A Case Study of Python and "No-Sql" Databases

Mozilla Raindrop and CouchDB

Mark Hammond

About Me

- Core Python committer.
- Primary developer and maintainer for the pywin32 package.
- Based in Melbourne, Australia.
- Currently working for Mozilla Messaging on Raindrop.

No-Sql Databases?

- Name is a misnomer
  - Some *could* use SQL if they wanted
    - But would be slow for many SQL operations – eg, joins
  - Tend to not be relational
  - Trade immediate consistency for scalability
    - Use terms like 'eventual consistency'
    - Push some concerns back to application code
- Designed for distributed scalability

Glorified key-value stores

- Sometimes 'document' oriented
- No central schema or validation
- No referential integrity
  - Almost impossible in a distributed system anyway.
  - App must manage this.

Map-Reduce

- Often use a variation of map-reduce to process data
  - Map phase calculates data for a single entity
    - Eg, single “document” or record.
  - Reduce summarizes or aggregates individual “map” results
- Particularly suitable for distributed processing
  - Can give different “chunks” to different nodes.

No-Sql implementations

- Hot new topic in the database world
- Many contenders
  - CouchDB
  - MongoDB
  - Google Bigtable
  - Cassandra
  - RIAK
- Each has different use-cases – choose carefully.

CouchDB

- Document oriented, schema-less
- Uses HTTP for the API
  - Use GET to fetch documents
  - PUT to save new documents
  - Eliminates requirement for bindings
    - Any HTTP client can talk to CouchDB
  - But bindings exist anyway!
    - Usually with convenience functions
    - Often more trouble than they are worth

JSON Oriented

- Uses JSON over HTTP for fetching and saving documents.
- Documents are persisted on disk using a simple JSON persistence mechanism.
  - “Attachments” used for BLOBS.
- Views, externals and all other features based on JSON.
- Python has excellent JSON support.

JavaScript view engine

- Views are defined using Javascript.
  - Map and reduce functions are Javascript functions.
  - CouchDB itself written in erlang
- View engines available for many other languages, including Python.
  - But views must be 'referentially transparent', so you can't use all its power.

Pre-built View/Query system

- Documents passed through a map function and result saved.
  - Extremely fast queries for existing documents.
  - Updated upon view request rather than as documents added
  - Fairly slow queries with many new documents – all must be passed though map function
    - But fast after that!
- Reduce may require per-query processing

CouchDB – Robust Storage

- Uses “append-only” operations for data integrity
  - Chews disk-space until explicit compaction
- Uses “crash-only” server termination
  - No real “shutdown” process – server just stops.
  - If you have to handle your server crashing, why not make that the default?

CouchDB – Builtin Replication

- Master-master replication built-in to initial design.
- Deceptively simple process
  - 'Change sequence' numbers with md5 hash of document body.
  - Replication remembers last sequence and restarts from there.

Conflicts handled by Application

- Any distributed system introduces possibility for conflicts
- CouchDB *detects* conflicts
  - Retains all conflicting versions
  - Arbitrary conflicting version used while unresolved.
- Application implements resolution of conflicts
  - Only the application knows how to handle this

Attachments

- Documents can have any number of binary attachments.
- Attachments individually addressable.
- Combined with HTTP API, CouchDB trivially hosts web-apps
  - HTML, Javascript, CSS etc all suitable for attachments.

Mozilla Raindrop

- Mozilla Messaging
  - Who brings you Thunderbird
- Experiment in the future of messaging applications
  - Web based
  - Suitable for mobile devices.
- Currently using CouchDB for storage and for hosting Javascript front-end code.

Python in the back-end

- Talking to IMAP, Twitter, Skype, parsing mime, etc
- Stores documents in the DB for later consumption by the front-end
- Implements the runtime REST API
    - CouchDB arranges to start a Python process on demand.
- May move *away* from CouchDB

Experiences

- To be fair, not yet 1.0
- Append-only model causes huge disk usage
  - Can be compacted, but this is very IO intensive.
  - Best for databases with few writes relative to reads.
- View model is inflexible
  - Map can only see one document – if data spans 2 documents you are screwed.
  - Adding new views has a huge cost in

Scalability we don't need

- Capable of scaling and multi-master writing
  - But raindrop doesn't need that!
  - Personal email doesn't need thousands of people hitting the same database.
- Cost-per-user is the scalability we require
  - End-user will not be paying.
  - Number of cents per user per year matters.

Massive data redundancy

- Data is heavily denormalized to work around lack of joins
  - Often create "summary documents" with 100% redundant data
- Further contributes to performance issues.
  - Extra documents and multiple modifications impacts disk usage.
  - Extra document indexing impacts CPU performance.

Cost-effective hosting challenges

- Difficult to load-balance CPU and Disk intensive tasks
  - View-indexing and compaction suck performance, but hard to run on other nodes.
- Replication requires view reindexing
  - Replicated database effectively unusable while this happens
- Supporting many users per node seems unachievable
  - Closer to “how many nodes per user”

Summary

- No-Sql is the hot new kid on the block
- Python has bindings for many No-Sql databases
  - But also for many traditional databases – choose what is right for you.
- Research various implementation strengths and weaknesses.
  - Various tradeoffs to be made between reliability and performance.
  - No magic bullet.

Summary (cont.)

- Don't over think your scalability requirements
  - Everyone wants to scale as if they they are building the next twitter
  - But twitter evolved that scalability
  - Worry more about that problem when you actually have it!
- Design for fast iteration
  - Give yourself the ability to respond to changes in requirements and knowledge
  - Eg, Python!

Questions?

- Any questions?
- Contact
    - mhammond@skippinet.com.au
- Thanks for coming!