CouchDB, PHP & PHPillow

http://joind.in/1445

Kore Nordmann
<kore@php.net>
@koredn

May 15, 2010

About me

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

Outline

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

CouchDB is paradigmn shift

- Structure
- Consistency
- API
- Applications
  - I will show this by providing some views for a “blog”

Outline


CouchDB
relax

- Apache top-level project

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

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

Outline

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

- We need to create a database first:

```php
<?php

$fp = fopen(
    'http://localhost:5984/phpday_blog', 'r', false,
    stream_context_create( array(
        'http' => array(
            'method'        => 'PUT',
            'content'       => null,
            'ignore_errors' => true,
            'header'        => 'Content-type: ␣
                application/json',
) ) ) );

var_dump( json_decode( stream_get_contents( $fp ), true
    ) );
/* array(1) {
  ["ok"] => bool(true)
} */
```

- Or do the same with curl:

```
1  $ curl -i -X PUT http://localhost:5984/phpday_blog
2
3  HTTP/1.1 412 Precondition Failed
4  Server: CouchDB/0.10.0 (Erlang OTP/R13B)
5  Date: Sun, 09 May 2010 10:11:09 GMT
6  Content-Type: text/plain;charset=utf-8
7  Content-Length: 95
8  Cache-Control: must-revalidate
9
10 {"error":"file_exists","reason":"The database could not
       be created, the file already exists."}
```

- Store a document

```php
<?php

$fp = fopen(
    'http://localhost:5984/phpday_blog/hello_world', 'r', false,
    stream_context_create( array(
        'http' => array(
            'method'        => 'PUT',
            'content'       => json_encode( array(
                'type'   => 'blog_entry',
                'title'  => 'Hello world!',
                'text'   => 'Some text ...',
                'author' => 'kore',
            ) ),
            'ignore_errors' => true,
            'header'        => 'Content-type: application/json',
) ) ) );

var_dump( json_decode( stream_get_contents( $fp ), true ) );
```

Structure

- Store a document

```
1  <?php
2
3  /* array(6) {
4     ["_id"] =>     string(11) "hello_world"
5     ["_rev"] =>    string(34) "1-0
          a73fbe05e737f0519ab2792d71d4911"
6     ["type"] =>    string(10) "blog_entry"
7     ["title"] =>   string(12) "Hello world!"
8     ["text"] =>    string(12) "Some ...text"
9     ["author"] => string(4) "kore"
10 } */
```

Structure

- Custom deteministic IDs can ensure uniqueness of documents
  - Just define the ID in the URL

Structure

- Custom deteministic IDs can ensure uniqueness of documents
  - Just define the ID in the URL
- CouchDB can also generate IDs for you
  - Which then are just unique hashes

Structure

- Retrieve a document

```php
<?php

var_dump( json_decode( file_get_contents( 'http://
    localhost:5984/phpday_blog/hello_world' ) ) );

/* array(5) {
  ["_id"]     => string(11) "hello_world"
  ["_rev"]    => string(34) "1-0
      f563b09aa9a73b56a906919f013d492"
  ["type"]    => string(10) "blog_entry"
  ["text"]    => string(12) "Hello world!"
  ["author"] => string(4) "kore"
} */
```

Migration / Refactoring

- Yeah, it is that easy.

Migration / Refactoring

- Yeah, it is that easy.
- Change document structure at any time

Migration / Refactoring

- Yeah, it is that easy.
- Change document structure at any time
- No need for (non-transaction-safe) Data Definition Language (DDL)

Migration / Refactoring

- Yeah, it is that easy.
- 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

- Yeah, it is that easy.
- 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

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

View

- Index all blog documents by their title

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

View

- Index all blog documents by their title

```
1  function( doc )
2  {
3      if ( doc.type == "blog_post" )
4      {
5          emit( doc.title , doc._id );
6      }
7  }
```

View

- Upload the view

```php
<?php

$fp = fopen(
    'http://localhost:5984/phpday_blog/_design/blog', 'r', false,
    stream_context_create( array(
        'http' => array(
            'method'        => 'PUT',
            'content'       => json_encode( array(
                'language' => 'javascript',
                'views' => array(
                    'list' => array(
                        'map' => file_get_contents( '005_map.js' ),
            ), ), ) ),
            'ignore_errors' => true,
            'header'        => 'Content-type: application/json',
) ) ) );
```

