BUILDING AND INTEGRATING A DATA PLATFORM
Benoît Chesneau

Craftsman working on P2P and custom data endpoints solution

Enki Multimedia: the corporate interface

Member of the Erlang Industrial User Group

About me
1. Does my service do only one thing?
2. Is my service autonomous?
3. Does this service own its own data?

*a good micro-service?*
- isolated
- own its own data
- resilient
- communicate with other by asynchronous messages

micro-service
sharing data in the mobile age between and across micro-services make applications more scalable and resilient

- Ex: messaging systems,
- **standard solution**: client call a webservices to query and update the data

- **problem**: if connection is slow or absent the microservice stops

**sharing data**
- **local storage** replicated always available
- eventually consistent

-sharing data

cloud storage

synchronize

update and query

microservice
barrel

Bring and keep a view of your data near your application
- a database focusing on simplicity
- document oriented
- Automatic indexing

Focusing on simplicity
Docs are maps

```json
{
  "id": "someid",
  "Key": "value"
}
```
Access by path: /locations/country/Germany
- local first: bring and keep a view of your data near your application
- data is synchronised with other storages
- Replication to and from any sources
partial view

query
- library embedded in your Erlang application(*)
- available as a micro-service via HTTP(1,2) or via the Erlang distribution
- Peer to peer: a barrel is the unit
- Semantic to allow distributed transactions

(*) including elixir or lfe, or ....
- every peers fork the master, updates are offline
- peers pull and merge from the main server
- works well for back pressure (writes can be delayed)
- CRDT semantic for conflict-free data structures
- no vector clock
- revision tree

Body of leaf node in winning rev tree must be kept

Body of leaf node in losing rev trees must be kept
Alice

state

Iko

Bob

merge

d

c

operations

pull

rejected

pull

A

B

C

D

E

F

t0

A

B

C

D

t1
state

Alice

Iko

Bob

operations

t2

dc

pull

pull
Erlang
- Erlang is slow
- Erlang is only for communications protocols
- I should do it in Rust...
- No access to low level memory and file system APIs

Why not Erlang
Barrel is more a data orchestration service than a database

- Basic indexing
- Focus on replicating the data
- Nifs to help

Why Erlang
Doc: Revision + Metadata data:

- Read-Modify-Write: concurrency issue

- Incremental changes log: append only

- Indexes: when a new winning version is found the doc is indexed.

- Blobs (attachments)

What we write
- Provides connectors for other storages
- RocksDB for local persistent storage
  https://gitlab.com/barrel-db/erlang-rocksdb.git
- Dirty-nifs
- ETS?

Use the right tool for ...
Goal: anticipate the resource usage at the node level

Return early to the client

Control applied to all resources in the nodes

Back-pressure

let it bend: be resilient
- worker_pool
  https://github.com/inaka/worker_pool

- Hard to debug your program

- Little control on the pending requests

- Ecpxocxy but handle back-pressure the reverse way

Simple pooling
- Clients and Jobs should be handled independently
- Active and passive regulation
- Request unit: to set the number of requests we want to serve / seconds
- Flow-Based programming?
- sbroker, partially fit the bill: https://github.com/fishcakez/sbroker

Dynamic regulation
- Started with a simple “Single Writer Multiple Readers” pattern

- bottleneck: A process to handle the final write to the database

- We do and // most of the work out of the write process

- Indexes are processed asynchronously (but a session can read its own writes if needed)

Concurrency challenge
- Read access is shared via ETS
- On request a monitor to the db is created
When using the erlang distribution, events are dispatched by nodes, processes always subscribe locally.
- Erlang distribution is not used to share the data
- Erlang distribution can be switched
- HTTP transports

Transport the data
Roadmap
Milestones

- 1.0: 24 March 2018
- 1.1: 24 April 2018
- ...


1.0: Websockets support (with new hackney)

1.1: Experimental: GRPC
barrel is released in **march 2018**

[https://barrel-db.org](https://barrel-db.org)

contact me @benoitc