A FIRST TOUCH ON NOSQL SERVERS: COUCHDB

GENOVEVA VARGAS SOLAR, JAVIER ESPINOSA

CNRS, LIG-LAFMIA, FRANCE

Genoveva.Vargas@imag.fr; Javier.Espinosa@imag.fr

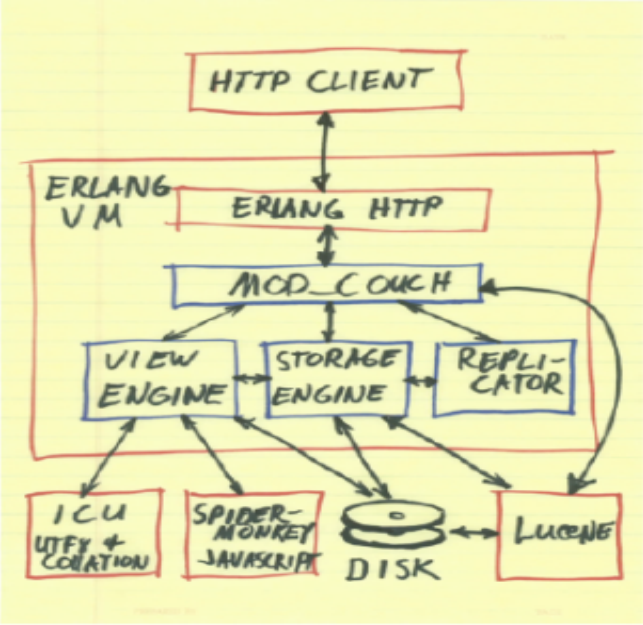
http://www.vargas-solar.com

ARCHITECTURE

HTTP
the access protocol

JavaScript
the query language

JSON
the storage format



DOWNLOADS

- For executing these examples download CouchDB:
  - Binary version for windows
  - http://wiki.apache.org/couchdb/Installing_on_Windows
- For communicating with the server we can use a browser or the tool cURL:
  - cURL is a tool for transfering data from the server using different protocols including HTTP
  - http://curl.haxx.se/download.html

COUCHDB (1)

- CouchDB is a document oriented DBMS, *i.e.*, it is not relational:
  - Without database schema
  - Key-value model
  - Distributed and fault tolerant
- Data are modeled as autocontained documents:
  - A document is represented by a JSON structure with attributes of any type
  - Queries are done via JavaScript

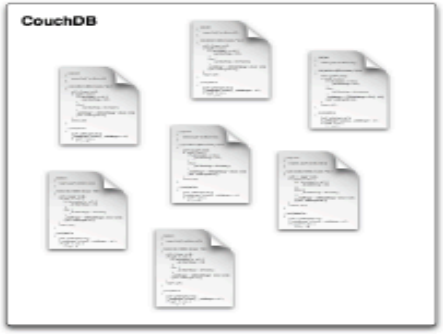
COUCHDB (2)

- Different data types are supported as add-in documets (video, audio, images, etc.)
- Communication with applications and users is done via RESTful services:
  - «*Representational State Transfer*» is a software client-server architecture model used for distributed hypermedia systems
  - The communication protocol HTTP:
    - Used the HTTP methods explicitly
    - Stateless
    - Exposes the structure via URIs
    - Data transfered are XML or JSON (for CouchDB)

