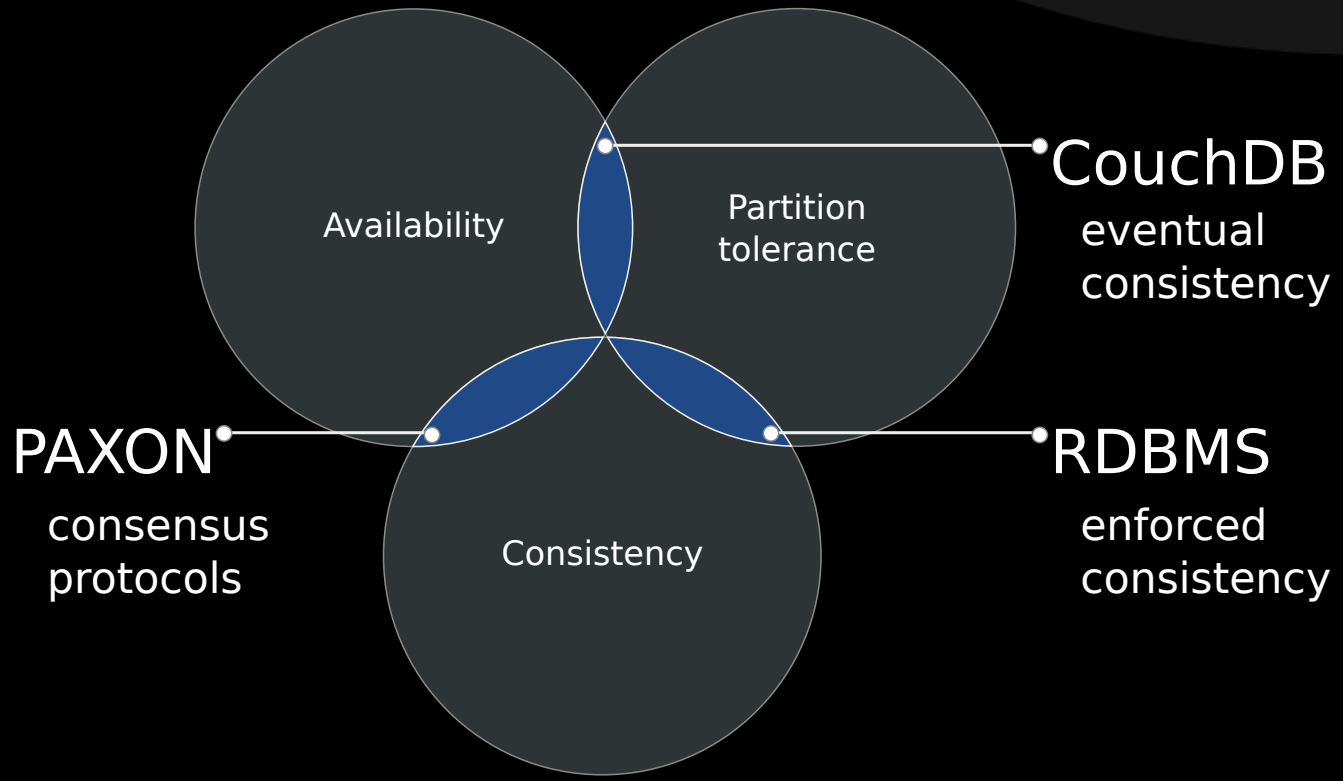
Inter document links

- There is no ensured inter document consistency in CouchDB
- Different possibilities of relating documents:
  - List IDs of related documents in document (n:m)
  - ... both directions are feasible
  - Embed the whole related document (1:n)
- Solution depends on update-ratio

```
{ "type":  "wiki",
  "title": "Hello world",
  "text":  "...",
  "comments": [
    { "comment": "..." },
  ],
  "creator": "user-foo",
}
```

