

The Platform of the System



Code BEAM 

Robert Virding  @rvirding
Mariano Guerra  @warianoguerra

Chesterton's Fence





Programming Language

Vs

Programming System

Interacting with a Compiler

Vs

Interacting with a System

Operating Systems

"An operating system is a collection of things that don't fit into a language. There shouldn't be one."

-- Dan Ingalls


Unix + Shell + Emacs

Plan 9 / Oberon

System

A system is a group of **interacting** or **interrelated** elements that act according to a set of rules to form a **unified** whole.

A system, surrounded and influenced by its **environment**, is described by its **boundaries**, **structure** and **purpose** and expressed in its functioning.

CPU[

2.0%]

Tasks: 16 total, 1 running

Mem[

13/123MB]

Load average: 0.37 0.12 0.04

Swp[

0/109MB]

Uptime: 00:00:50

PID	USER	PRI	NI	UIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
3692	per	15	0	2424	1204	980	R	2.0	1.0	0:00.24	htop
1	root	16	0	2952	1852	532	S	0.0	1.5	0:00.77	/sbin/init
2236	root	20	-4	2316	728	472	S	0.0	0.6	0:01.06	/sbin/udevd --daem
3224	dhcp	18	-2	2412	552	244	S	0.0	0.4	0:00.00	dhclient3 -e IF_ME
3488	root	18	0	1692	516	448	S	0.0	0.4	0:00.00	/sbin/getty 38400
3491	root	18	0	1696	520	448	S	0.0	0.4	0:00.01	/sbin/getty 38400
3497	root	18	0	1696	516	448	S	0.0	0.4	0:00.00	/sbin/getty 38400
3500	root	18	0	1692	516	448	S	0.0	0.4	0:00.00	/sbin/getty 38400
3501	root	16	0	2772	1196	936	S	0.0	0.9	0:00.04	/bin/login --
3504	root	18	0	1696	516	448	S	0.0	0.4	0:00.00	/sbin/getty 38400
3539	syslog	15	0	1916	704	564	S	0.0	0.6	0:00.12	/sbin/syslogd -u s
3561	root	18	0	1840	536	444	S	0.0	0.4	0:00.79	/bin/dd bs 1 if /p
3563	klog	18	0	2472	1376	408	S	0.0	1.1	0:00.37	/sbin/klogd -P /va
3590	daemon	25	0	1960	428	308	S	0.0	0.3	0:00.00	/usr/sbin/atd
3604	root	18	0	2336	792	632	S	0.0	0.6	0:00.00	/usr/sbin/cron
3645	per	15	0	5524	2924	1428	S	0.0	2.3	0:00.45	-bash