INTERACTING WITH COUCHDB

HTTP Client

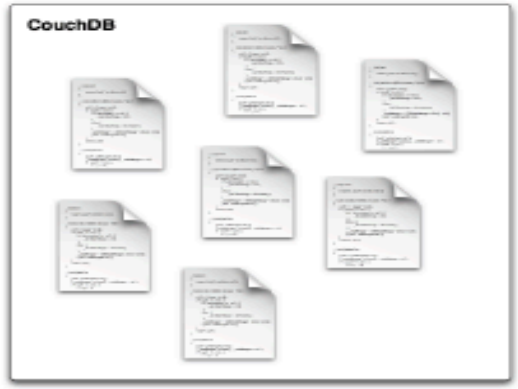
PUT /dbname/ID



INTERACTING WITH COUCHDB

HTTP Client

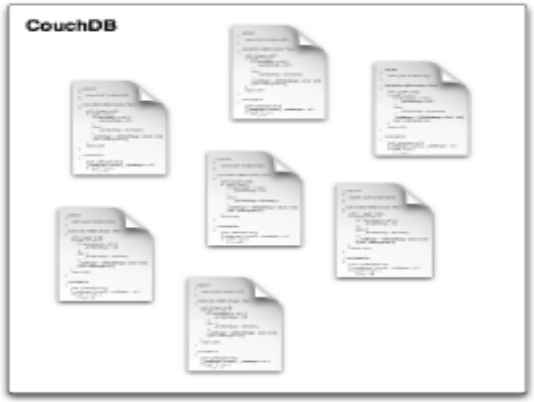
GET /dbname/ID



INTERACTING WITH COUCHDB

HTTP Client

DELETE /dbname/ID



CREATING AND RETRIEVING

- Creating the database "albums":

```
curl -X PUT http://localhost:5984/albums
```

- Creating the document "album1":

```
curl -X PUT http://localhost:5984/albums/album1 -d @-
{
     "artista": "Megadeth",
     "titulo": "Endgame",
     "anio": 2009
}
<EOF>    // en Windows es ^z y en Unix ^d
```

- Retrieving the created document:

```
curl -X GET http://localhost:5984/albums/album1
```

UPDATING (1)

- For updating a document:
  - Give the last version
  - Otherwise an error (code 409) will be generated, as shown in the following example:

```
curl -X PUT http://localhost:5984/albums/album1 -d @-
{
  "artista": "Megadeth",
  "titulo": "Endgame",
  "anio": 2010
}
^z
```

UPDATING (2)

- The attribute "_rev" specifies the version that will be updated:

```
curl -X PUT http://localhost:5984/albums/album1 -d @-
{
    "_rev": "1-142438dc8c583cda2a1f292c62291215",
    "artista": "Megadeth",
    "titulo": "Endgame",
    "anio": 2010
}
^z
```

DELETING (1)

- Delete the document "album1":
  - `curl -X DELETE http://localhost:5984/albums/album1?rev=2-d05127b44500ec19a2e5a25adc610380`
- If you try to retrieve it, an error is generated:
  - `curl -X GET http://localhost:5984/albums/album1`
  - `{"error":"not_found","reason":"deleted"}`
- You have access to the version generated by the deletion operation:
  - `curl -X GET http://localhost:5984/albums/album1?rev=3-fac16c94309ed5ff842ffa89cc6048b1`
  - `{"_id":"album1","_rev":"3-fac16c94309ed5ff842ffa89cc6048b1","_deleted":true}`

DELETING (2)

- We purge the document from the database:

```
curl -X POST -H "Content-Type: application/json"  http://localhost:5984/albums/_purge
-d @-
{
   "album1": ["3-fac16c94309ed5ff842ffa89cc6048b1"]
}
```

- We try to query the version again:
  - `curl -X GET http://localhost:5984/albums/album1?rev=3-fac16c94309ed5ff842ffa89cc6048b1`
  - `{"error":"not_found","reason":"missing"}`

ATTACHMENTS (1)

- Any binary type can be stored by adding it to a document
- Let us create again "album1":

```
curl -X PUT http://localhost:5984/albums/album1 -d @-
{
   "artista": "Megadeth", "titulo": "Endgame", "anio": 2010
}
```

- The method HTTP PUT is used for attaching a file to the document using the attribute "cover.jpg":

```
curl -X PUT -H 'Content-Type: image/jpg' --data-binary @300px-Endgame_album_art.jpg
http://localhost:5984/albums/album1/cover.jpg?rev="1-8a015dd26403219af66f05542cb540b2"
```

ATTACHMENTS (2)

- On adding an attachment to a document its version number changes:
  - For adding an attachment it is imperative to specify the version number of the document object
  - Whe an attachment is created, the special attribute "_attachments" is created
- The method GET enables the retrieval of the attachment through the corresponding attribute:

```
curl -X GET http://localhost:5984/albums/album1/cover.jpg?rev="2-
31e1ce62601aac5b9de7059788361641" > tmp.jpg
```

VIEWS

- Views are useful for many purposes:
  - Filtering the documents in your database to find those relevant to a particular process.
  - Extracting data from your documents and presenting it in a specific order.
  - Building efficient indexes (B-Trees)to find documents by any value or structure that resides in them.
  - Use these indexes to represent relationships among documents.
  - Views you can make all sorts of calculations on the data in your documents.
    - E.g., if documents represent your company's financial transactions, a view can answer the question of what the spending was in the last week, month, or year.