View

- Fetching data

```php
<?php

var_dump( json_decode( file_get_contents(
    'http://localhost:5984/phpday_blog/_design/blog/
        _view/list'
), true ) );

/* array(3) {
  ["total_rows"] => int(1)
  ["offset"]     => int(0)
  ["rows"]       => array(1) {
    [0] => array(3) {
      ["id"]    => string(11) "hello_world"
      ["key"]   => string(12) "Hello world!"
      ["value"] => string(11) "hello_world"
    }
  }
} */
```

Views

- Comparable to `SELECT * FROM table WHERE column = "value"`

Views

- Comparable to `SELECT * FROM table WHERE column = "value"`
  - Allows range requests

Views

- Comparable to `SELECT * FROM table WHERE column = "value"`
  - Allows range requests
  - Allows limit / offset

Views

- Comparable to `SELECT * FROM table WHERE column = "value"`
  - Allows range requests
  - Allows limit / offset
  - Allows key accross multiple fields

Views

- Comparable to `SELECT * FROM table WHERE column = "value"`
  - Allows range requests
  - Allows limit / offset
  - Allows key accross multiple fields
  - Even allow multiple indexes per document (tags)

Views

- Comparable to `SELECT * FROM table WHERE column = "value"`
  - Allows range requests
  - Allows limit / offset
  - Allows key accross multiple fields
  - Even allow multiple indexes per document (tags)
  - Allow complex keys `[category, title]`

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

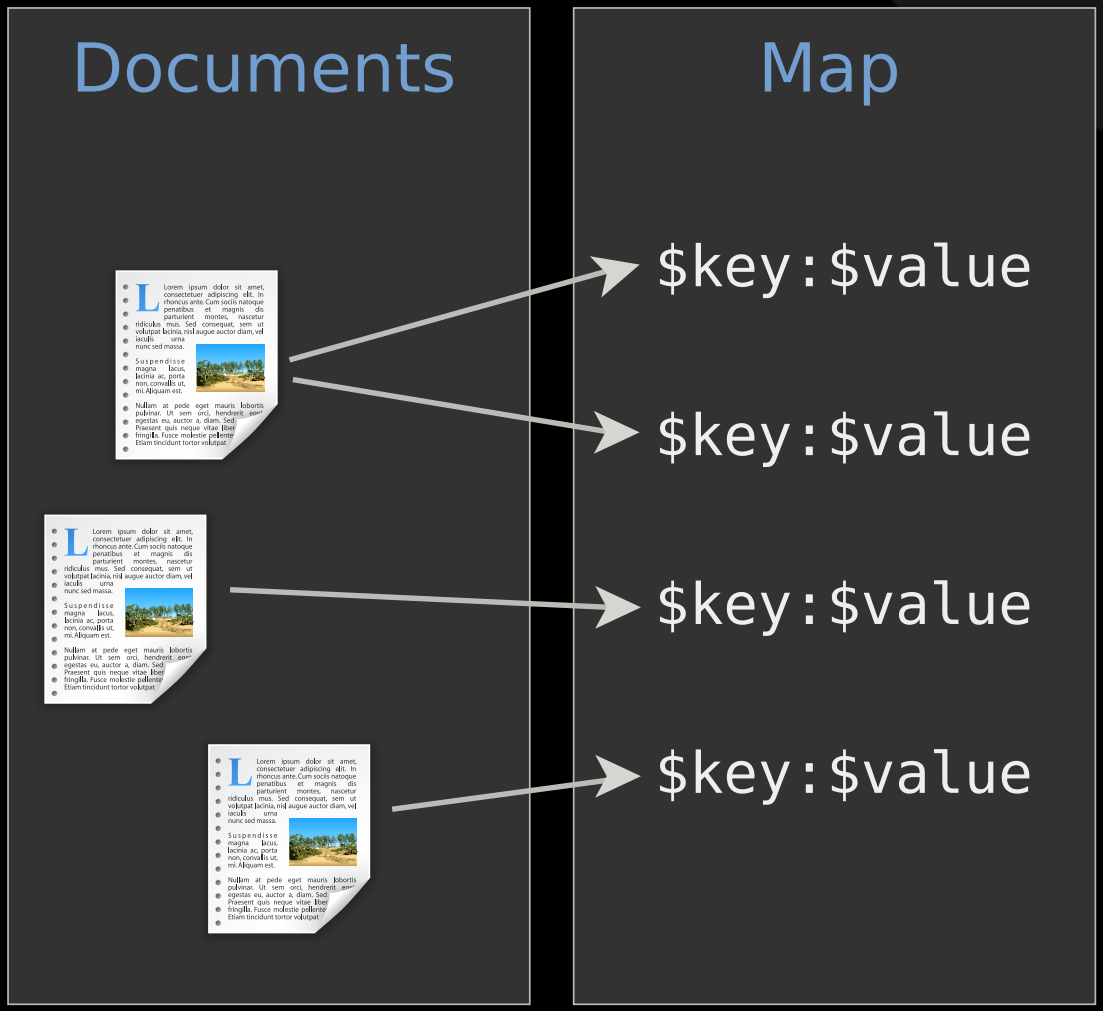
- “MapReduce is a software framework introduced by Google to support distributed computing on large data sets on clusters of computers.” [Wik09]
- Used by CouchDB to implement views
- Just a framework / pattern: You can implement “any” algorithm using map-reduce.

