BugSense

Big Data for mobile with Erlang, C, LISP

Dionisis "Dio" Kakoliris
Head of Engineering
dgk@bugsense.com

BugSense Trivia

- Third biggest mobile SDK in the world
- Analyzing data from ~400M devices
- More than ~120k writes/updates per sec
- Custom Big Data database (LDB)
- Eleven engineers
- Cash positive

Data map landscape

Data processing landscape

Enter LDB

Distributed concurrent updates

Overview

- Complex Event Processing, In-Memory DB
- Time-series Stream Processor
- Super easy to setup/use - one package
- No big locks (fine-grained locking)
- Describe-your-data mentality
- C is fast
- C is ideal for destructive updates (imperative)
- "Let it crash!"

Overview (cont'd)

- LISP-like DSL for custom processing / views
- Ideal for parallelism (functional)
- Lazy loading of files (asynchronous)
- Saves data to disk (asynchronous)
- Modular / Extendable architecture
- Custom reducers / off-line processing

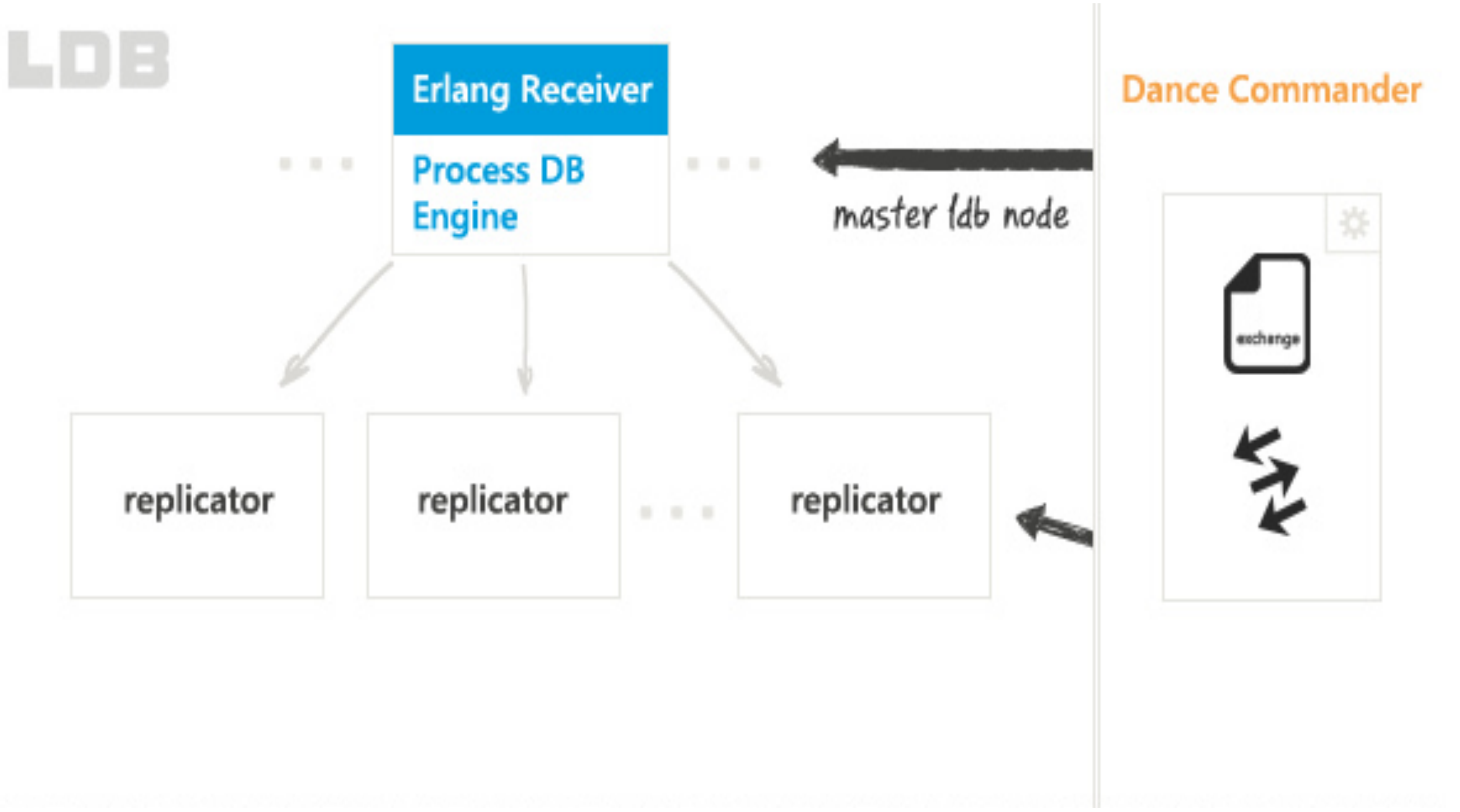
...And now, let's go BIG

- We love Erlang!
- Node isolation, replication, supervision
- Request processing and forwarding
- Distributed
- Ideal for building real-time systems
- Fast, robust, reliable
- YOU TRUST ERLANG

Why Erlang?