DEFINING VIEWS (1)

- Views are based on the working model MapReduce:
- Map and reduce function are specified in javascript
- Built-in views are provided
  - `curl -X GET http://localhost:5984/albums/_all_docs`

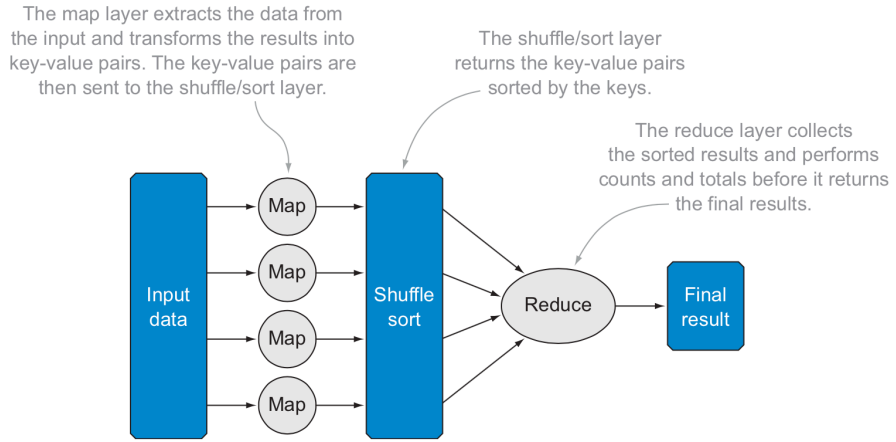


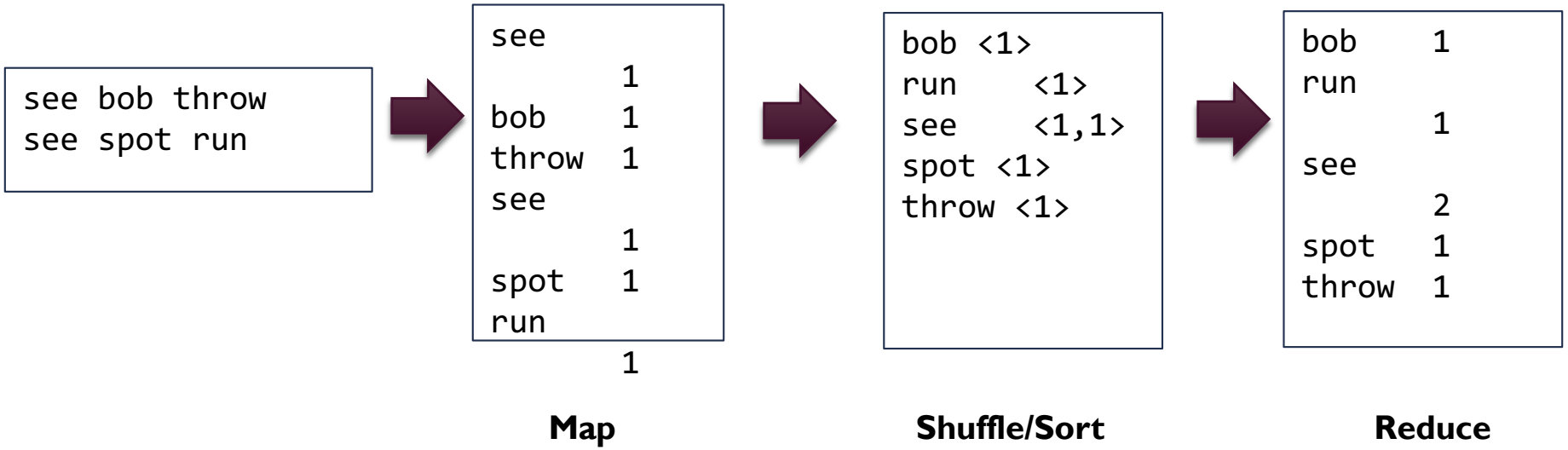
Figure 1.2 **The map and reduce functions are ways of partitioning large datasets into smaller chunks that can be transformed on isolated and independent transformation systems. The key is isolating each function so that it can be scaled onto many servers.**