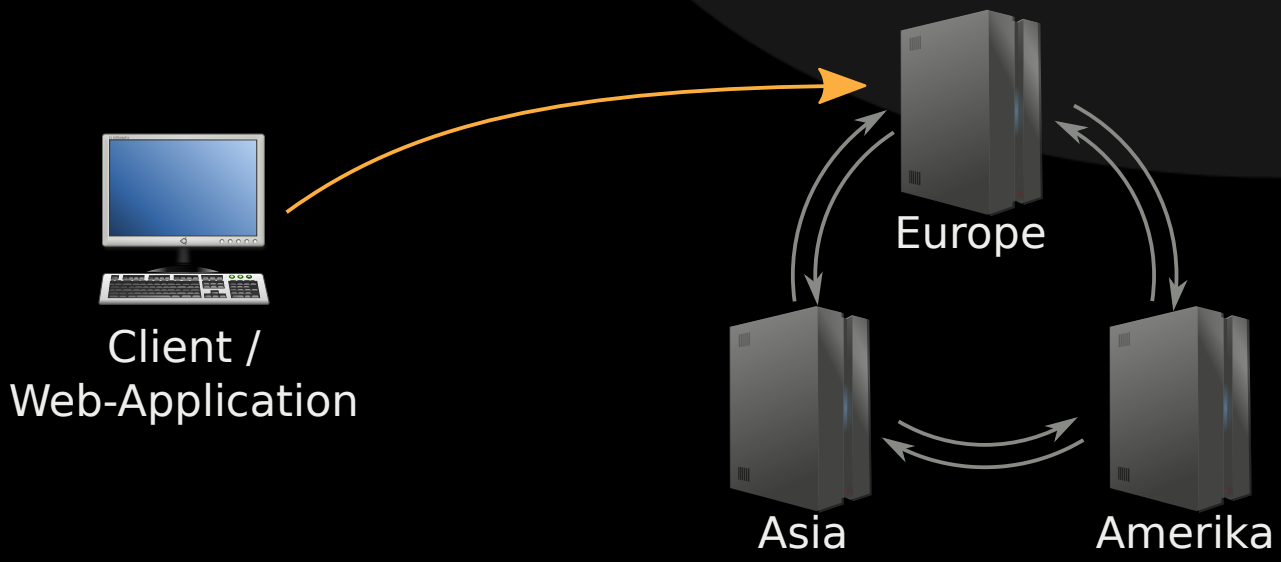
Scaling: The CAP theorem

- The CAP theorem, read more in “CouchDB: The Definitive Guide” [JCA09]



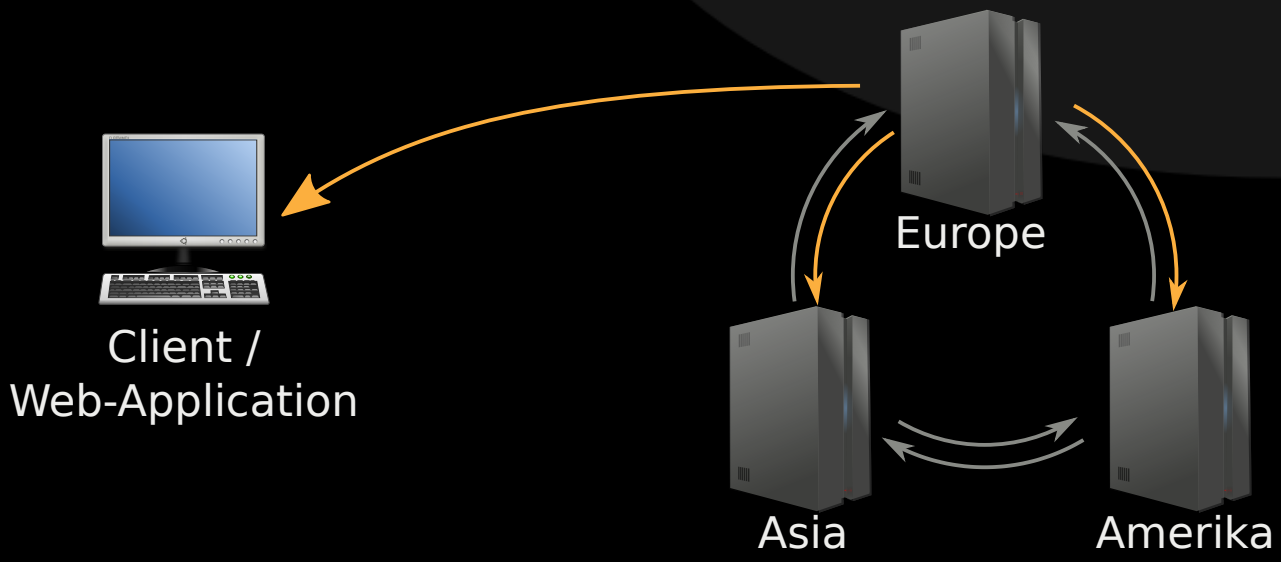
- CouchDB employs “Eventual Consistency” [Vog09]