F1Help F2Setup F3SearchF4InvertF5Tree F6SortByF7Nice -F8Nice +F9Kill F10Quit

```

Home(H) | Network(N) | System(S) | Ets(E) | App(A) | Doc(D) | Plugin(P) | recon:proc_count(memory, 35) Interval:1500ms | 0Days 0:1:19
Erlang/OTP 22 [erts-10.6] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-threads:1] [hipe] [dtrace]

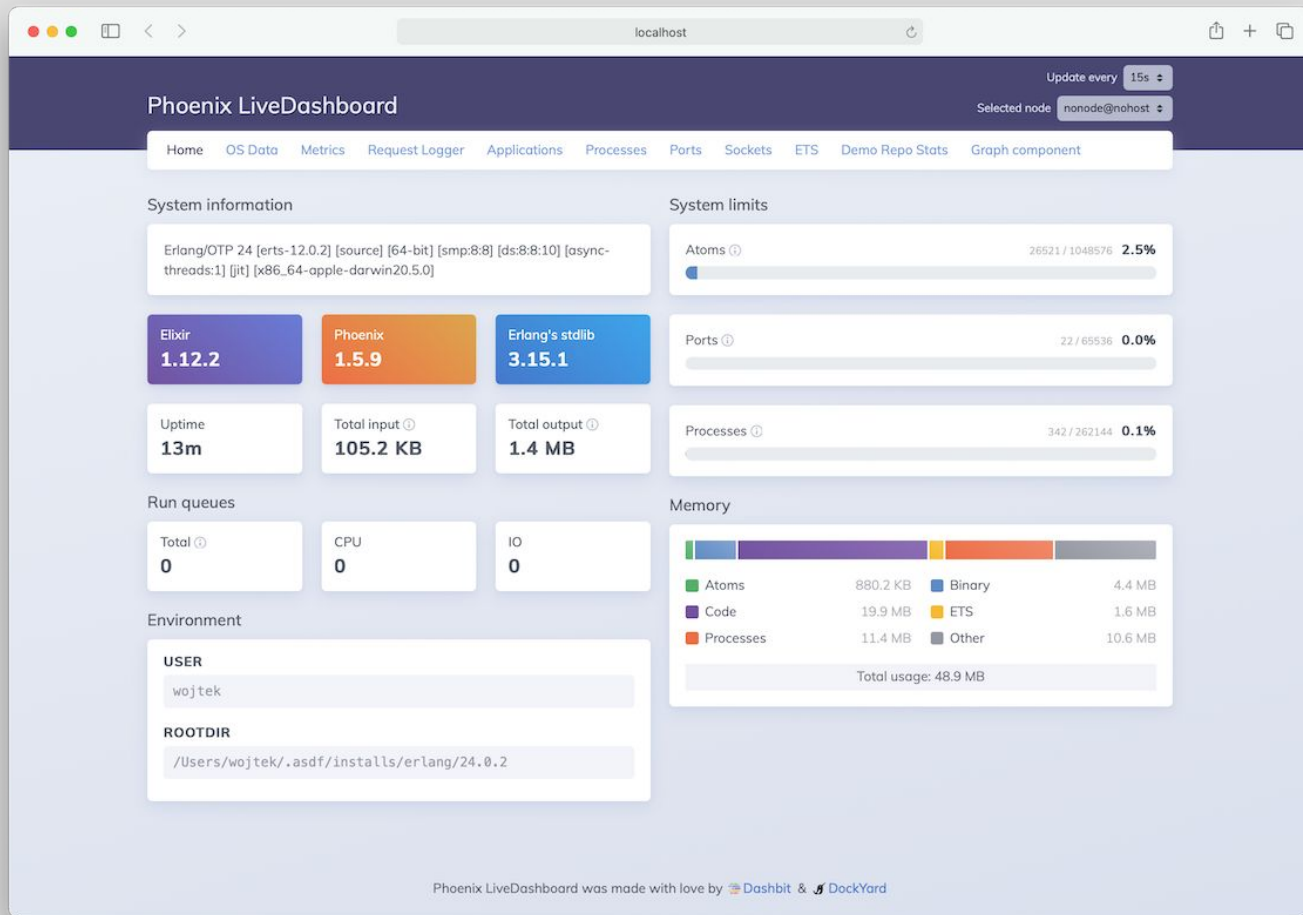
System      Count/Limit      System      Status      Stat Info      Size
Proc Count  74/262144         Version     22.2        Active Task     1
Port Count  4/65536           ps -o pcpu  0.4%        Context Switch  153987
Atom Count  17961/1048576    ps -o pmem  0.5%        Reds(Total/SinceLast 46553122/95492
Mem Type     Size              Mem Type     Size         IO/GC:(1500ms)  Total/Increments
Total       35.5375 MB       Binary      1.0249 MB   02.88%         IO Output      364.3145 KB/7.2813 KB
Process     15.9133 MB       Code        7.7241 MB   21.74%         IO Input       8.8525 KB/24 B
Atom        481.6416 KB      Port Parallelism (+spp) false        Gc Count       6988/14
Ets         1.1406 MB        RunQueue    0           Gc Words Reclaimed 74880677/277557

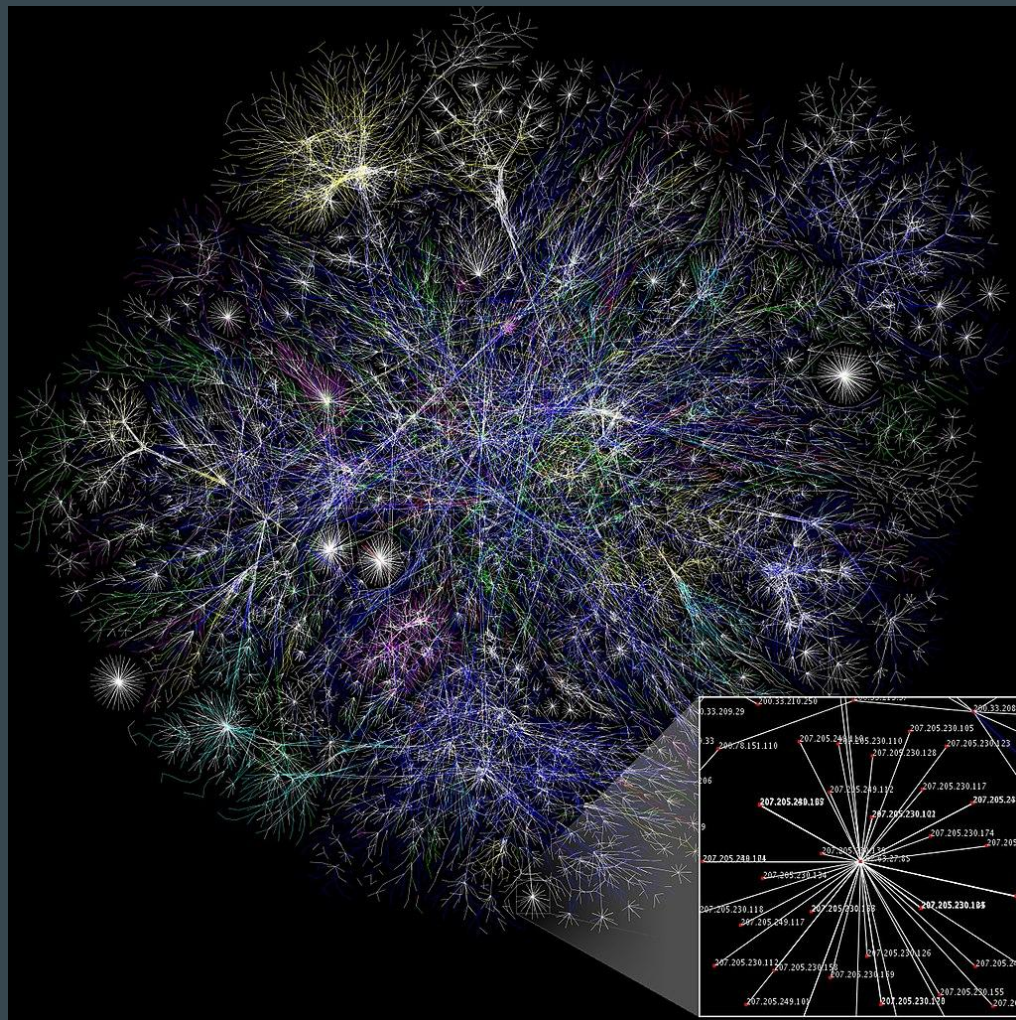
1 00.33% | 3 00.00% | 5 00.00% | 7 00.00%
2 00.01% | 4 00.00% | 6 00.01% | 8 00.00%

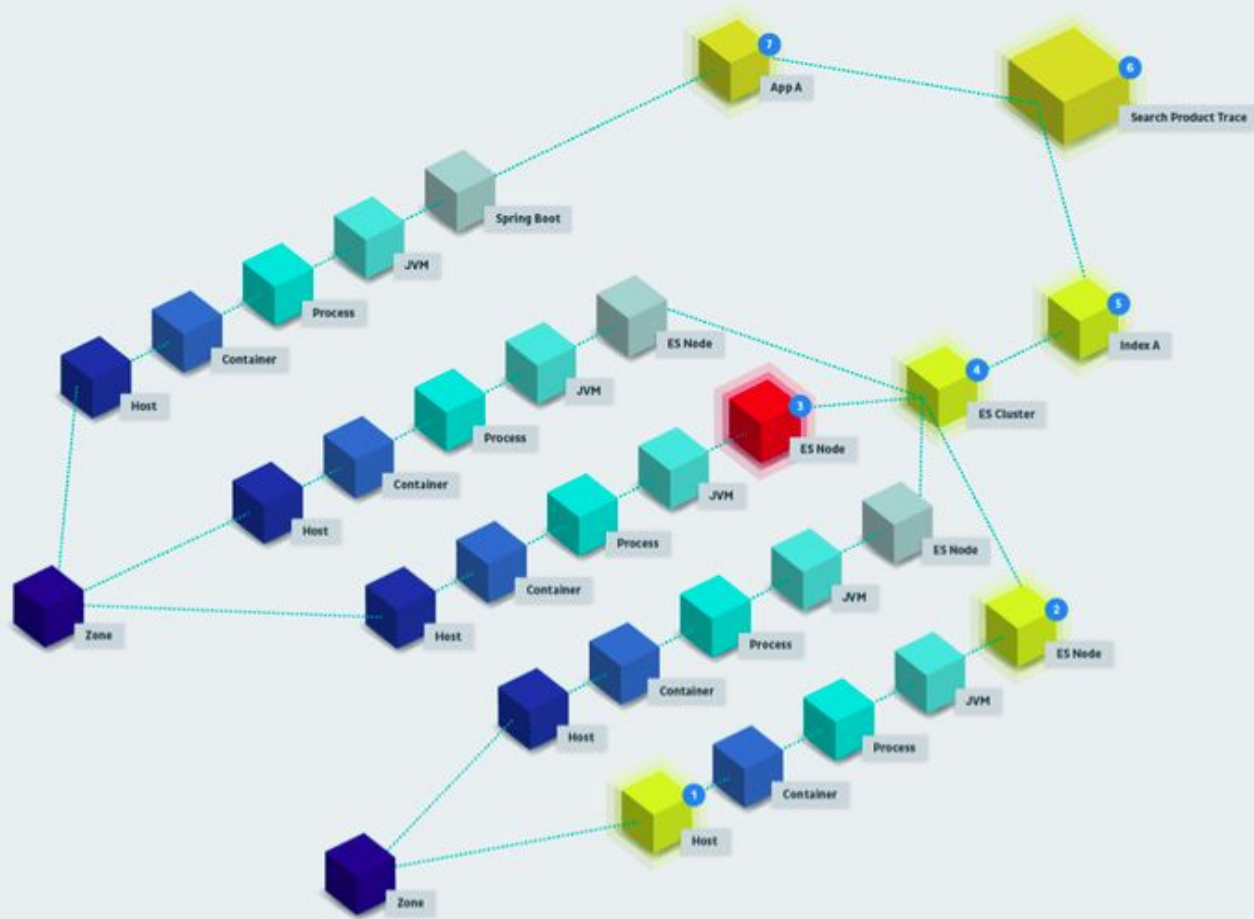
No  Pid      Memory      Name or Initial Call      Reductions  MsgQueue  Current Function
