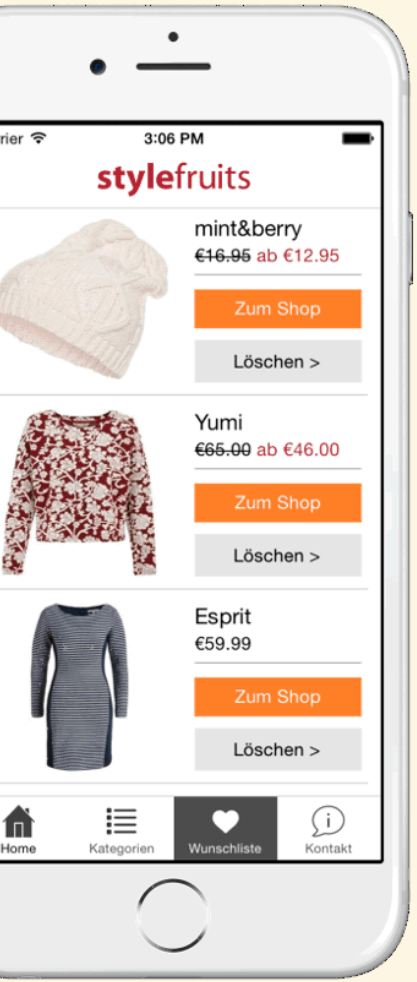
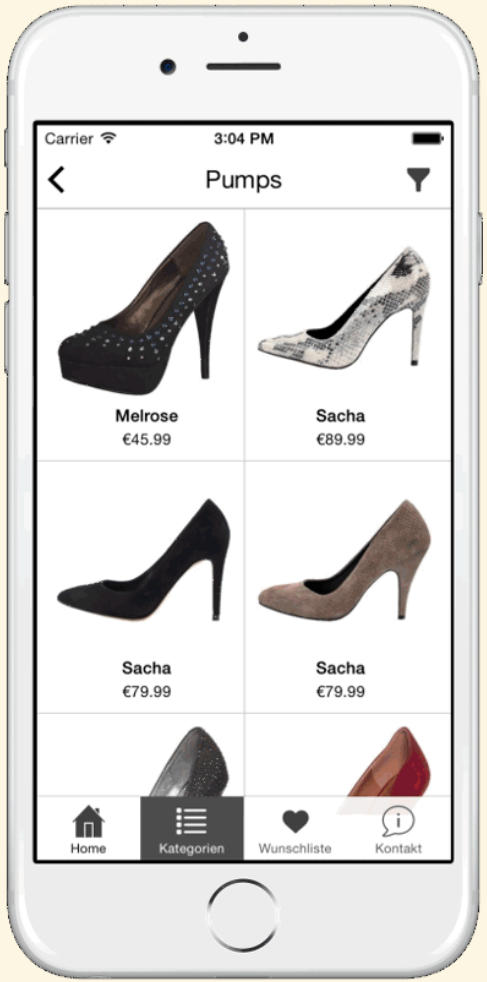
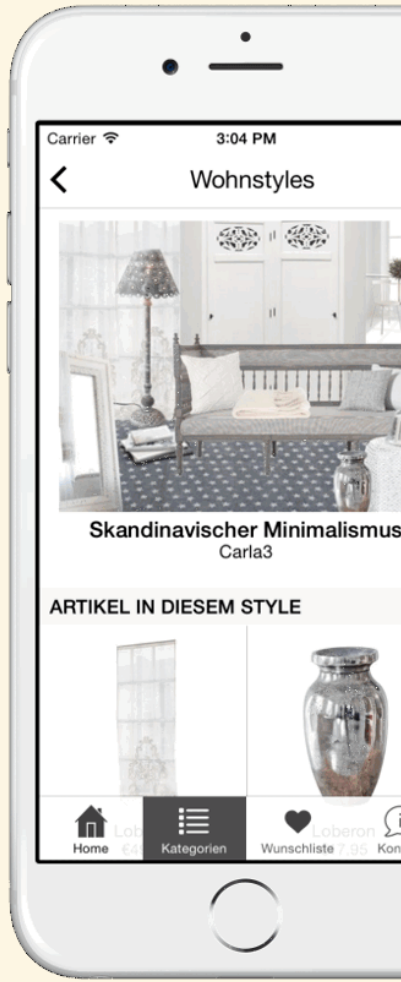
# Erlang in the Land of Lisp

with Jan Stępień @janstepien


rier
3:06 PM
stylefruits
mint&berry
€16.95 ab €12.95
Zum Shop
Löschen >
Yumi
€65.00 ab €46.00
Zum Shop
Löschen >
Esprit
€59.99
Zum Shop
Löschen >
Home
Kategorien
Wunschliste
Kontakt