Eventual consistency



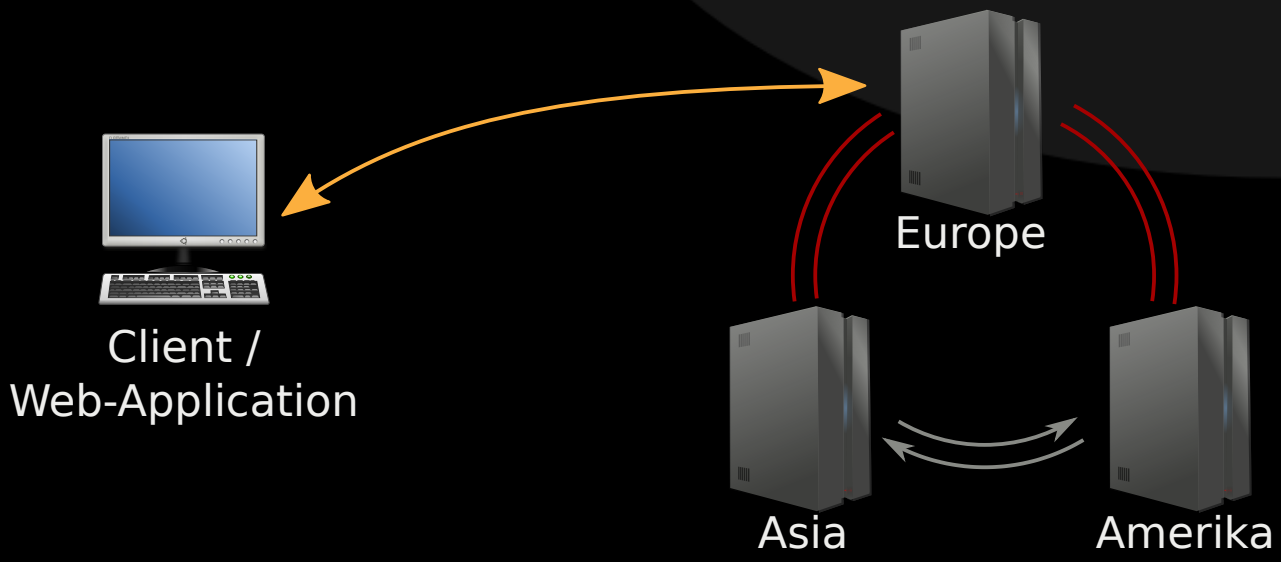
- Structure your

Eventual consistency



- Delayed, triggered synchronization (push, pull)
  - Deterministic (manual) conflict resolution on replication on all nodes
  - Structure your