1 <0.378.0>  9.1280 MB  group:server/3            1948386  0  group:more_data/6
2 <0.9.0>    1.0635 MB  erl_prim_loader           8316160  0  erl_prim_loader:loop/3
3 <0.379.0>  588.0820 KB disk_log:init/2           27655  0  disk_log:loop/1
4 <0.43.0>  449.4531 KB application_controller    113080  0  gen_server:loop/7
5 <0.381.0>  225.4375 KB erlang:apply/2            17830  0  shell:shell_rep/4
6 <0.49.0>  172.1758 KB code_server               288594  0  code_server:loop/1
7 <0.8.0>   143.1406 KB rebar_agent              1208920  0  erlang:hibernate/3
8 <0.48.0>  91.8477 KB kernel_sup                2745  0  gen_server:loop/7
9 <0.57.0>  41.4492 KB file_server_2            31102  0  gen_server:loop/7
10 <0.390.0> 30.1836 KB erlang:apply/2           468  0  observer_cli_store:loop/1
11 <0.0.0>   21.1953 KB init                     4237  0  init:loop/1
12 <0.376.0> 18.2930 KB user_drv                 35744  0  user_drv:server_loop/6
13 <0.80.0>  13.3906 KB application_master:start_it/4 320  0  application_master:loop_it/4
14 <0.66.0>  11.7773 KB logger_sup               747  0  gen_server:loop/7
15 <0.41.0>  11.6914 KB logger                   872  0  gen_server:loop/7
16 <0.82.0>  10.6758 KB logger_std_h_ssl_handler 236  0  gen_server:loop/7
17 <0.103.0> 10.5742 KB httpc_profile_sup        498  0  gen_server:loop/7
18 <0.385.0>  8.9258 KB logger_std_h_default     232  0  gen_server:loop/7
19 <0.122.0>  8.7891 KB disk_log_server          348  0  gen_server:loop/7