Map-Reduce

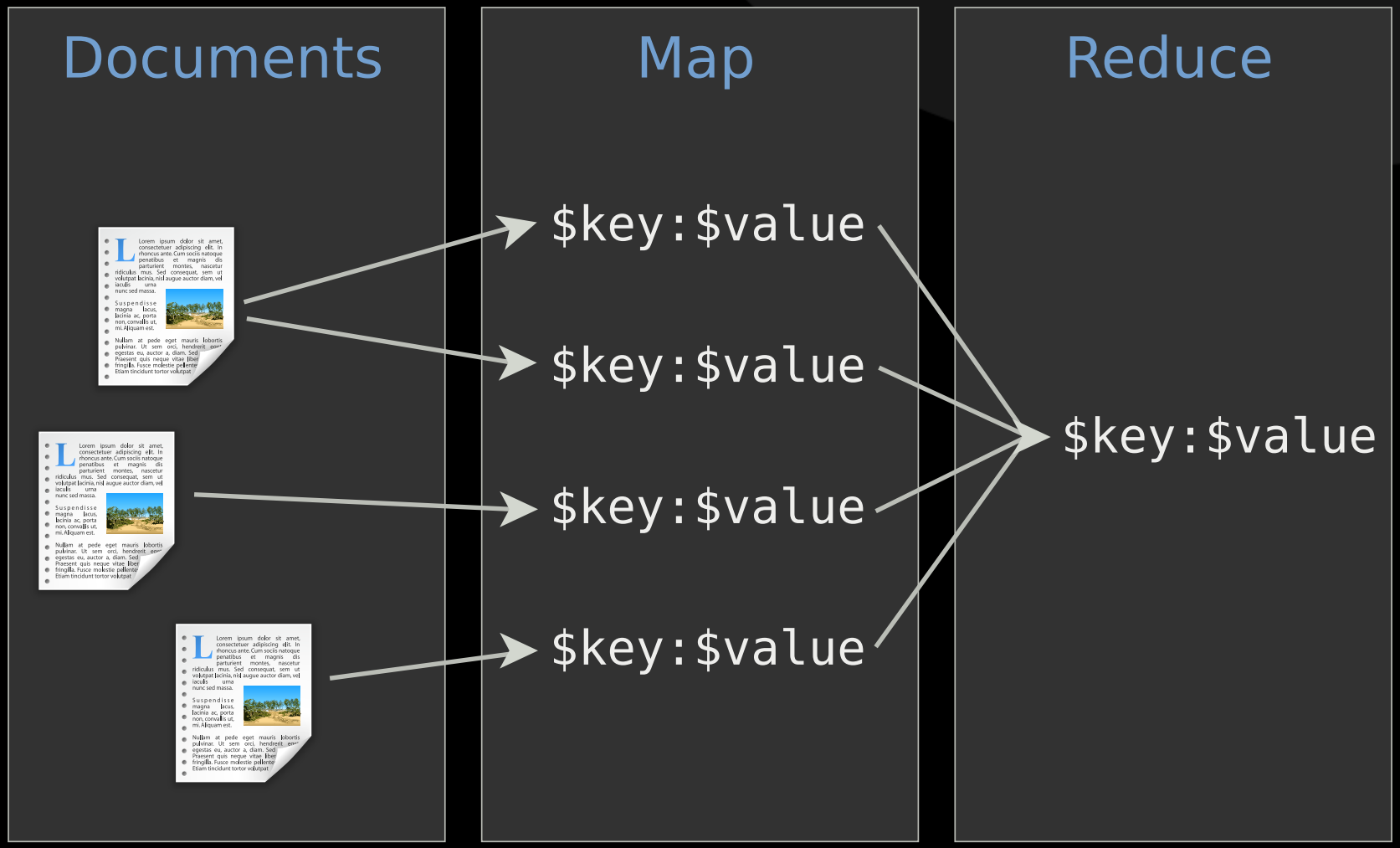
Documents



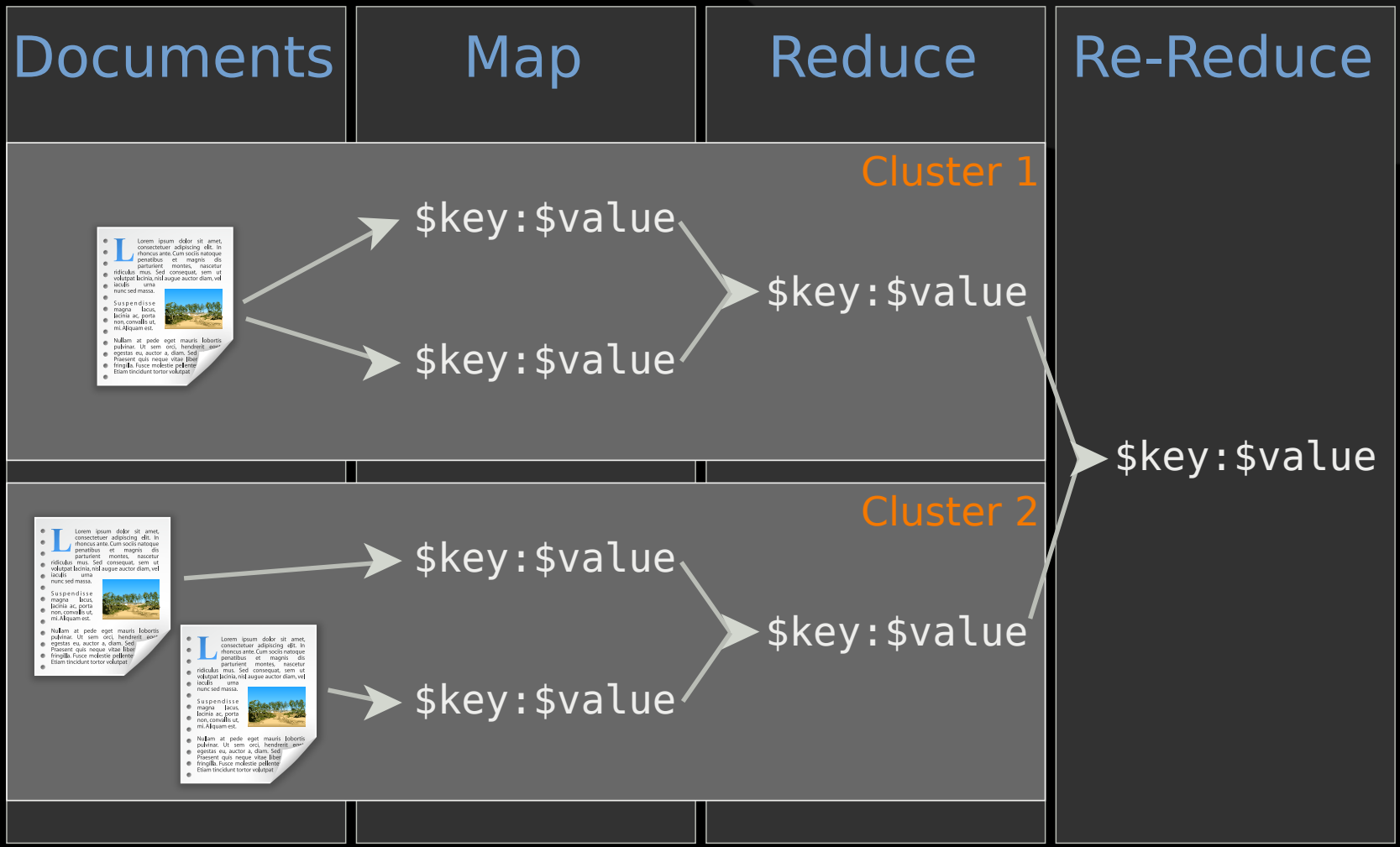
Map-Reduce



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 map function

```
1  function( doc )
2  {
3      if ( doc.type == "blog_post" )
4      {
5          date = new Date();
6          date.setTime( doc.edited * 1000 );
7          emit( [
8                  date.getUTCFullYear(),
9                  date.getUTCMonth() + 1,
10                 date.getUTCDate(),
11                 date.getUTCHours(),
12                 date.getUTCMinutes(),
