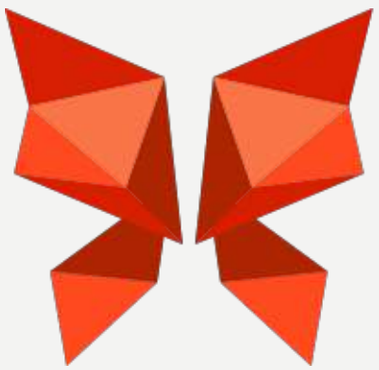
CODEBEAM 2019 Stockholm - 05/2019

GEN_PERSIST

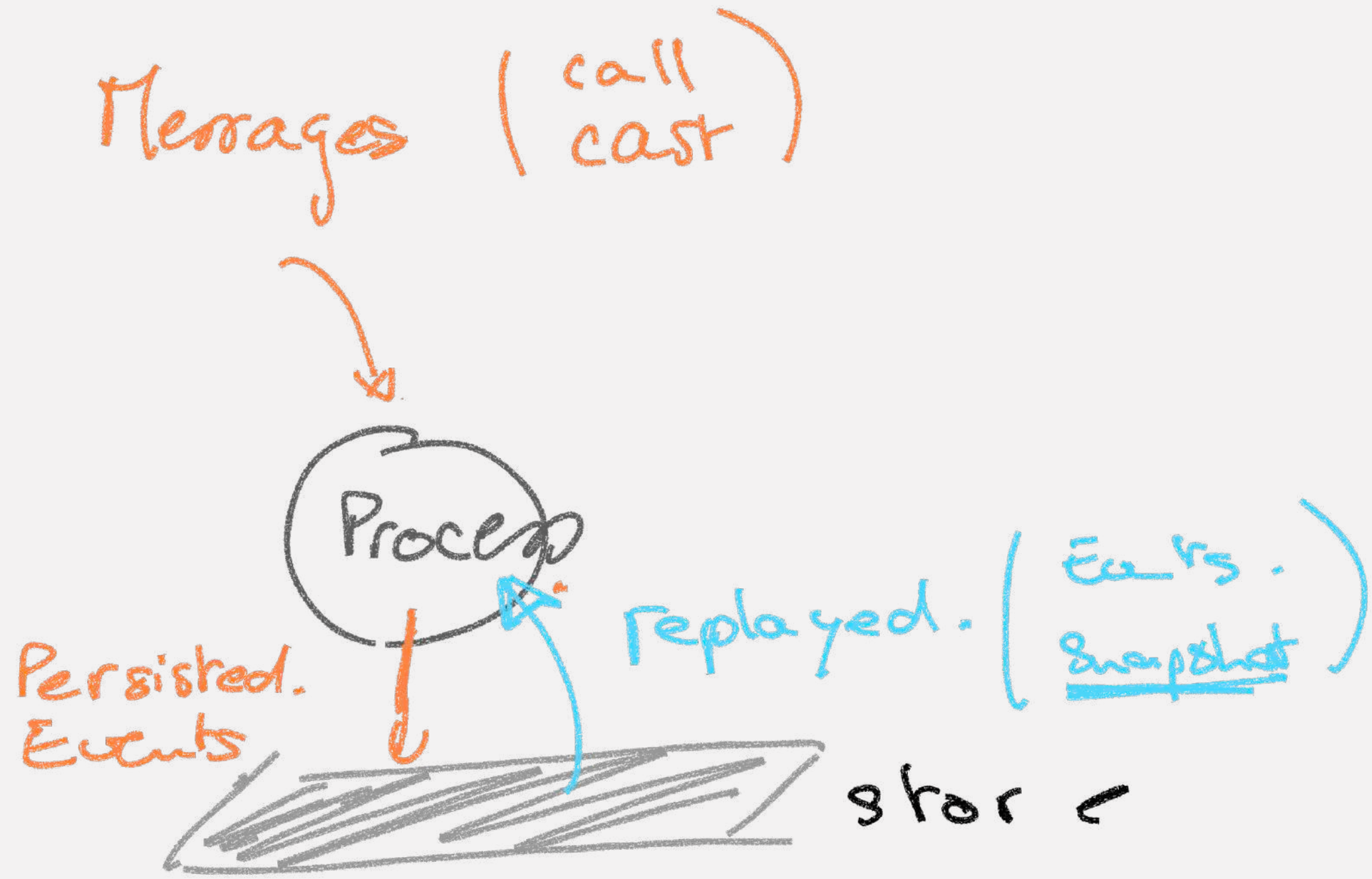
PERSIST THE STATE OF YOUR PROCESSES



enki multimedia

persist the state of your process

enki multimedia



overview

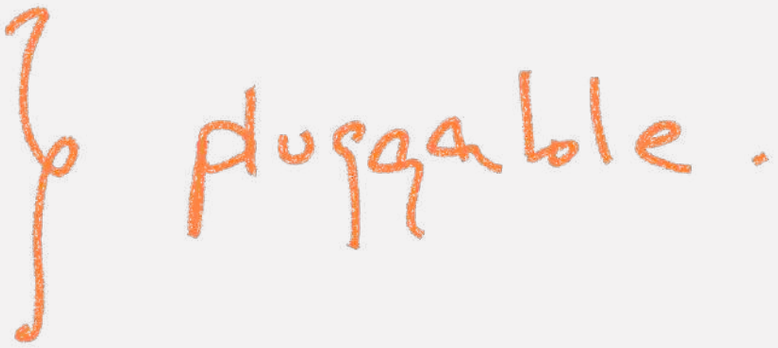
enki multimedia

- persist the state of your processes
- recovered on start (restart, crash, migration ..)
- events are stored, not the state itself
- snapshots are available