20 <0.85.0>  8.7891 KB ssl_admin_sup            455  0  gen_server:loop/7
21 <0.107.0>  8.7461 KB httpc_rebar              324  0  gen_server:loop/7
22 <0.65.0>  6.9609 KB kernel_safe_sup          480  0  gen_server:loop/7
23 <0.382.0>  6.9492 KB erlang:apply/2          1447  0  io:execute_request/2
24 <0.92.0>  6.9180 KB tls_server_sup           337  0  gen_server:loop/7
25 <0.121.0>  6.8750 KB disk_log_sup             335  0  gen_server:loop/7
26 <0.110.0>  5.8438 KB inet_gethost_native      289  0  inet_gethost_native:main_loop/1
27 <0.102.0>  5.8008 KB httpc_sup                334  0  gen_server:loop/7
28 <0.101.0>  5.8008 KB inet_sup                 347  0  gen_server:loop/7
29 <0.95.0>  5.8008 KB dtls_sup                 329  0  gen_server:loop/7
30 <0.90.0>  5.8008 KB tls_sup                  332  0  gen_server:loop/7
31 <0.89.0>  5.8008 KB ssl_connection_sup        331  0  gen_server:loop/7
32 <0.84.0>  5.8008 KB ssl_sup                  331  0  gen_server:loop/7
33 <0.46.0>  5.6719 KB application_master:start_it/4 436  0  application_master:loop_it/4
34 <0.68.0>  3.9805 KB logger_proxy             148  0  gen_server:loop/7
35 <0.51.0>  3.9805 KB inet_db                  370  0  gen_server:loop/7

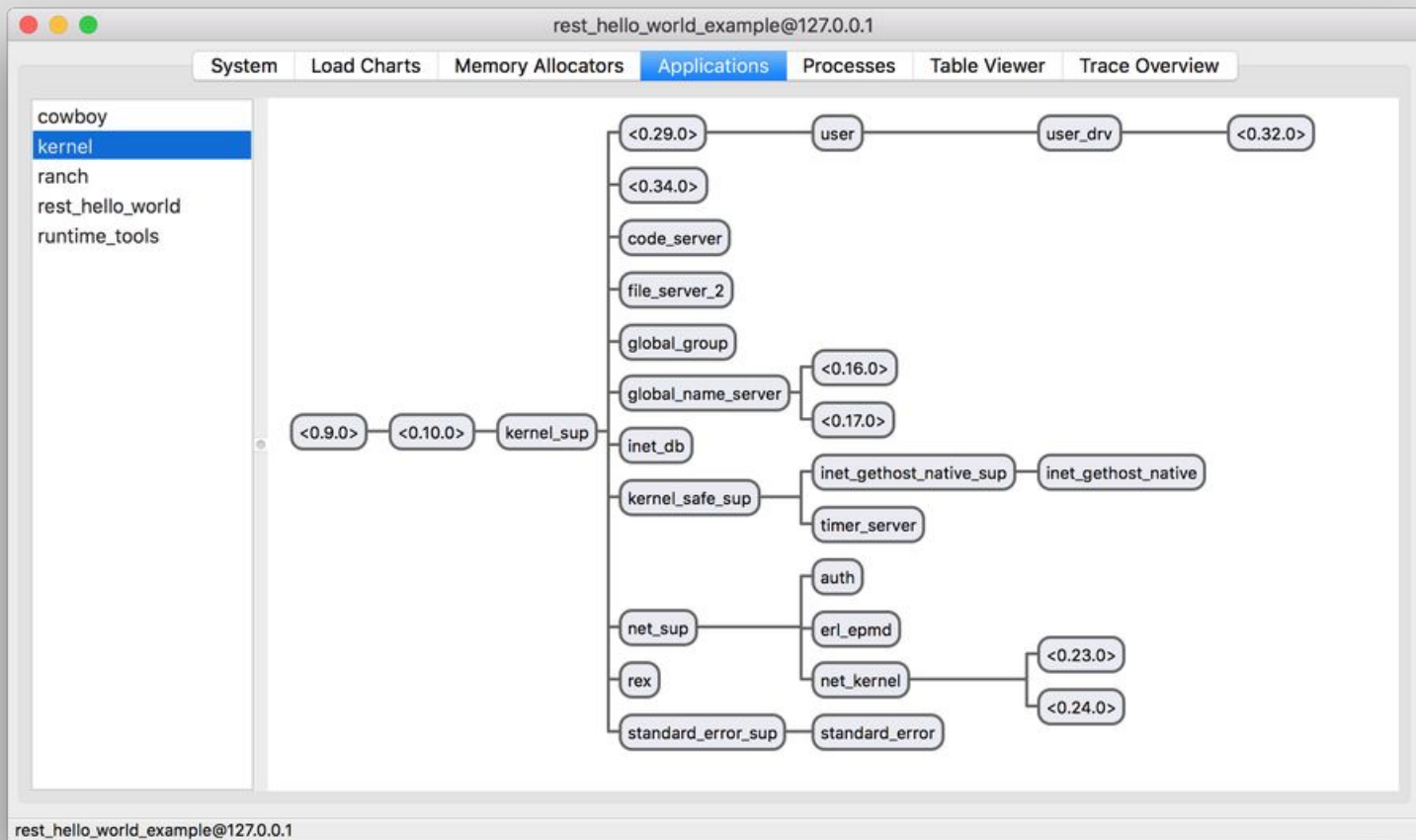
q(quit) p(pause) r(rr(reduction) m/mm(mem) b/bb(binary mem) t/tt(total heap size) mq/mmq(msg queue) 9(proc 9 info) F/B(page forward/back)