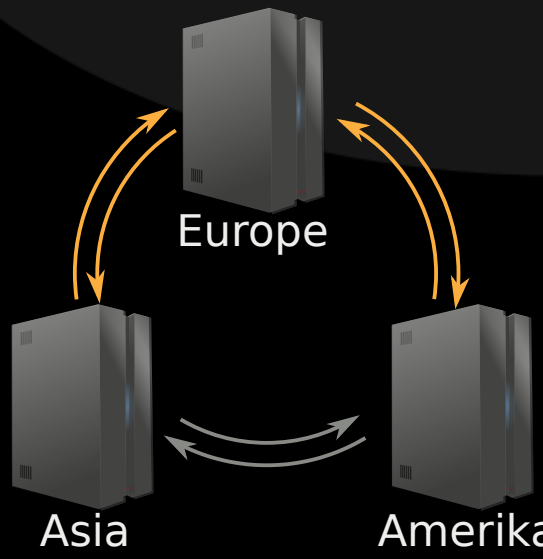
Eventual consistency



- Delayed, triggered synchronization (push, pull)
  - Deterministic (manual) conflict resolution on replication on all nodes
  - Structure your

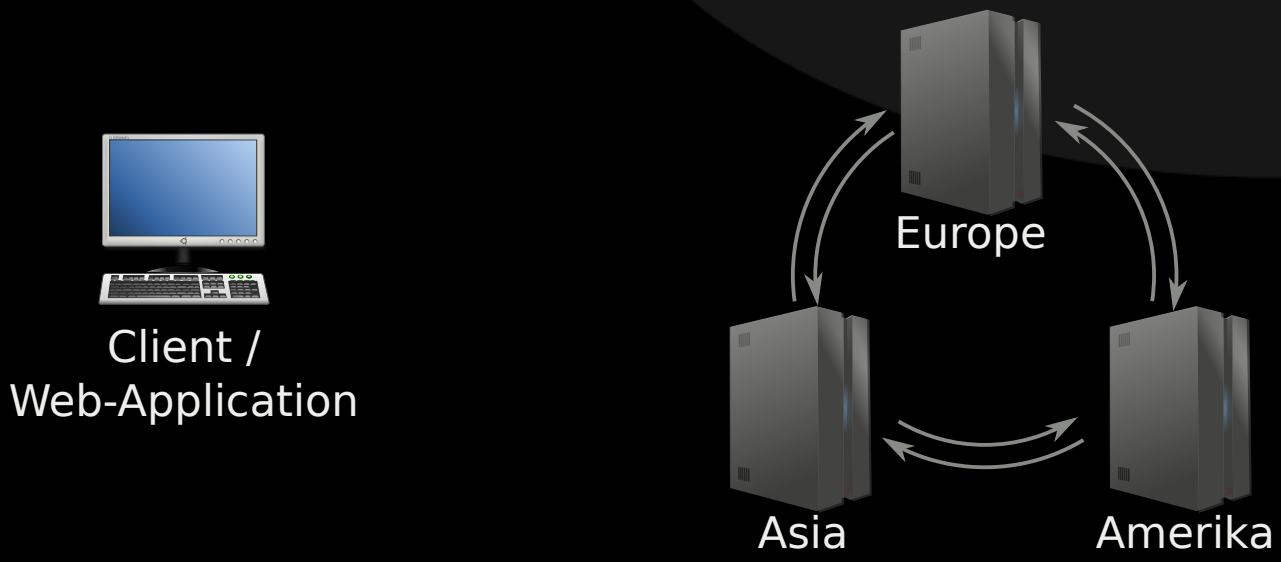
Eventual consistency

Client /
Web-Application



- Delayed, triggered synchronization (push, pull)
  - Deterministic (manual) conflict resolution on replication on all nodes
- Scales well for seldom concurrent writes
  - Structure your

Eventual consistency



- Delayed, triggered synchronization (push, pull)
  - Deterministic (manual) conflict resolution on replication on all nodes
- Scales well for seldom concurrent writes
  - Structure your documents accordingly