overview

enki multimedia

- Persistent server process
- Journal: append only log
- Snapshot storage
- Event sourcing



architecture

enki multimedia

a quick glance

enki multimedia

unique persistId in the store.

```
start_link() ->
  PersistId = <<"test-1">>,
  erlang:send_after(1000, self(), do_snapshot),
  persist_proc:start_link(PersistId, ?MODULE, []).

init([]) ->
  {ok, #{}}.
```

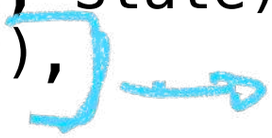
init

```
%%%
% > persist_proc:cast(Pid, <<"hello">>).

handle_command({call, _From}, Msg, State) ->
  persist_proc:persist(to_binary(Msg)),
  {reply, ok, State};

%% receive : <<"hello">>
handle_command(cast, Msg, State) ->
  persist_proc:persist(to_binary(Msg)),
  {reply, ok, State}.

%% receive
%% #{ seq := Seq,
%%    data := <<< "hello">> }
handle_event(Event, State) ->
  NewState = update_state(Event, State),
  erlang:send(someworker, Event),
  {noreply, NewState}.
```



Forward the event.

persist

enki multimedia

- persist is synchronous and appended in order in the storage
- when persist happen, all others messages are postponed
- when persist happen the event is also notified to all subscribers.

persist: overview

enki multimedia

```
handle_info(do_snapshot, State) ->
  persist_proc:save_snapshot(State),
  {noreply, State}.
```

- snapshots are persisted to the snapshot storage for the last sequence

enki multimedia

```erlang
%% receive : <<"hello">>
handle_command(cast, Msg, State) ->
  persist_proc:persist(<< to_binary(Msg)/binary, "-1" >>),
  persist_proc:persist(<< to_binary(Msg)/binary, "-2" >>),
  {reply, ok, State}.


handle_event(Event, State) ->
  NewState = update_state(Event, State),
  persist_proc:persist(<<"inline-1">>),
  {noreply, State}.

%% receive
%% #{ seq := Seq1,
%%    data := <<< "hello-1">> }

%% receive
%% #{ seq := Seq2,
%%    data := <<< "hello-1">> }
%% receive
%% #{ seq := Seq3,
%%    data := <<"inline-1">> }
%% receive
%% #{ seq := Seq4,
%%    data :=<<"inline-1">> }
```

nested

persist.

enki multimedia

latest snapshot persisted

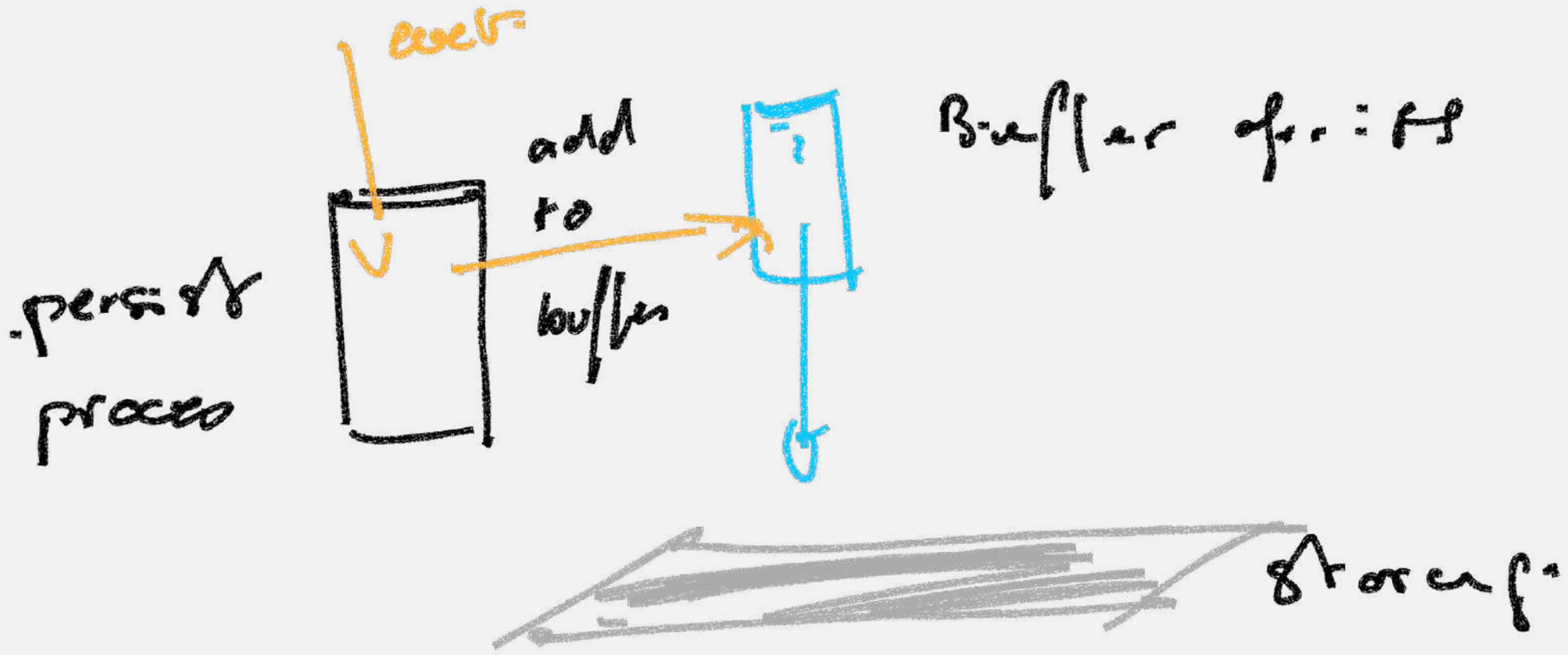
```
handle_recover({snapshot, Snapshot}, OldState) ->
  NewState = Snapshot,
  {noreply, NewState};
```

```
handle_recover({event, Event}, OldState) ->
  NewState = update_state(Event, OldState),
  {noreply, NewState}.
```