```









Programming Systems

Lisp Machines

Display Debugger on Peek

Abort	Exit	Edit function	Breakpoints
Proceed	Switch windows	Find Frame	Monitor
Return	Help	Backtrace	Exit traps
Reinvoke	Bug Report	Source code	Call traps

```
Proceed without any special action
Allow process to continue
Return to Lisp Top Level in Dynamic Lisp Listener 1
Restart process Dynamic Lisp Listener 1
```

```

Backtrace
PROCESS:PROCESS-BL
PROCESS:PROCESS-WA
⇒ SCL:PROCESS-WAIT
TV:AWAIT-WINDOW-EX
SI:COM-SELECT-ACTI
CP::COMMAND-LOOP-E
(:PROPERTY SYS:DTP
PROCESS::WITH-DELA
TV:WITH-NOTIFICATI
PROCESS::WITH-PROC
SI:LISP-COMMAND-LO
SI:LISP-COMMAND-LO
SI:LISP-TOP-LEVEL

```

```

Backtrace
PROCESS:PROCESS-BL
PROCESS:PROCESS-WA
⇒ SCL:PROCESS-WAIT
TV:AWAIT-WINDOW-EX
SI:COM-SELECT-ACTI
CP::COMMAND-LOOP-E
(:PROPERTY SYS:DTP
PROCESS::WITH-DELA
TV:WITH-NOTIFICATI
PROCESS::WITH-PROC
SI:LISP-COMMAND-LO
SI:LISP-COMMAND-LO
SI:LISP-TOP-LEVEL