EXAMPLE: DEFINING A VIEW

```
curl -X PUT http://localhost:5984/albums/_design/vistas1 -d @-
      {
          "language": "javascript",
          "views": {
              "por_anio": {
                  "map": "function( doc ) { if( doc.anio ) { emit( doc.anio, 1 );}}",
                  "reduce": "function( keys, values, rereduce ) {return sum( values );}"
              }
          }
      }
```

COUNTING WORDS EXAMPLE

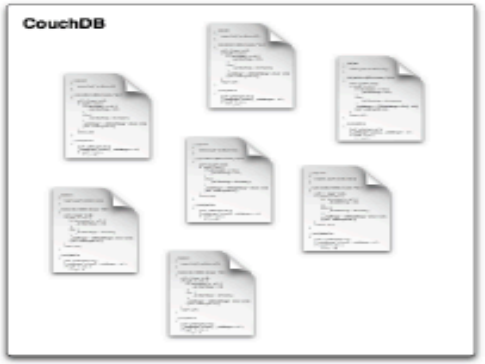
(URI, document) → (term, count)



QUERYING A VIEW

GET /dbname/_design/hats/_view/all?
include_docs=true

HTTP Client



EXAMPLE: USING A VIEW

- Reduce values retrieved without considering the keys:
  - `curl http://localhost:5984/albums/_design/vistas1/_view/por_anio`
  - `curl -X GET http://localhost:5984/albums/_design/vistas1/_view/por_anio`
- Reduce the values retrieved considering the values of the different keys:
  - `curl http://localhost:5984/albums/_design/vistas1/_view/por_anio?group=true`

MANAGEMENT TOOLS

- **Web management console** : complete interface for configuring , managing and monitoring a CouchDB instalation
- REST **API**: management interface exported under a REST HTTP protocol
- **Command interface**: tools providing information and control of a CouchDB instalation
  - Use the REST API
  - Can be used with scripts and management proceedures (*failover*, *backups*)

USING COUCHDB WITH JAVA (1)

- There are several projects that enable the use of CouchDB with Java
- Visit the following links:
  - CouchDB4Java (http://github.com/mbreese/couchdb4j)
  - JRelax (https://github.com/isterin/jrelax/wiki)

USING COUCHDB WITH JAVA (2)

- CouchDB4Java is easy to use, you only have to download the application and integrate the JARs located in the folder lib

- Java code for connecting an application:

```
public static void main( String [] args ) throws Exception {
Session s = new Session( "localhost", 5984 );
Database db = s.getDatabase( "albums" );
Document newdoc = new Document();
newdoc.put( "artista", "Megadeth" );
newdoc.put( "titulo", "Endgame" );
newdoc.put( "anio", 2010 );
db.saveDocument( newdoc, "album1" );
}
```

USING COUCHDB WITH JAVA (3)

JRelax

- For this case you have to download the project and its dependencies:
  - Restlet 2.0 (http://www.restlet.org/)
  - Jackson (JSON process - http://jackson.codehaus.org/)
- Particularly the following JARs:
  - `org.json.jar`
  - `jackson-all-1.6.2.jar`
  - `org.restlet.jar`
  - `org.restlet.ext.json.jar`

USING COUCHDB WITH JAVA (4)

Java code for connecting a JRelax application

```
public static void main( String[] args ) {
        ResourceManager resourceMgr = new
            DefaultResourceManager( "http://localhost:5984" );
        List<String> dbs = resourceMgr.listDatabases();
        for( String db : dbs ) {
            System.out.println( db );
        }
        Document doc = resourceMgr.getDocument( "albums", "album1" );
        System.out.println( doc.asJson() );
          }
```

EXAMPLE: ALBUM DB

Adding more documents to the Albums database

```
curl -X PUT http://localhost:5984/albums/album4 -d @-
{
    "artista": "Pantera",
    "titulo": "Reinventing the Steel",
    "anio": 2009
}

curl -X PUT http://localhost:5984/albums/album5 -d @-
{
    "artista": "Slayer",
    "titulo": "South of Heaven",
    "anio": 2009
}
```

BACK TO THEORY …