CouchDB

relax

Michael Parker

February 24, 2013

HACKER NEWS

What is CouchDB?

- One of those hipster NoSQL databases
- Document-oriented, not relational
  - No joins, prefer denormalization
- RESTful API for all operations
- *Views* add structure via secondary indexes
- Other perks:
  - Multiversion concurrency control (lock-free)
  - MapReduce framework
  - Great web admin interface
  - Multi-master replication

Document-oriented

- Like a key-value store, flat namespace
  - In CouchDB, key is called a *document identifier*, or *doc id*
- Documents stored in JSON format
  - "Schemaless," but the schema resides in your app
  - JSON also format for HTTP body
- Similar: MongoDB (no relation), Redis, Cassandra

JSON Document in CouchDB


```
{
  "_id": "emp_001",
  "_rev": "1-4c6114c65e295552ab1019e2b046b10e",
  "name": {
    "first": "Dante",
    "last": "Hicks",
  },
  "phone": "310-555-1212",
  "interests": ["philosophy", "Star Wars"]
}
```

RESTful URLs

- All resources in database identified by URL
- Basic URL structure:
  - **`/db_id`**: A database
  - **`/db_id/doc_id`**: A document in a database
- URL components and JSON fields starting with an underscore are special, e.g.:
  - **`/_config`**: CouchDB configuration parameters
  - **`/db_id/_all_docs`**: A cursor across all documents
  - **`/db_id/_design/design_doc`**: A design document for a database

RESTful Methods

- HTTP methods for CRUD actions
  - **`GET`**: Read data, typically a document
  - **`HEAD`**: Like **`GET`** without a body, typically used to check if a document exists
  - **`PUT`**: Creates new databases, documents, and other resources
  - **`POST`**: Updates these resources
  - **`DELETE`**: Deletes these resources

RESTful Status Codes

- HTTP status codes for server responses
  - **`200 OK`**: Request completed successfully (e.g. retrieving, updating, deleting documents)
  - **`201 Created`**: Resource created (used with **`PUT`**)
  - **`202 Accepted`**: Request completed and operation pending (e.g. for background operations)
  - **`401 Unauthorized`**: Bad username or password
  - **`404 Not Found`**: Resource missing
  - **`409 Conflict`**: MVCC failure, or concurrent modification to a document
  - **`500 Internal Server Error`**: Everybody panic

Creating a Database

- Request (abbr.):
  `PUT /new_db/ HTTP/1.1`

- Response (abbr.):
  `HTTP/1.1 201 Created`

  `{"ok": true}`

Creating a Database

- Request (abbr.):
  `GET /_all_dbs HTTP/1.1`

- Response (abbr.):
  `HTTP/1.1 200 OK`

  `["new_db"]`

Creating a Document

- Request (abbr.):

```
PUT /my_db/emp_001 HTTP/1.1

{
  "name": {
    "first": "Dante", "last": "Hicks",
  },
  "phone": "310-555-1212",
  "interests": ["philosophy", "Star Wars"]
}
```

Creating a Document

- Response (abbr.):

```
HTTP/1.1 201 Created

{
  "ok": true,
  "id": "emp_001",
  "rev": "1-4c6114c65e295552ab1019e2b046b10e"
}
```

Retrieving a Document

- Request (abbr.):
  `GET /my_db/emp_001 HTTP/1.1`

Retrieving a Document

- Response (abbr.):

```
HTTP/1.1 200 OK

{
  "_id": "emp_001",
  "_rev": "1-4c6114c65e295552ab1019e2b046b10e",
  "name": {
    "first": "Dante", "last": "Hicks",
  },
  "phone": "310-555-1212",
  "interests": ["philosophy", "Star Wars"]
}
```

Retrieving a Document

- Request (abbr.):

```
GET /my_db/missing_emp_007 HTTP/1.1
```

- Response (abbr.):

```
HTTP/1.1 404 Object Not Found

{
  "error": "not_found",
  "reason": "missing"
}
```

Views

- Secondary indexes for querying by other than `_id`
- Written in JavaScript, executed with Mozilla SpiderMonkey engine
- Defined in the design document

Views

- Define view **`emps_by_interest`**:

```
function(emp_doc) {
  for (var i = 0; i < emp_doc.interests.length; ++i) {
    var interest = emp_doc.interests[i];
    emit(interest.toLowerCase(), null);
  }
}
```

- Request (abbr):

```
GET
/my_db/_design/my_dd/_view/emps_by_interest?key=philosophy
HTTP/1.1
```

Views

- Response (abbr):