```

Operation on SI:LISP-TOP-LEVEL1:

- Clear trap-on-exit for this frame
- Disassemble the function for this frame
- Edit this frame's function
- Reinvoke this frame
- Return from this frame
- Set the current frame
- Set the current frame (detailed)
- Set trap-on-exit for this frame
- Show arguments with which frame was called
- Show this function's argument list
- Marking and yanking menu
- Presentation debugging menu
- System menu
- Window operation menu

Operation on SI:LISP-TOP-LEVEL1:

- Clear trap-on-exit for this frame
- Disassemble the function for this frame
- Edit this frame's function
- Reinvoke this frame
- Return from this frame
- Set the current frame
- Set the current frame (detailed)
- Set trap-on-exit for this frame
- Show arguments with which frame was called
- Show this function's argument list
- Marking and yanking menu
- Presentation debugging menu
- System menu
- Window operation menu

```

SCL:PROCESS-WAIT
0 ENTRY: 2 REQUIRED, 0 OPTIONAL, 5 ARC ;Creating PROCESS::WHOSTATE
2 PUSH NIL ;Creating PROCESS::ARGUMENTS
4 START-CALL-INDIRECT #'PROCESS:PROCESS-WAIT
6 PUSH FP|2 ;PROCESS::WHOSTATE
10 PUSH-INDIRECT #'PROCESS:VERIFY-FUNCTION
7 PUSH FP|3 ;LISP:FUNCTION
12 PUSH LP|0 ;PROCESS::ARGUMENTS
13 FINISH-CALL-APPLY-4-RETURN

```

```

SCL:PROCESS-WAIT
0  ENTRY: 2 REQUIRED, 0 OPTIONAL, 5 ARC ;Creating PROCESS::WHOSTATE
2  PUSH NIL ;Creating PROCESS::ARGUMENTS
4  START-CALL-INDIRECT #'PROCESS:PROCESS-WAIT
6  PUSH FP|2 ;PROCESS::WHOSTATE
10 PUSH-INDIRECT #'PROCESS:VERIFY-FUNCTION
7  PUSH FP|3 ;LISP:FUNCTION
12 PUSH LP|0 ;PROCESS::ARGUMENTS
13 FINISH-CALL-APPLY-4-RETURN

```

Inspect history

Break:
The current frame is
s-A, **RESUME**: Proc
s-B, **ABORT**: Allow
s-C: Return
s-D: Resta
+ Set Current Frame
+ |

Break:
The current frame is
s-A, **RESUME**: Proc
s-B, **ABORT**: Allow
s-C: Return
s-D: Resta
+ Set Current Frame
+ |

```
Arguments, locals, and specials
Arg 0 (PROCESS::WHOSTATE): "Await Exposure"
Arg 1 (LISP:FUNCTION): (#<Compiled function TV:SHEET-EXPOSED-P 20051107372>)
Rest Arg: (#<DYNAMIC-LISP-LISTENER Dynamic Lisp Listener 1 20006402557 deexp
Local 3 (PROCESS::ARGUMENTS): (#<DYNAMIC-LISP-LISTENER Dynamic Lisp Listener
```

```
Arguments, locals, and specials
Arg 0 (PROCESS::WHOSTATE): "Await Exposure"
Arg 1 (LISP:FUNCTION): (#<Compiled function TV:SHEET-EXPOSED-P 20051107372>)
Rest Arg: (#<DYNAMIC-LISP-LISTENER Dynamic Lisp Listener 1 20006402557 deexp
Local 3 (PROCESS::ARGUMENTS): (#<DYNAMIC-LISP-LISTENER Dynamic Lisp Listener
```

sh-**Mouse-L, -M, -R**: Set trap-on-exit for this frame.
To see other commands, press Shift, Control, Meta-Shift, or Super.