Carrier
3:04 PM
Pumps
Melrose
€45.99
Sacha
€89.99
Sacha
€79.99
Sacha
€79.99
Home
Kategorien
Wunschliste
Kontakt


Carrier
3:04 PM
stylefruits
WOHNEN
HOME
DAMEN
In meinem kleinen Winterwunderl...
AlyshaLynn
Home
Kategorien
Wunschliste
Kontakt


Carrier
3:04 PM
Wohnstyles
Skandinavischer Minimalismus
Carla3
ARTIKEL IN DIESEM STYLE
Home
Kategorien
Wunschliste

api.stylefruits.de

↑ ↓ ↑↑↑ ↓↓↓

mobile client

GET /v1/give-me-all-i-need-right-now
Host: api.stylefruits.de
Accept: application/truckload-of-json

api.stylefruits.de

↑ ↓ ↑↑↑ ↓↓↓

batching proxy

↑ ↓

mobile client

HTTP client

JSON processing

HTTP server


# Requests in Batches

# Erlang

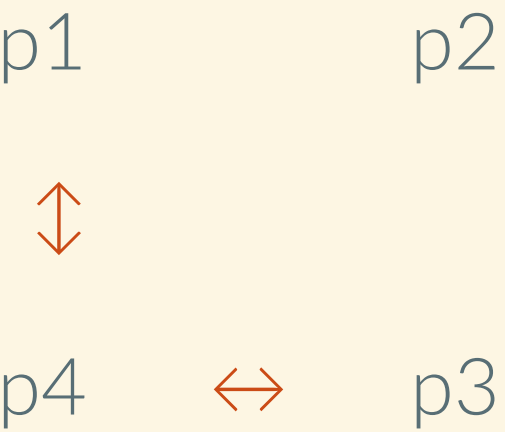
The syntax and the beauty beneath

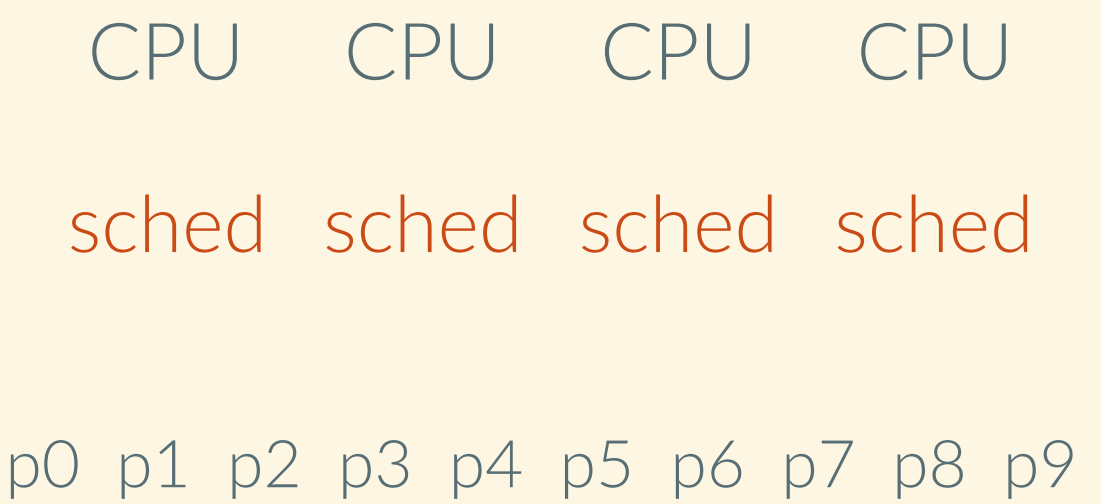
```
-module(qsort).
-export([qsort/1]).

qsort([]) -> [];
qsort([Pivot|Rest]) ->
    qsort([Front || Front <- Rest, Front < Pivot])
    ++ [Pivot] ++
    qsort([Back || Back <- Rest, Back >= Pivot]).
```

en.wikipedia.org/wiki/Erlang (programming language)

