Erlang Solutions Ltd

Mnesia Backend Plugin Framework and a LevelDB Based Plugin

Presented 2012-05-28 on EUC by

Roland Karlsson - ESL

Malcolm Matalka - Klarna

Introduction

- Malcolm Matalka
  - Joined Klarna in February
  - Worked as developer in banking and academia before Klarna
  - First task at Klarna to integrate LevelDB

- Roland Karlsson
  - Joined Erlang Solutions in August 2010
  - Main Erlang work in telecom
  - Worked at CSLAB at Ericsson some years
  - Done lots of other programming and hardware related stuff
  - Originally researcher in Computer Science and Physics

Overview of the talk

- Motivation
- What it is
- Usage
- Implementation
- Testing/Experience
- Wrap Up

Motivation

- Klarna has lots of data in Mnesia, and it is growing ...
- Mnesia is slow to start with a lot of data
- Long start up times scare us
- Some table types have 2 GB limit, annoyance
- Long term plan is to move out of Mnesia to Riak and Postgres
- Need a stop-gap though
- Say hello to ESL ...

What is it?

- Plug in framework (aka EXT)
  - (Based on work by Ulf Wiger - mnesia_ext_filesystem)

- A LevelDB based plugin (mnesia_ext_eleveldb)

Why LevelDB?

- Basho approved (used as a datastore in Riak)
  Basho also have a well tested Erlang binding for LevelDB
- Interface maps to Mnesia well
- Just plain feels more trustworthy than other KV stores

LevelDB

- Key-value store library from Google (inspired by BigTable)
- Keys are a set
- Stores keys sorted (good for prefix searches)
- Some optimizations
  - Supports prefix search
  - Tunable options for cache size, write buffer, block size
  - Fast compression through Snappy library
  - Supports batch writes

Why a framework?

- Too hard to keep adding new backends
- There is no backend interface layer in Mnesia
  (The current backends are accessed in hundreds of places)
- The transaction flow is complicated
- Some functions, e.g. table conversions, are tricky

- In short - it's very hard without a framework

How to use it

- Register a plugin under a name, aka alias
  - `mnesia:add_backend_type(Alias, Module)`

- Create a table of type ‘alias’
  - `mnesia:create_table(Name, [ {Alias,[node()]} ])`

- Use it as normal

How to make a plugin?

- You need some kind of key/value store
- You need to make an interface module to that store (and maybe also an Erlang binding)
- The interface module has to implement the mnesia_backend_type behaviour:
  - Query the plugin properties
  - Create/load/destroy tables
  - Key/value operations, such as insert, lookup, ...

The mnesia_backend_type behaviour

```
{add_aliases, 1},            % (Aliases)
{check_definition, 4},       % (Alias, Tab, Nodes, Properties)
{create_table, 3},           % (Alias, Tab, Properties)
{delete, 3},                 % (Alias, Tab, Key)
{delete_table, 2},           % (Alias, Tab)
{first, 2},                  % (Alias, Tab)
{fixtable, 3},               % (Alias, Tab, Bool)
{init_backend, 0},           % ()
{info, 3},                   % (Alias, Tab, Item)
{insert, 3},                 % (Alias, Tab, Object)
{last, 2},                   % (Alias, Tab)
{load_table, 3},             % (Alias, Tab, Reason)
{lookup, 3},                 % (Alias, Tab, Key)
{match_delete, 3},           % (Alias, Tab, Pattern)
{next, 3},                   % (Alias, Tab, Key)
{prev, 3},                   % (Alias, Tab, Key)
{real_suffixes, 0},          % ()
{remove_aliases, 1},         % (Aliases)
{repair_continuation, 2},    % (Continuation, MatchSpec)
{select, 1},                 % (Continuation)
{select, 3},                 % (Alias, Tab, Pattern)
{select, 4},                 % (Alias, Tab, MatchSpec, Limit)
{semantics, 2},              % (Alias, storage | types | index_fun)
{slot, 3},                   % (Alias, Tab, Pos)
{tmp_suffixes, 0},           % ()
{update_counter, 4},         % (Alias, Tab, Counter, Val)
{validate_key, 6},           % (Alias, Tab, RecName, Arity, Type, Key)
{validate_record, 6}         % (Alias, Tab, RecName, Arity, Type, Obj)
```

Implementation details (framework)

- Index tables are the same type as the primary table
- To facilitate usage, the plugin needs to be registered
- Does not support table conversion from EXT
- 5000 lines changed or added in the Mnesia application

Implementation detail (LevelDB plugin)

- Uses eleveldb by Basho
- Does not support bag semantics
- Optimizations
  - Supports prefix optimization for select/match
  - Table size tracking is optional, for performance
- 650 lines of code
- Over to Klarna ...

Testing methods

- Using was easy, just call register function and then set table type
- Initial testing just running our Common Test suite
- Basho Bench using live data
  - Started with subset of data
  - Migrated almost all of our data

Experience - The Good

- Application performance the same
- First attempt reduced startup time by 1/4
- We are confident we can reduce it to 1/2 or more
- Reduced startup memory by 2/3
- Basic migration straight forward, just change copy type
- Great for testing as we can use machines with less resources

Experience - The Bad

- Unproven technology, so far stable though
- Code is a bit messy - performance vs purity
- Reading deleted keys in LevelDB expensive (fixed in latest eleveldb)
- Migrate out of LevelDB not supported by the framework
- LevelDB raw speed less than DETS but robust

Experience - Watch Out

- Searching for a key that starts with a wild cards will do a full table iteration
- Iterating the whole table might blow out the LevelDB cache
- Some semantics are different (table_info)
- Does not track table size (by default)

Wrap Up

- Not in production but heading there
- Plugin framework coming to an OTP release near you (R16?)
- LevelDB plugin, not scheduled for release yet

- Thank you for listening!