Outline

Introduction

General

Structure

API

Views

Consistency

PHPillow

Applications

QA

PHPillow

- Object-oriented client for CouchDB
- PHP >= 5.2 since last release (5.3 only before)
- >96% test coverage

PHPillow

- Object-oriented client for CouchDB
- PHP >= 5.2 since last release (5.3 only before)
- >96% test coverage
- Still in alpha state

PHPillow

- Object-oriented client for CouchDB
- PHP >= 5.2 since last release (5.3 only before)
- >96% test coverage
- Still in alpha state
  - Since CouchDB just got “beta” recently, and no new release was required.

PHPillow

- Lightweight layer

PHPillow

- Lightweight layer
- Features
  - Simple document validation constraints

PHPillow

- Lightweight layer
- Features
  - Simple document validation constraints
  - Automatic synchronization of views

PHPillow

- Lightweight layer
- Features
  - Simple document validation constraints
  - Automatic synchronization of views
  - Automatic versioning of documents

PHPillow

- Lightweight layer
- Features
  - Simple document validation constraints
  - Automatic synchronization of views
  - Automatic versioning of documents
  - couchdb-python compatible tool for dump and import

PHPillow

- Lightweight layer
- Features
  - Simple document validation constraints
  - Automatic synchronization of views
  - Automatic versioning of documents
  - couchdb-python compatible tool for dump and import
- Different connection handlers
  - PHP HTTP stream wrapper
  - Custom HTTP protocol implementation

PHPillow

- Lightweight layer
- Features
  - Simple document validation constraints
  - Automatic synchronization of views
  - Automatic versioning of documents
  - couchdb-python compatible tool for dump and import
- Different connection handlers
  - PHP HTTP stream wrapper
  - Custom HTTP protocol implementation
    - Which is faster, most likely because of `Connection: Keep-Alive`

A document class

- A custom document class

```php
<?php
class myUserDocument extends phpillowDocument {
    protected $versioned = true;

    // ...
}
```

A document class

- A custom document class

```php
<?php
class myUserDocument extends phpillowDocument {
    protected $versioned = true;

    protected $requiredProperties = array(
        'login',
    );

    // ...
}
```

A document class

- A custom document class

```
1  <?php
2  class myUserDocument extends phpillowDocument {
3      // ...
4      public function __construct() {
5          $this->properties = array(
6              'login'    => new new
                   phpillowRegexpValidator( '(^[\x21-\x7e
                   ]+$)' ),
7              'name'     => new phpillowStringValidator(),
8              'friends' => new
                   phpillowDocumentArrayValidator( '
                   myUserDocument' ),
9              ...
10         );
11         parent::__construct();
12     }
13     // ...
14 }
```

A document class

- A custom document class

```php
<?php
class myUserDocument extends phpillowDocument {
    // ...
    protected function generateId() {
        // Return null for auto-generated IDs
        return $this->stringToId( $this->storage->login
            );
    }
    // ...
}
```

A document class

- A custom document class

```php
<?php
class myUserDocument extends phpillowDocument {
    // ...
    protected function getType() {
        return 'user';
    }
}
```

Document examples

- Document creation example

```php
// Create a document
$doc = new myUserDocument();
$doc->login = 'kore';
$doc->name  = 'Kore Nordmann';
$doc->data  = array(
    'mail' => "kore@php.net",
    // ...
);

try {
    $id = $doc->save();
} ( phpillowResponseConflictErrorException $e ) {
    // Document already exists
}

// Fetch a document by ID
$doc = new myUserDocument( $id );
```

View examples

- View class

```
1  class myUserView extends phpillowFileView {
2      public function __construct() {
3          parent::__construct();
4
5          $this->viewFunctions = array(
6              'all' => array(
7                  'map'    => 'map/user_all.js',
8              ),
9              'user' => array(
10                 'map'    => 'map/user_user.js',
11                 'reduce' => 'reduce/sum.js',
12             ),
13         );
14     }
15
16     // ...
17 }
```