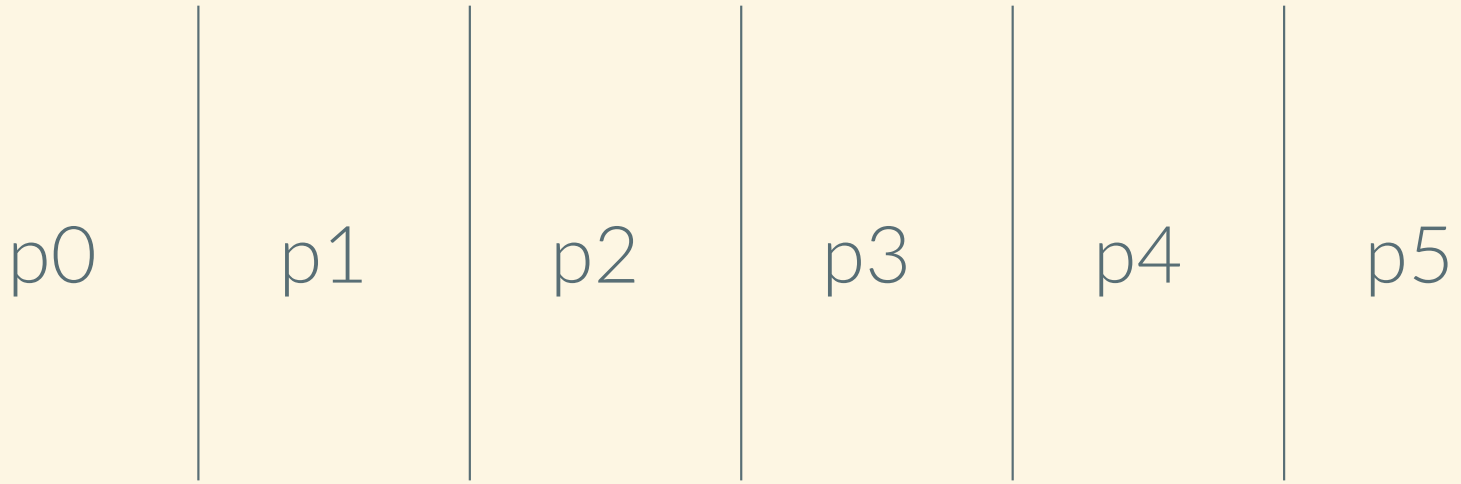
# The Erlang VM

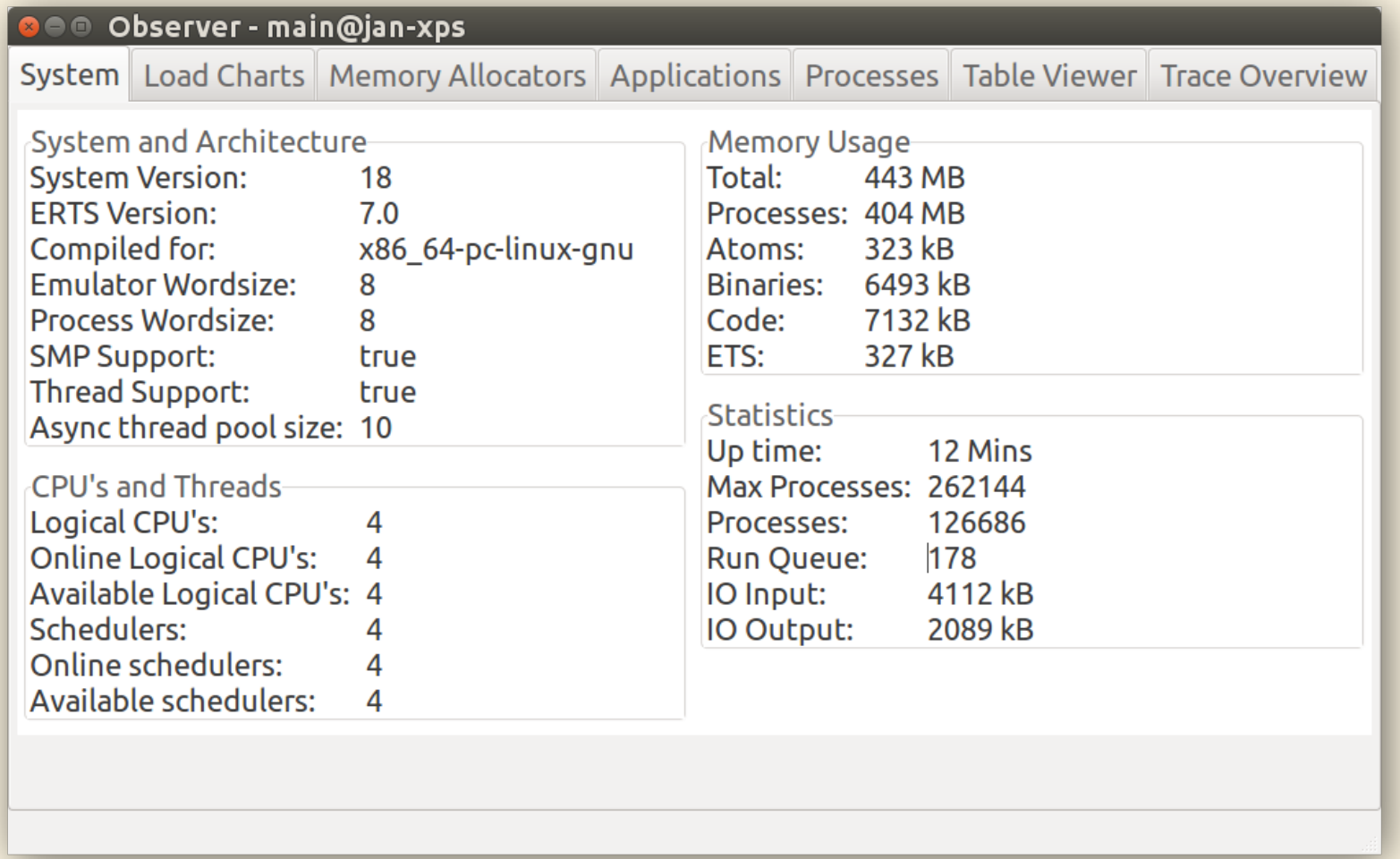
where processes dwell

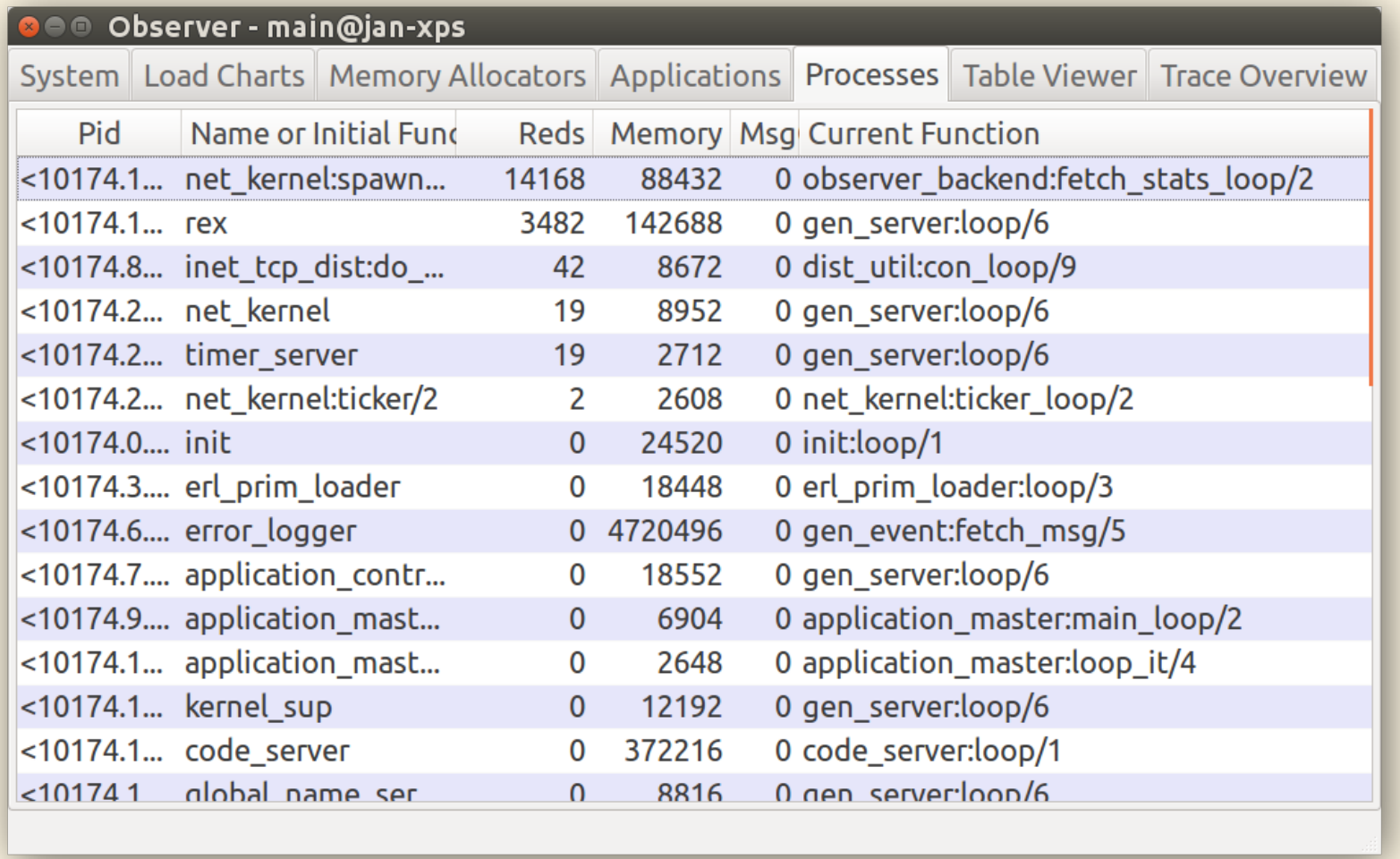


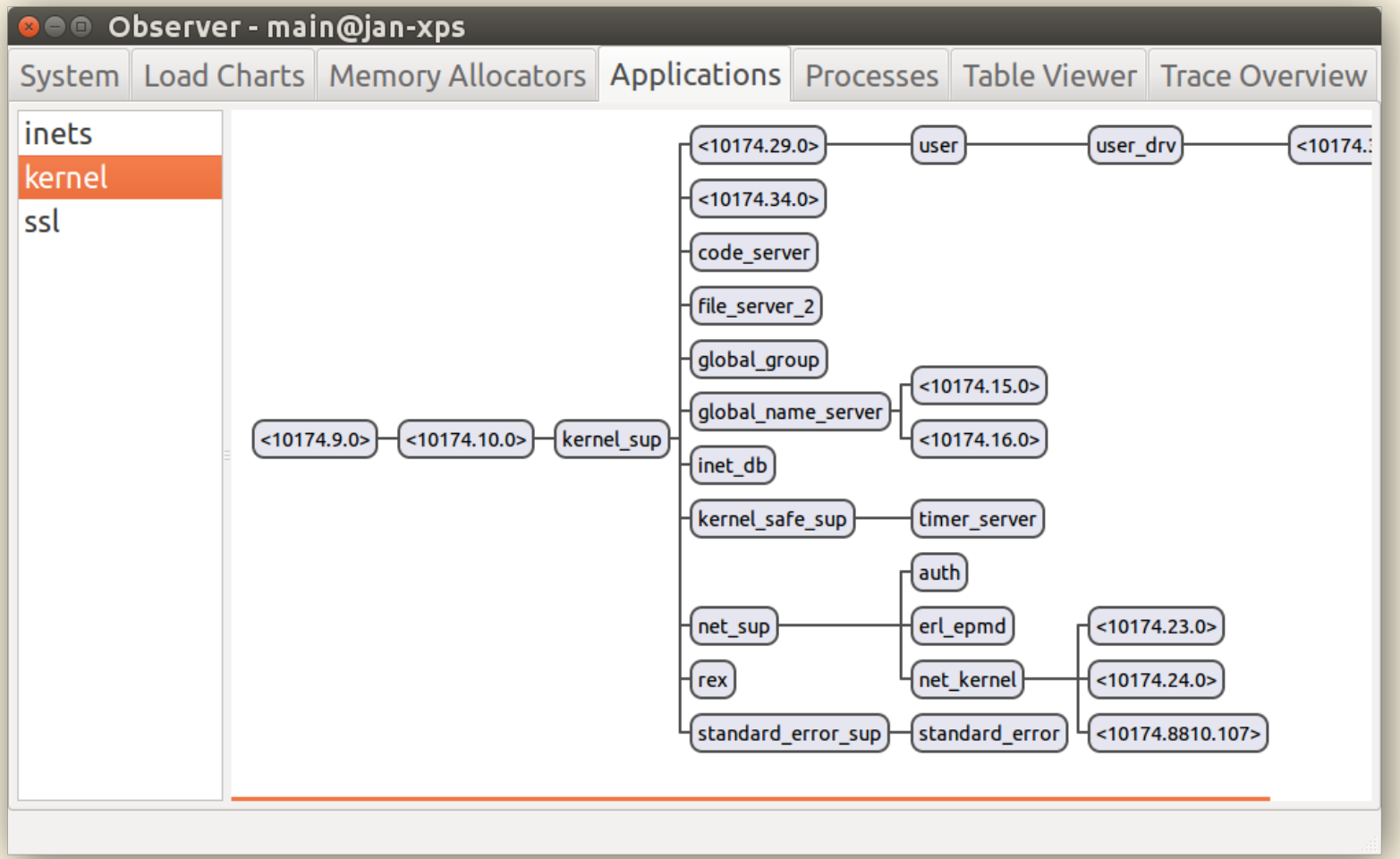

CPU CPU CPU CPU
sched sched sched sched
p0 p1 p2 p3 p4 p5 p6 p7 p8 p9

# Processes' heaps are separated




Observer - main@jan-xps
System
Load Charts
Memory Allocators
Applications
Processes
Table Viewer
Trace Overview
System and Architecture
System Version: 18
ERTS Version: 7.0
Compiled for: x86_64-pc-linux-gnu
Emulator Wordsize: 8
Process Wordsize: 8
SMP Support: true
Thread Support: true
