Finding Change in the CouchDB

Dr. James E. Marca

January 13, 2013

Big Data

We've always had big data!

Old tool: PostgreSQL

Old queries:

"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 with current

But we had a new server and a terabyte drive

... and all the raw data for a year ...

... In PostgreSQL ...

... Ready to query dynamically ...

A Big Query using "WITH" and subselect clauses

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

My database broke

- The index was too big to fit in RAM,
- Extensive disk swapping
- Super-duper slow!

Alternatives for Big Data: NoSQL

- Google (Big Table)
- Amazon (Dynamo)

Options we considered

- flat files
- CouchDB
- TokyoTyrant
- Cassandra
- Hadoop

Options available now

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

Why do I think CouchDB is good for transportation data?

What is transportation data?

- largely observations and measurements
- write once
- read raw data for short term applications
- process raw data into 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?

- A document oriented *database*

What is a database?

- store data
- get data
- allow multiple users

Why not just use flat files?

Consider loop detectors

- One file per loop detector per year, or
- One file for all detectors per day (what PeMS does)
- Easy to organize and distribute
- Fine for single user

But 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

Databases are limited by the CAP theorem

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

Choose any two

CouchDB

- Chooses **A**vailability, **P**erformance

  "Eventually Consistent"

Consistency isn't that big a deal for traffic data

- This isn't stock trading or e-commerce
- It's okay if:
  - you are 30s behind reading a loop
  - Controller A has slightly more current info than Controller B

A universally consistent view isn't mission critical

Bonus: CouchDB has master-master replication

Replication is:

super awesome

the *change* I was looking for

Replication enables a new data collection architecture

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

Uses:

- A TMC can still pull-replicate from all detector databases
- A Local or Freeway specific TMC can limit replication to relevant detectors
- A traveler can replicate only the traffic DBs along common routes

All replicating databases will be eventually consistent

Master:Master means all are equally good candidates for replication

Relax

Practical experience

1. Processing, storing raw loop detector data
2. Imputing missing detector data
3. A single stash for storing detector metadata

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
- 280GB of data (three years)
- 590GB of generated views

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
- Impossible to *split* or *shard* over different machines (now can use BigCouch)
- 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
- 400GB of data
- 700GB of generated views
- 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 metadata

- 20GB of data
- 222MB of generated views
- Uses GeoCouch extensions, stores location of each detector
- Stashes all known metadata about each detector in a single place
- Uses binary attachments to save R analysis output (plots, files)
- Small enough to replicate to my laptop too

Questions?

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