- Scaling
- Handles lots of connections efficiently (HTTP)
- Sending/receiving messages from/to nodes is trivial
- Building a replication/take over engine is easy
- Mnesia for storage and shared info
- C Linkedin Drivers
- Being able to connect to remote nodes
- All of the above - integrated in one language

Architecture



Snippet

```
RDLOCK_HASH_STR(kvm1->hashtab, id, bucket, 1, &hashv1, &hashv2,
          &bloom);
if (!bloom) {
     RDUNLOCK_HASH_STR(kvm1->hashtab, bucket);

     return 0;
}

FIND_HASH_STR(kvm1->hashtab, bucket, id, val, struct kvm1_val_t *);
if (val) {
     result = val->cntr;
     RDUNLOCK_HASH_STR(kvm1->hashtab, bucket);

     return result;
}

RDUNLOCK_HASH_STR(kvm1->hashtab, bucket);
```

Erlang <--> LDB

- Erlang is the right tool for the job
- Small language that handles a very crucial sector
- Actors are magnificent!
- Mnesia is your "faithful" companion
- Constantly evolving / getting better
- Erlang processes are your (million) friends!

More at: ***http://highscalability.com/blog/2012/11/26/bigdata-using-erlang-c-and-lisp-to-fight-the-tsunami-of-mobi.html***

Snippets...

```
spawn_ldb_app(NumOfKeyCells) ->
        Pid = spawn(ldb_appnode, create, [self()]),
        receive
                {ok, Port} ->
                        io:format("New ldb_app started ~n", []),
                        Pid ! {?START, self(), integer_to_list(1), integer_to_list(NumOfKeyCells)},
                        receive
                                {ok, []} ->
                                        ok
                        end
        end,
        {Pid, Port}.
```

Snippets...

```
create(ParentPid) ->
        Port = open_port({spawn_driver, lethe_drv}, [binary]),
        ParentPid ! {ok, Port},
        loop(Port).

loop(Port) ->
        receive
                {?VIEW, Caller, Key, Value} ->
                        port_command(Port, term_to_binary({?VIEW, Key, Value})),
                        receive
                                {ok, []} ->
                                        Caller ! {ok, []};  % THIS SHOULDNT BE FOO!
                                {view, Key, Value, Result} ->
                                        Caller ! {view, Key, Value, Result};
                                _ ->
                                        ignored
                        end,
                        loop(Port);
```

Bonus Stage: Why LISP?

- Very easy lexer, parser, analyzer, interpreter impl.
- SQL-ish queries
- Expressive power, abstraction
- S-expressions, heterogeneous multi-dim. lists
- Immutability => Ideal for parallel computing
- Tons of Data => Prefix notation
- Tons of Data => Data transformation FTW!

Yet another snippet...

```
(define (in-all? arrays uid)
  (reduce (lambda (x acc) (and x acc))
          (map (lambda (x) (in? x uid))
               arrays)))

(define (in-all-timespan arrays uid from to)
  (map (lambda (x)
         (timebubble (timewarp (current-timespace) x)
                     (in-all? arrays uid)))
       (range from to)))

(define (at-least-once-in-all? arrays uid from to)
  (reduce (lambda (x acc) (or x acc))
          (in-all-timespan arrays uid from to)))

(define (always-in-all? arrays uid from to)
  (reduce (lambda (x acc) (and x acc))
          (in-all-timespan arrays uid from to)))

(define (never-in-all? arrays uid from to)
  (not (at-least-once-in-all? arrays uid from to)))
```

Moving Forward

- Not easy hot code swapping
- Auxiliary tools for monitoring, support, configuration
- GUIs for the above
- Cross-platform
- Interoperability with other systems
- Support role for big and complex systems
- Tweaking - LDB has the potential to run everywhere!
- ...From your cell-phone to an HPC cluster!

And?

- Use Erlang for your projects!
- It's small, easy and above-all it DELIVERS!
- Don't be scared by the complexity of modern systems
- Distributed systems are the present and future
- Harness this massive potential with Erlang
- START A PROJECT NOW!

Thank you!

Stay tuned for fresh stuff!
www.bugsense.com
blog.bugsense.com