13                 date.getUTCSeconds(),
14             ], 1 );
15         // You could also emit the whole doc as value
16     }
17 }
```

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 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
```

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
```

Outline

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

JOIN query

```
1  function( doc )
2  {
3      if ( doc.type == "blog" )
4      {
5          emit( [doc._id, 0], doc._id );
6      }
7
8      if ( doc.type == "blog_comment" )
9      {
10         emit( [doc.blog_post, doc._id], doc._id );
11     }
12 }
```

Issue tracker

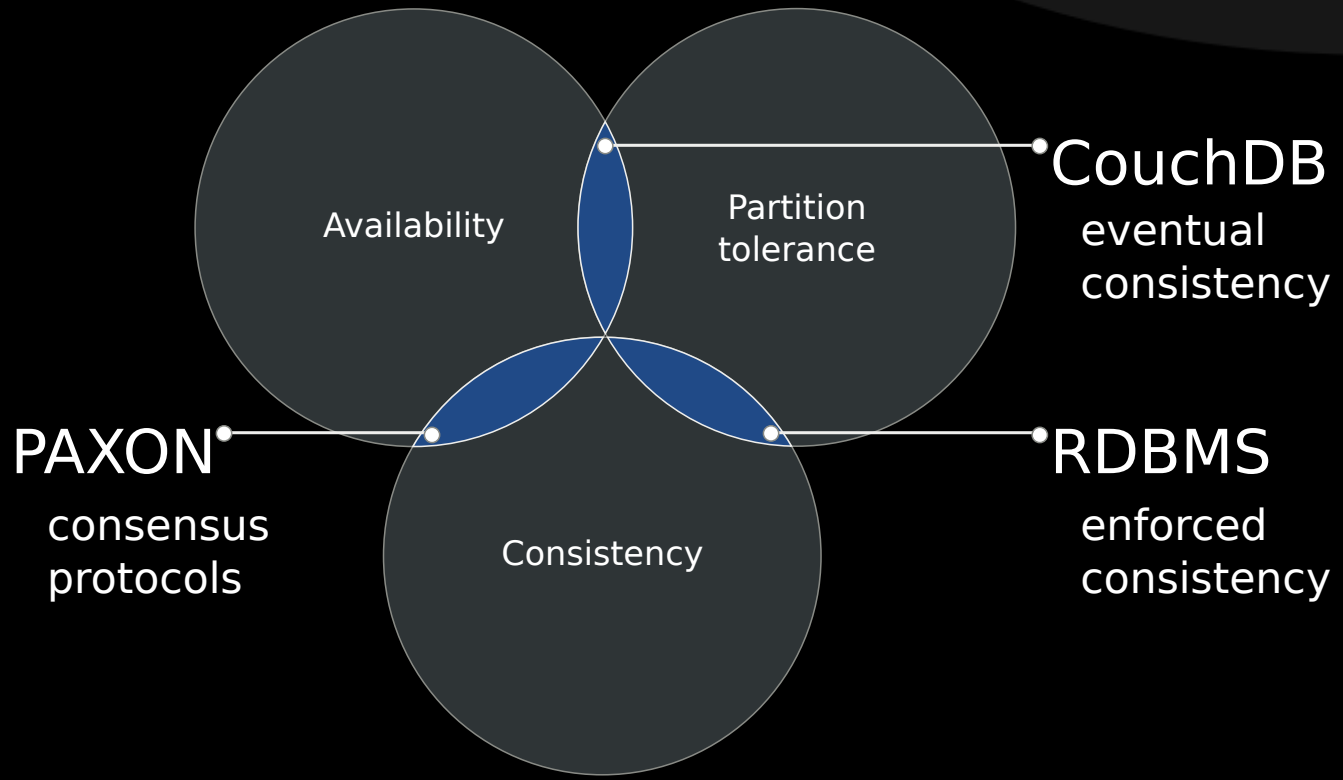
- JOIN query result

```
["blog_post-1",   0] => "blog_post-1"
["blog_post-1",   "eaaa503adbec07"] => "eaaa503adbec07"
["blog_post-1",   "e77ebd5ea0383c"] => "e77ebd5ea0383c"
["blog_post-1",   "0502afb84a80a4"] => "0502afb84a80a4"
["blog_post-10",  0] => "blog_post-10"
["blog_post-10",  "3872cd8a4d530f"] => "3872cd8a4d530f"
["blog_post-10",  "af0dd3c4d184cf"] => "af0dd3c4d184cf"
```