Async thread pool size: 10
CPU's and Threads
Logical CPU's: 4
Online Logical CPU's: 4
Available Logical CPU's: 4
Schedulers: 4
Online schedulers: 4
Available schedulers: 4
Memory Usage
Total: 443 MB
Processes: 404 MB
Atoms: 323 kB
Binaries: 6493 kB
Code: 7132 kB
ETS: 327 kB
Statistics
Up time: 12 Mins
Max Processes: 262144
Processes: 126686
Run Queue: 178
IO Input: 4112 kB
IO Output: 2089 kB


Observer - main@jan-xps
System
Load Charts
Memory Allocators
Applications
Processes
Table Viewer
Trace Overview
Pid
Name or Initial Func
Reds
Memory
Msg
Current Function
<10174.1...
net_kernel:spawn...
14168
88432
0
observer_backend:fetch_stats_loop/2
<10174.1...
rex
3482
142688
0
gen_server:loop/6
<10174.8...
inet_tcp_dist:do_...
42
8672
0
dist_util:con_loop/9
<10174.2...
net_kernel
19
8952
0
gen_server:loop/6
<10174.2...
timer_server
19
2712
0
gen_server:loop/6
<10174.2...
net_kernel:ticker/2
2
2608
0
net_kernel:ticker_loop/2
<10174.0....
init
0
24520
0
init:loop/1
<10174.3....
erl_prim_loader
0
18448
0
erl_prim_loader:loop/3
<10174.6....
error_logger
0
4720496
0
gen_event:fetch_msg/5
<10174.7....
application_contr...
0
18552
0
gen_server:loop/6
<10174.9....
application_mast...
0
6904
0
application_master:main_loop/2
<10174.1...
application_mast...
0
2648
0
application_master:loop_it/4
<10174.1...
kernel_sup
0
12192
0
gen_server:loop/6
<10174.1...
code_server
0
372216
0
code_server:loop/1


Observer - main@jan-xps
System
Load Charts
Memory Allocators
Applications
Processes
Table Viewer
Trace Overview
inets
kernel
ssl
<10174.9.0>
<10174.10.0>
kernel_sup
<10174.29.0>
user
user_drv
<10174.34.0>
code_server
file_server_2
global_group
global_name_server
<10174.15.0>
<10174.16.0>
inet_db
kernel_safe_sup
timer_server
auth
net_sup
erl_epmd
<10174.23.0>
rex
net_kernel
<10174.24.0>
standard_error_sup
standard_error
<10174.8810.107>

