Tokyo Cabinet and CouchDB with Mnesia

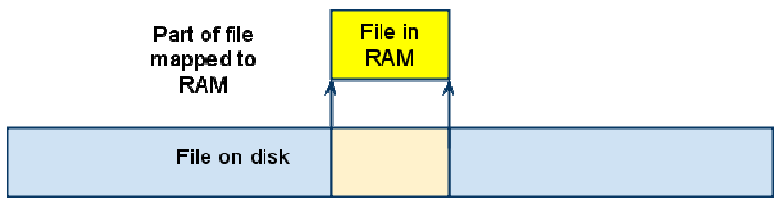
by Rickard Cardell

Tokyo Cabinet

- Key-value store
- space efficient
- several storage types:
  Hash, B+tree and more
- several API:s: Perl, Java, Ruby, LUA, Erlang
- apps for distributing: Tokyo Tyrant
- used by large community

Tokyo Cabinet cont.

- disk resident - both in RAM and on disk
- need sync() for resident storage
- no repair of broken tables
- mmap() - memory mapped file



CouchDB -basic features

- made in Erlang!
- HTTP Restful interface
- replication
- non-sql
- views for queries
- documents for storage
- no type constraints in database
- MVCC -MultiVersionConcurrencyControl
  - revisions
  - no locks or transactions
  - conflict resolution on application level
- non destructive updates
- much more..

Mnesia's shortcomings

- infamous 2GB table limit of DETS
- ETS is RAM hungry
- repair broken table takes time

Prerequisites

- a large system highly integrated with Mnesia
- had to integrate my solution to the system
    - replacing Mnesia is a big effort
        - all data stored in tables as Erlang terms
        - need to replace lots of mnesia:select to X:select
        - table definitions as records - untyped
        - complex relations between tables

- How to solve this?
    - Use a totally different DBMS

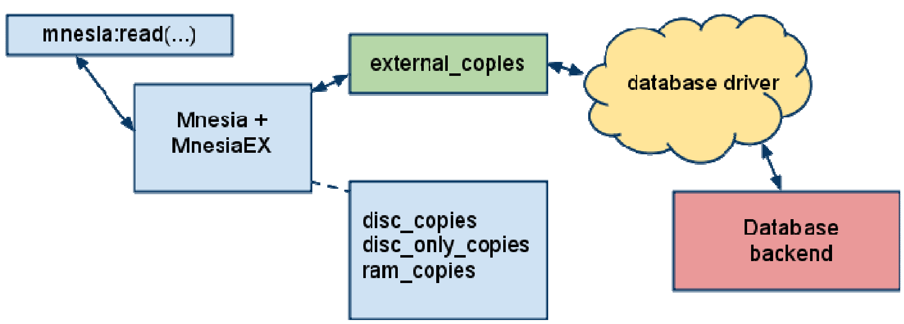
      or
    - Replace ETS and DETS in Mnesia

My solution

- make backends of Tokyo Cabinet and CouchDB
  - less code changes to the system
  - transparent to the user
  - will make use of Mnesia locks and transactions
- already an extension to Mnesia: MnesiaEX

MnesiaEX - Mnesia extension

- ability to apply arbitrary storage to Mnesia
- almost transparent to the user
- adds a new storage type external_copies
  - works together with current storage types
- ACID issues
- Tokyo Cabinet has already API: Tcerl



Tokyo Cabinet with MnesiaEX -Tcerl

- API for Tokyo Cabinet B+tree:*Tcerl*
- written by Paul Mineiro
- used in production

- interface to Mnesia via linked-in-driver
  - speed over uptime
- good support for mnesia's functions
  - select, match_object, read, write, next, previous ...
- ordered_set
- store the records as binaries
- sync or async writes
- need clean exit

CouchDB with MnesiaEX - Cdberl

- implemented Mnesiaex behaviour for CouchDB
  - named it Cdberl

- Multi Version Concurrency Control means no locks/transactions
- ignored MVCC
  - can't use replication
  - can't use revisions
- using the HTTP interface
- Erlang terms to JSON
- cache revisions for faster updates

- improvements to do: use bulk documents

Cdberl - impedance mismatch

- map/reduce want JSON, not binaries

- no direct translation from Erlang terms to JSON
  - non trivial problem
  - example: BigInt

Cdberl - queries

- a query needs a precomputed view

- mnesia:match_object -> create a view and then invoke
  - not very dynamic
  - long time to generate views

Representing Erlang terms in JSON

Erlang		JSON
atom, >asdf		string, >"asdf"
list (string), >"otp"		array, >[111,116,112]
Integer, >1234		>32-bit Integer/float 1234
tuple, >{1,2,3,4}		object with array, > {tuple, [1,2,3,4]}

```
example:
1>to_json({person, 1}).
"{obj:{tuple:[person, 1]}}"
```

Stress testing

- TPC-B, a standard DBMS benchmark / stress test
- measures transactions per second
- updates to four tables per transaction
  - 3 reads, 3 writes, 1 update
- serial transactions

- Account table, 100000 records/rows
- Teller table, 10 records/rows
- Branch table, 1 record/row
- History table, 0 records/rows from start

TPC-B -result

Result:

- ram_copies:	5000tps
- disc_copies:	4200tps
- Tcerl (large cache):	2000tps
- Tcerl (small cache):	1200tps
- disc_only_copies:	200tps
- Cdberl:	30 tps

Stress test -result cont.

Disk space of database files

Account table, 100000 records. Actual disk size:

Cdberl (CouchDB):	111MB / 30MB*
disc_copies:	21MB
disc_only_copies:	15MB
Tcerl (TokyoCabinet):	4MB
ram_copies:	n/a

* before and after compaction

My conclusion

- CouchDB
  - robust storage
  - easy to create powerful views
  - easy to communicate with
  - easy to replicate
  - growing user base
  - no load time on startup
  - designed for parallell use

  - takes time to generate a view on large table
  - no real dynamic queries
  - a bit slow write performance
  - quite large files until compaction
  - doesn't integrate well with Mnesiaex

My conclusion cont.

Tokyo Cabinet

- integrates quite good with Mnesia
  - although experienced memory leaks and crashes
- good read and write performance
- simple API
- very small database files
- no startup time on load

- little documentation
- only one developer
- must be synced to disk

So...

- Mnesiaex is a fine interface
    - very easy to apply other database manager

- Tokyo Cabinet and Tcerl need more investigation in regard to durability issues.
- CouchDB, can be a part of the system, but probably not the general solution for Klarna

Questions?

Appendix - misc info

Cdberl source code and information is located at GitHub: http://github.com/RCardell/cdberl
The TPC-B benchmark that I've used can also be found there.

All tests ran for between 15minutes to 4hours, ~20 times per storage types until stable result was found. The tables was checked for consistency afterwards.
Test setup:
Erlang/OTP R12B5 w. HiPE (default setup was fastest)
Mnesiaex 4.4.7.6 http://code.google.com/p/mnesiaex/
CouchDB 0.90

Tokyo Cabinet 1.4.21
- bucket number: 2-5 times n records
- size of leaf node cache
small cache setup: smallest possible = 1
large cache setup: best result with n = ~5000x

Tcerl 1.3.1h http://code.google.com/p/tcerl/
Tcerldrv 1.3.1g

Ubuntu 9.04 64bit
1x4 Cores
8 GB RAM
2 SATA Disks Raid 0

Rickard.Cardell@gmail.com