View examples

- View class

```
1  class myUserView extends phpillowFileView {
2      // ...
3
4      protected function getViewName() {
5          return 'users';
6      }
7  }
```

View examples

- Query a view

```
1  $users = myUserView::all( array(
2          'key' => 'Kore Nordmann',
3  ) );
```

- PHPillow validates and converts allowed view parameters

View examples

- Query a view

```
1  $users = myUserView::all( array(
2          'key' => 'Kore Nordmann',
3  ) );
```

- PHPillow validates and converts allowed view parameters
- What happens:
  - View function will be uploaded to the database

View examples

- Query a view

```
1  $users = myUserView::all( array(
2          'key' => 'Kore Nordmann',
3  ) );
```

- PHPillow validates and converts allowed view parameters
- What happens:
  - View function will be uploaded to the database
  - CouchDB will index all documents in the database using the view function

View examples

- Query a view

```
1  $users = myUserView::all( array(
2          'key' => 'Kore Nordmann',
3  ) );
```

- PHPillow validates and converts allowed view parameters
- What happens:
  - View function will be uploaded to the database
  - CouchDB will index all documents in the database using the view function
  - View results will be returned as an array

Outline

Introduction

General

Structure

API

Views

Consistency

PHPillow

Applications

QA

Attachments

- CouchDB allows you to attach files to documents

Attachments

- CouchDB allows you to attach files to documents
- Files are replicated

Attachments

- CouchDB allows you to attach files to documents
- Files are replicated
- You can serve full Web-Applications from a CouchDB
  - See CouchApp
- Deploy using PUSH-replication

Eventual consistency

- Mirror database into userspace

Eventual consistency

- Mirror database into userspace
- Offline usage and synchronization of Browser applications

Eventual consistency

- Mirror database into userspace
- Offline usage and synchronization of Browser applications
- Mozilla develops a JavaScript implementation of the CouchDB API [Moz09]

Ubuntu One

- Ubuntu One uses CouchDB
- Synchronize contacts & date between nodes, or to a server

Ubuntu One

- Ubuntu One uses CouchDB
- Synchronize contacts & date between nodes, or to a server
- Yes, all Ubuntu Karmic users already have a CouchDB running

Various applications

- Arbit uses CouchDB for issue tracking, wiki, FAQ and more
- Other applications: `http://wiki.apache.org/couchdb/CouchDB_in_the_wild`

Summary

- CouchDB is fast (enough)
- Document oriented approach allows new application development approaches
- CouchDB scales really well, horizontaly and verticaly
- CouchDB fits web applications really well
  - RDBMS are still better for single-cluster scalable applications with strong integrity requirements.

Outline

Introduction

General

Structure

API

Views

Consistency

PHPillow

Applications

QA

Resources

- Apache CouchDB: `http://couchdb.org/`
- Free CouchDB book: `http://books.couchdb.org/relax/`
- PHPillow: `http://arbitracker.org/phpillow.html`

The end

- Open questions?
- Further remarks?
- Contact
  - Mail: <kore@php.net>
  - Web: `http://kore-nordmann.de/` (Slides will be available here soonish)
  - Twitter: `http://twitter.com/koredn`
  - Comment: `http://joind.in/1465`

Bibliography I

[JCA09] Noah Slater J. Chris Anderson, Jan Lehnardt, *Couchdb: The definitive guide*, O'Reilly Media, Inc., 2009.

[Moz09] Mozilla, *Browsercouch documentation*, November 2009.

[Vog09] Werner Vogels, *Eventually consistent - revisited*, `http://www.allthingsdistributed.com/2008/12/eventually_consistent.html`, December 2009.

[Wik09] Wikipedia, *Mapreduce — wikipedia, the free encyclopedia*, 2009, [Online; accessed 27-August-2009].