all events from the last snapshot (or 0.)

recover persisted state

enki multimedia



- for optimisation persist events are added to a batch before a write.
- atomic writes (persist_all function) are directly sent to the storage plugin

write is optimised

enki multimedia

- events adapters: migrate your events, transform your events from the disk
- journal and store plugin:
  - rocksdb
  - memory
- instrumentable using opencensus

miscellaneous

enki multimedia

query

enki
multimedia

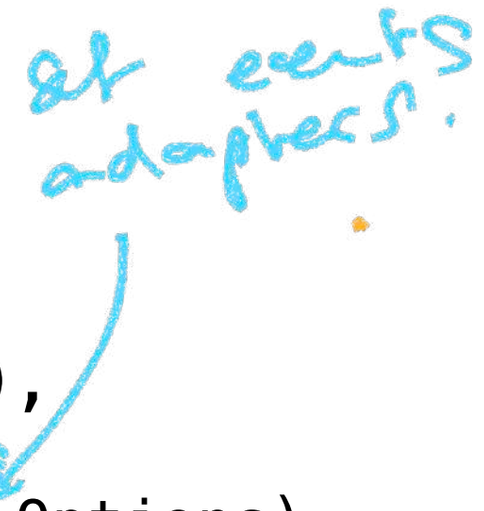
- 2 types of query
  - runtime events
  - replay the journals
  - depends on the journal plugin
- processing a query can be done concurrently.

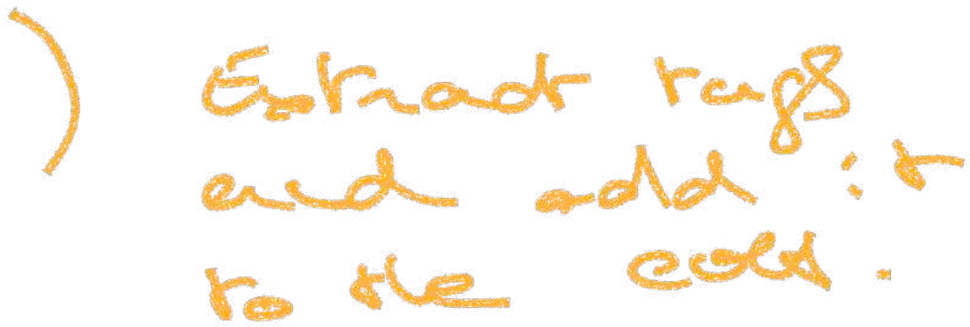
query

enki multimedia

```
start_link() ->
  PersistId = <<"test-1">>,
  erlang:send_after(1000, self(), do_snapshot),
  Options = [{events_adapers, [SomeModule]}]
  persist_proc:start_link(PersistId, ?MODULE, Options).


%% in the module
to_journal(Event) ->
  Tags = process_tags(Events),
  Events#{ tags => Tags }.
```





- the rocksdb plugin offers a way to tag the events and persist the result on write

- query_by_tag(PersistId, Tag, ...)

query

enki multimedia

ʔ

enki
multimedia

- opensource this month under Apache License 2
- todo:
  - documentation
  - more coverage
- access top the beta, drop me a mail: beta@enki-multimedia.eu
- any feedback on the current api, ideas are welcome.

query

enki multimedia

want to get a preview:
beta@enki-multimedia.eu

about me

- benoît chesneau
- craftsman working on P2P and custom data endpoints solution
- Owner of Enki Multimedia created 12 years ago
- Founding member of Erlang Ecosystem Foundation

enki multimedia