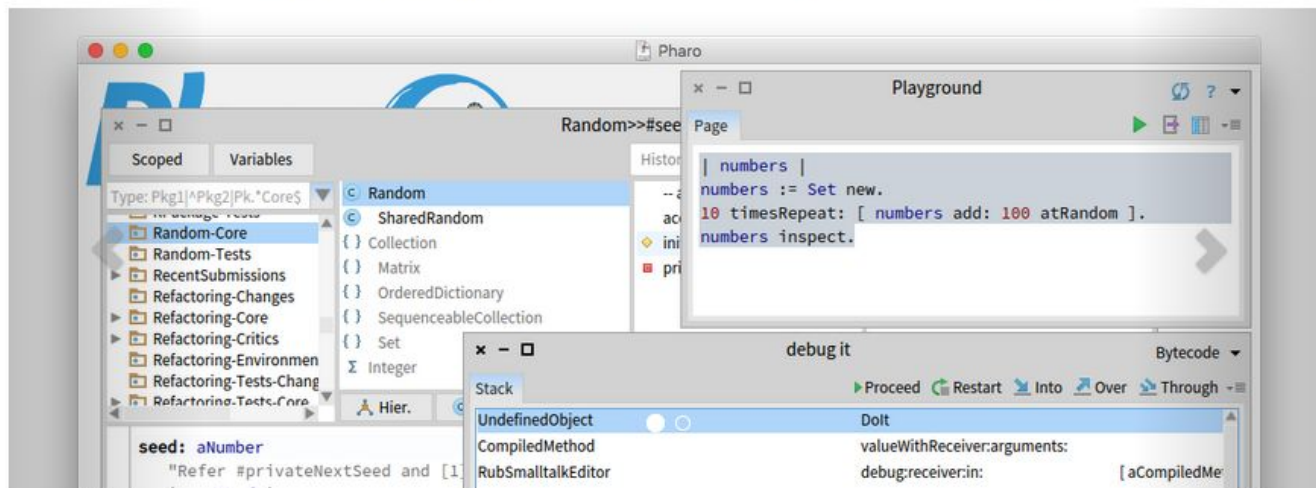
Smalltalk

gtoolkit.com



The immersive programming experience

Pharo is a pure object-oriented programming language *and* a powerful environment, focused on simplicity and immediate feedback (think IDE and OS rolled into one).



Erlang

Designed to build a specific type
of system

Telecom systems

- Concurrency
- Fault tolerance
- Scalability/adaptability

View Erlang as a SYSTEM
with a language, not a
language with concurrency

The Erlang Language

The result!

- Processes
 - Asynchronous messages, selective receive
 - Primitives for error detection and control
 - Primitives for managing and loading code
 - No mutable, shared, global data
 - Functional language - originally logical
 - No user defined datatypes!
-

The BEAM

Designed to run Erlang!

- Very close coupling between Erlang and the BEAM
- Directly implements most language features (all of the above)
- Provides communication with outside world - ports and NIFs

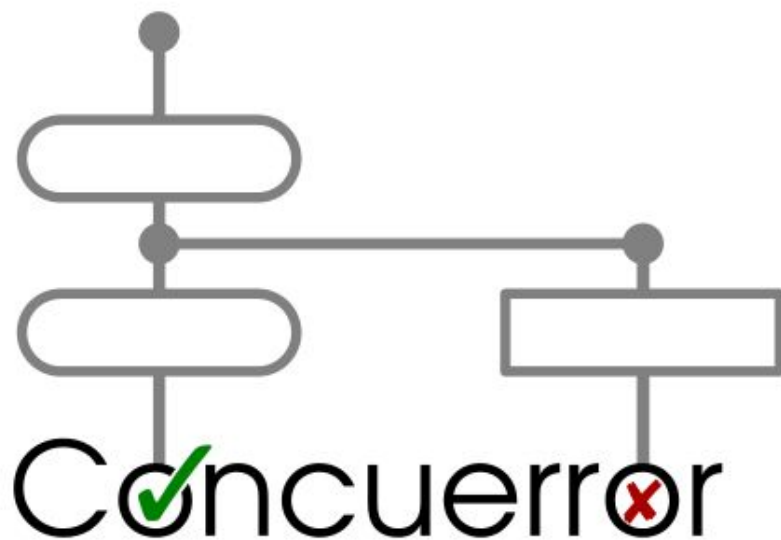
The BEAM System

- Tracing
 - Introspection
 - Hot Code Loading
 - External Format
-
- Scheduling
 - Networking
 - Remote Shell
 - Preemptive Multitasking
-

Languages on the BEAM

- **Native Languages**
(Erlang semantics)
Elixir, LFE, Caramel, ...
 - “Skins” on top of Erlang
 - **Non-Native Languages**
(different semantics)
Luerl (Lua on the BEAM)
 - Implemented in Erlang
 - Run external to the BEAM
-

```
get_file(Host, Path) ->  
    {ok, Bin} = rpc:call(Host, file, read_file, [Path]),  
    file:write_file(Path ++ ".copy", [Bin]).
```



Concuerror is a **stateless model checking tool** for **Erlang** programs. It can be used to **detect** and **debug** concurrency errors, such as **deadlocks** and errors due to **race conditions**. Moreover it can **verify** the absence of such errors, because it tests programs **systematically**, unlike techniques based on randomness.



Partisan

High-Performance Distributed Erlang

 Star 625

Partisan

Partisan is the design of an alternative runtime system for improved scalability and reduced latency in actor applications.