# Back to Rib

## Setting the environment up

```
$ rebar create-app appid=rib
==> rib (create-app)
Writing src/rib.app.src
Writing src/rib_app.erl
Writing src/rib_sup.erl
```

```
%% rebar.config
{deps,
 [{jiffy, ".*",
   {git,
    "git://github.com/davisp/jiffy",
    {tag, "0.13.3"}}},
  {ejsonpath, ".*",
   {git,
    "git://github.com/rustyio/sync",
    "de3c42df58"}}]}.
```

```
$ rebar sh
==> rib (shell)
Erlang/OTP 18
Eshell V7.0  (abort with ^G)
1> sync:go().
Starting Sync (Automatic Code Compiler / Reloader)
Scanning source files...
ok
2>

=INFO REPORT==== 7-Feb-2016::12:21:23 ===
src/rib.erl:0: Recompiled.

=INFO REPORT==== 7-Feb-2016::12:21:23 ===
rib: Reloaded! (Beam changed.)
```

# Elli

and callback modules

```
-module(elli_minimal_callback).

-behaviour(elli_handler).

handle(Req, _Args) ->
  handle(Req#req.method, elli_request:path(Req), Req).

handle('GET',[<<"hello">>, <<"world">>], _Req) ->
  {200, [], <<"Hello World!">>};

handle(_, _, _Req) ->
  {404, [], <<"Not Found">>}.
```

github.com/knutin/elli

Let’s make it all concurrent

```
pmap(Fun, List) ->
  Parent = self(),
  Workers =
    [spawn(fun() ->
              Parent ! {self(), Fun(X)}
           end)
     || X <- List],
  [receive
     {Worker, Value} -> Value
   end
   || Worker <- Workers].
```

```
pmap(Fun, List) ->
  Parent = self(),
  Workers =
    [spawn_link(fun() ->
                    Parent ! {self(), Fun(X)}
                end)
     || X <- List],
  [receive
     {Worker, Value} -> Value
   end
   || Worker <- Workers].
```

```
pmap(Fun, List) ->
  Parent = self(),
  Workers =
    [spawn_link(fun() ->
                    Parent ! {self(), Fun(X)}
                end)
     || X <- List],
  [receive
     {Worker, Value} -> Value
   after
     1000 -> error(timeout)
   end
   || Worker <- Workers].
```

You shall not

# OTP

and generic behaviours in particular

# Use case

HTTP connection killer

```erlang
-module(rib_conn_killer).
-export([start_link/0]).

start_link() ->
  {ok, spawn_link(fun go/0)}.

go() ->
  {ok, Url} = application:get_env(rib, backend),
  Headers = [{"connection", "close"}],
  {ok, _} = httpc:request(get, {Url, Headers}),
  ok = timer:sleep(30000),
  go().
```

```erlang
-module(rib_conn_killer_sup).
-behaviour(supervisor).
-export([start_link/0, init/1]).

start_link() ->
  supervisor:start_link({local, ?MODULE},
                        ?MODULE,
                        []).

init([]) ->
  {ok, {{one_for_one, 5, 10},
        [{rib_conn_killer, ...}]}}.
```

# Use case

## Request limiter

```
-module(rib_limiter).
-behaviour(gen_server).

start_link(Opts) ->
  gen_server:start_link(?MODULE, Opts, []).

subtract(ServerRef, N) ->
  gen_server:cast(ServerRef, {subtract, N}).

init({max, N}) ->
  {ok, N}.

handle_cast({subtract, N}, State) ->
  NewState = State - N,
  case NewState >= 0 of
    true -> {noreply, NewState};
    false -> {stop, limit_exceeded, NewState}
  end.
```

# Deployment

## Lightweight Docker images

# Something completely different

Happy path programming, letting it crash

# On shoulders of giants

Virtual machine, OTP

# Needs more research

Polymorphism, editor integration

# Can’t stand the syntax?

LFE, Joxa, Elixir

github.com/stylefruits/rib

# Erlang in the Land of Lisp

with Jan Stępień @janstepien