Finding Change in the CouchDB

Dr. James E. Marca

January 13, 2013

Big Data

We've got big data!

New tools: non-relational databases

(aka “NoSQL”)

Old tool: PostgreSQL

The question:

"Is the current reading of volume and occupancy from some detector more or less than what should be expected?"

Old way

1. Precompute some average value
2. by time of day
3. and day of week
4. Compare.

But we had all the data

In PostgreSQL

Ready to query dynamically, so. . .

Big Query

"What is the average volume and occupancy for every 5 minute period over the prior 365 days (less holidays) for any loop detector"

Slow

- The index was too big to fit in RAM,
- The index was swapped in and out
- Super duper slow!

My database broke

Big Data, NoSQL

- Google (Big Table)
- Amazon (Dynamo)

Options we considered

- flat files
- CouchDB
- TokyoTyrant
- Cassandra
- Hadoop

Options available now

- Hadoop
- Riak
- MongoDB
- CouchDB
- Cassandra
- RethinkDB
- Voldemort
- Datalog
- and more all the time. . .

Why do I think CouchDB is good for transportation data?

Transportation data

- largely observations and measurements
- write once
- read raw data for short term applications
- process raw data into spatio-temporal summary stats

Example: Loop Detectors

- 30s volume, occupancy
- some misses and noise
- run-once cleanup procedures

Example: Personal GPS and activity stream

- second by second GPS
- slowly growing sets of:
  - routes,
  - destinations,
  - time windows
- small set of repeated queries:
  - Optimize likely activities?
  - Does traffic affect my usual pattern?
  - etc

Why do I think CouchDB is good for transportation data?

What is CouchDB?

What is a database?

- store data
- get data
- allow multiple users

Compare with flat files

- One file per loop detector per year
- Append to file as data arrives
- Easy to organize and distribute

Flat files are:

- Good for data cleaning, stats generation, imputation of missing data, etc
- Fine for single user
  - Consistent
  - Available
- Not *performant*

Is performant a word?

Flat files can lead to trouble

- Bad for multiuser
  - race conditions,
  - version problems,
  - etc
- Difficult to query

"What was the volume and occupancy like last Monday?"

So use a database

Limited by the CAP theorem

- **C**onsistency
- **A**vailability
- **P**erformance

Choose any two

CouchDB

- Chooses **A**vailability, **P**erformance
- Document model, not schema based

"Eventually Consistent"

Consistency isn't that big a deal for traffic data

- Events are observed by one or more sensors
- Sensors write their observations

Events vs Observations

- The *observations* don't change
- The interpretation of the *event* might change
- But a consistent interpretation isn't mission critical

Bonus: CouchDB has master-master replication

Replication is:

- super awesome
- a massive *change*

Replication enables a new data collection architecture

- Put the database at the detector (distributed databases)
- Move data around by using replication

Examples:

- Central database replicates all detector databases
- Global queries can go to central DB
- Local queries can go directly to detector DBs
- A traveler can replicate only the traffic DBs along common routes

All replicating databases will be eventually consistent

Relax

Practical experience

what we are doing with CouchDB

Processing Raw Loop Detector Data

Orange County, California (CalTrans District 12)

- about 900 mainline detectors
- Process in R
  - compute 27 different measures per location
  - for 20 minute running time window
  - (vol, occ per lane + 27) per 30s period
  - run models estimating relative risk of accident types

Database design:

- One CouchDB database per detector
- One document per day

Document per day reasons:

- Based on ~~painful experience~~ informal testing
- One document per year is too big to process
- One document per timestamp would work okay,
- But the web *application* uses daily data
- CouchDB sorts by document id, id is based on timestamp
- HTTP GET:
  /vdsdata/d12/2007/1202248/ 1202248 2007-01-03 00:00:00

Database per detector per year reasons:

- One big database is possible, but
- Prior to BigCouch, impossible to *split* or *shard* over different machines
- makes better use of multi-core machines when generating views

Use views to run models and summarize data

- Views are CouchDB's version of map/reduce
- Write JavaScript code for the map function that is run on each document (to apply models, run summaries, etc)
- ProTip© Only use embedded Erlang reduce functions like `_count`,`_sum`, and `_stats`

Difficulty: Need to use another database to collate model output

- Pipe summaries of the per-detector views to a single database for all detectors in the district
- Requires external programming
- Difficult to automate

Application 2: Storing the results of imputation runs

- Similar to prior example
- Databases spread over three machines
- Uses per-district collation databases
- Analysis step used CouchDB to coordinate multiple processes
  - Local "state" database on each machine
  - State databases replicated with each other
  - No overlapping runs were observed

Application 3: Convenient stash for detector information

- Uses GeoCouch extensions, stores location of each detector
- stashes all known information about each detector in a single place
- uses binary attachments to save R analysis output (plots, files)

Questions?

James E. Marca
UCI ITS
jmarca@translab.its.uci.edu