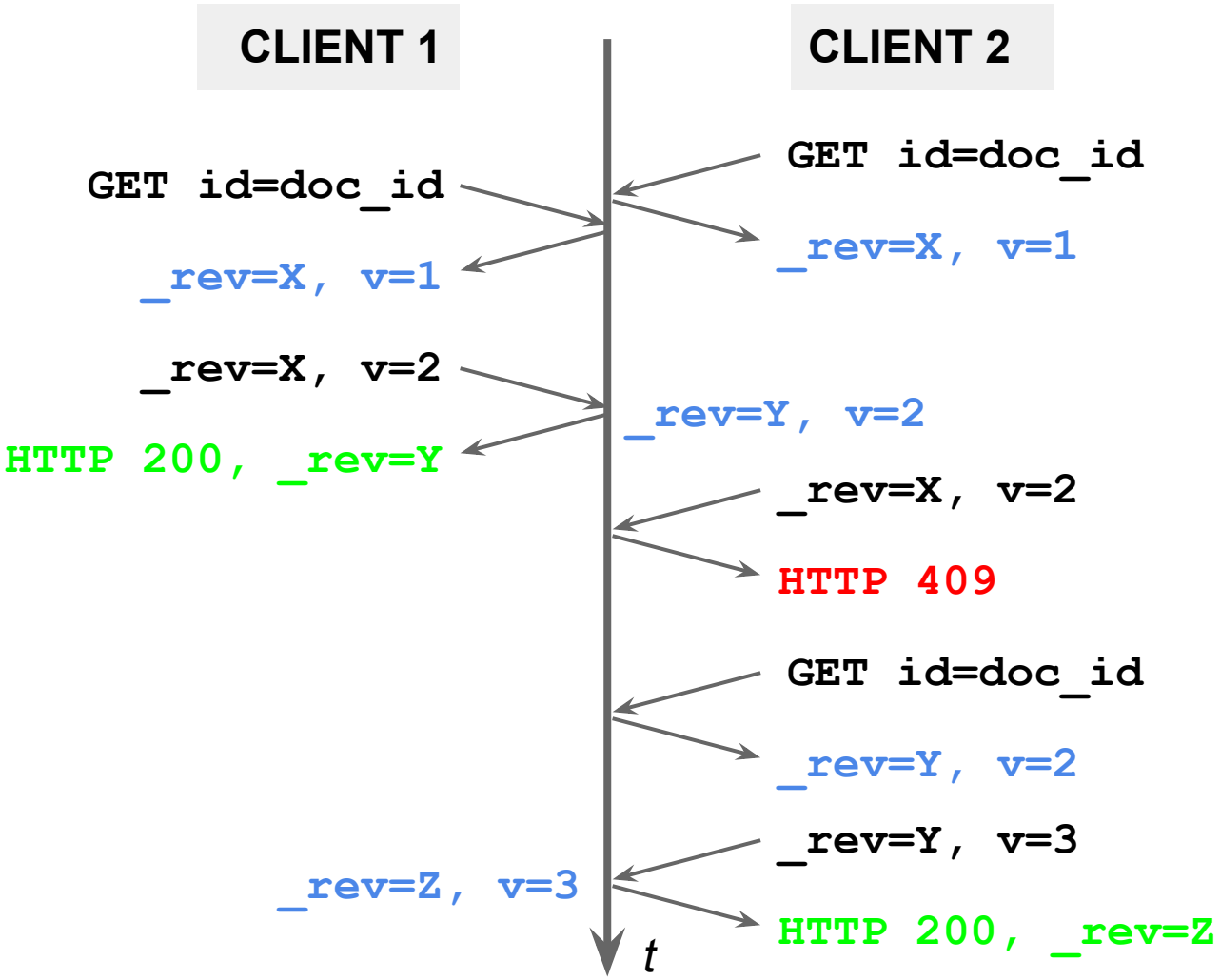
```
HTTP/1.1 200 OK

{
  "total_rows": 1, "offset": 0,
  "rows": [
    {"id": "emp_001", "key": "philosophy", "value": null}
  ]
}
```

- Append **`&include_docs=true`** in request to return documents with results

Multiversion Concurrency Control (MVCC)

- Every document has a **`_rev`** attribute
  - Only required field other than **`_id`**
- Ensures that client is updating latest data
  - No accidental clobbering
- Lock-free concurrency control
- If multiple clients attempt to write concurrently, exactly one succeeds every time
  - Always making "forward progress"

Multiversion Concurrency Control (MVCC)



Benchmarking

IRON CUSHION

`https://github.com/mgp/iron-cushion`

- Setup:
  - Server: Intel Core 2 2.83GHz quad-core, 4GB RAM
  - Client: 1.83 GHz Intel Core Duo MacBook
  - 100Mbit LAN, 100 concurrent connections
  - first, bulk insert 2,000,000 documents
  - second, intersperse 20,000 create and read operations, 30,000 update and delete operations
- Caveat: no indexes

Benchmarking

```
bulkInsertRate: 10,003.030 docs/sec

createProcessingRate: 949.141 docs/sec
readProcessingRate: 9,015.862 docs/sec
updateProcessingRate: 980.172 docs/sec
deleteProcessingRate: 980.154 docs/sec
```

Thanks!

http://couchdb.apache.org/

michael.g.parker@gmail.com
https://github.com/mgp

http://mgp.github.com/couchdb-la-hn.pdf