- Can again be filtered...
- Using "`?include_docs=true`" also provides you all documents

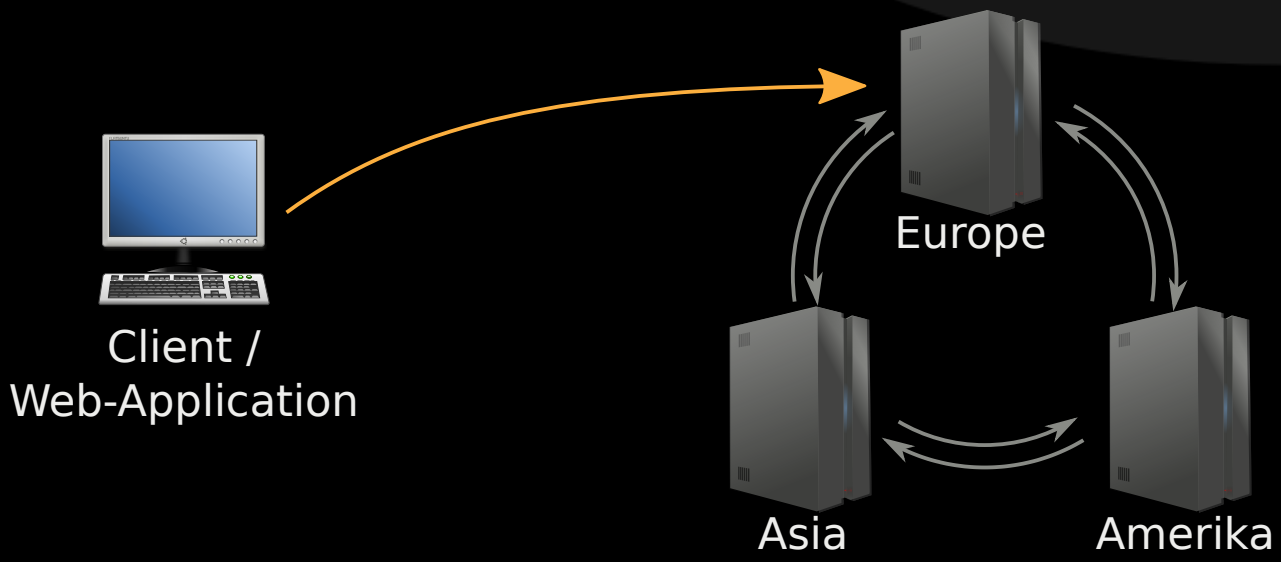
Scaling: The CAP theorem

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



- CouchDB employs “Eventual Consistency” [Vog09]

Eventual consistency



- Important for seperated nodes (sharding)

Outline

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`

View example

- Query a view

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

- PHPillow validates and converts allowed view parameters

View example

- Query a view

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

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

View example

- 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, if the view is new

View example

- 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, if the view is new
  - View results will be returned as an array

Outline

Attachments

- CouchDB allows you to attach files to documents

Attachments

- CouchDB allows you to attach files to documents
- Files are replicated
  - Even incrementally since 0.11

Attachments

- CouchDB allows you to attach files to documents
- Files are replicated
  - Even incrementally since 0.11
- 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

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/1445`

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}
...
```

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].