Partisan provides:

- **Better scalability** by leveraging different network topologies for communication
- **Reduced latency** through efficient parallel message scheduling for actor-to-actor communication

Partisan is provided as a user library in Erlang and achieves up to an order of magnitude increase in the number of nodes the system can scale to through runtime overlay selection, up to a 34.96x increase in throughput, and up to a 13.4x reduction in latency over Distributed Erlang.

Scaling Erlang distribution: going beyond the fully connected mesh

Authors:  [Adam Lindberg](#),  [Sébastien Merle](#),  [Peer Stritzinger](#) [Authors Info & Affiliations](#)

Erlang 2019: Proceedings of the 18th ACM SIGPLAN International Workshop on Erlang • August 2019 • Pages 48–55 • <https://doi.org/10.1145/3331542.3342572>

ABSTRACT

Distributed Erlang, the process of transparently running Erlang programs over networks, has a long history of immense usefulness but has problems when distributed systems reach certain scales. We explain the issues and show research done towards the goal of transparently enhancing Erlang distribution, so that changes to existing applications and systems can be avoided. We propose several research directions together with prototype implementations that all serve the purpose of improving the current status quo. This includes using different transport protocols, generalizing implementation efforts and incorporating routing protocols for more dynamic node constellations. We then describe some background and history of various work to solve Erlang distribution scalability issues. We show that there is much room for improvement on the Erlang distribution layer without breaking abstractions that developers are used to and rely on.

erlyberly-1462009440221@mac connected to dev1@127.0.0.1

Show Processes Hide Modules Refresh Modules Erlang Memory xref Analysis Disconnect Preferences Crash Reports

Processes @ 10:44:03.32 977

Filter functions i.e. gen_s:call or #t for all traces
qry_com:

- riak_kv_qry_compiler
 - acc_lower_bounds/2
 - acc_upper_bounds/2
 - add_types2/3
 - add_types_to_filter/2
 - break_out_timeseries/3
 - check_if_timeseries/2
 - col_index_and_type_of/2
 - compile/3
 - compile_select_clause/2
 - compile_select_col/2
 - compile_select_col_stateless/2
 - compile_select_col_stateless2/3
 - compile_where/2
 - compile_where_clause/3
 - expand_query/4
 - expand_where/3
 - extract_options/1
 - extract_stateful_functions/2
 - extract_stateful_functions2/3
 - finalise_aggregate/2
 - finalise_aggregate2/3
 - find_quantum_field_index_in_key/

Traces <0.24802.5> riak_kv_qry_compiler:my_mapfold/3 x

Function arguments

```
#Fun<riak_kv_qry_compiler.1.55826920>
[ ]
[
  {
    identifier
    [
      <<"*">>
    ]
  }
]
```

Result

```
{
  [
    [
      sint64
      timestamp
    ]
  ]
}
```

Stack Trace (13)

- riak_kv_qry_compiler:compile_select_clause/2
- unknown function
- riak_kv_qry_compiler:compile_select_clause/2
- riak_kv_qry_compiler:compile/3
- unknown function
- riak_kv_qry_compiler:compile/3
- riak_kv_qry:do_select/2
- riak_kv_ts_svc:sub_tsqueryreq/4

erlyberly-1462099934987@mac connected to e@mac

Processes @ 11:52:15.208 37

Hide Processes Show Modules Refresh Modules Erlang Memory xref Analysis Disconnect Preferences Crash Reports

Filter on process pid and registered name Refresh Start Polling

Pid	Name	Reductions	M. Queue	Heap Siz...	Stack Si.
<0.25.0>	code_server	195169	0	141848	24
<0.3.0>	erl_prim_loader	166812	0	20688	48
<0.11.0>	kernel_sup	48198	0	33480	72
<0.12.0>	rex	42641	0	972288	72
<0.143.0>		11453	0	12784	80
<0.20.0>	net_kernel	8085	0	7896	72
<0.0.0>	init	4771	0	20688	16
<0.32.0>		4146	0	141848	128
<0.21.0>		2435	0	4880	72
<0.138.0>	erlyberly_tcollector	2078	0	229520	64
<0.139.0>		1705	0	141848	24
<0.22.0>		1413	0	1864	24
<0.29.0>	user_drv	936	0	7896	64
<0.132.0>		833	0	7896	88
<0.31.0>		533	0	1864	176

Traces Process State for kernel_sup ×

```
#state {
  name = {local, kernel_sup}
  strategy = one_for_all
  children =
    [
      #child {
        pid = #Pid<e@mac.34.0>
        name = kernel_safe_sup
        mfargs =
          {
            supervisor
            start_link
            [
              {local, kernel_safe_sup}
              kernel
              safe
            ]
          }
      ]
    ]
}
```



```

%% All calls to lists:seq(A,B), with 100 calls printed at most:
recon_trace:calls({lists, seq, 2}, 100)

%% All calls to lists:seq(A,B), with 100 calls per second at most:
recon_trace:calls({lists, seq, 2}, {100, 1000})

%% All calls to lists:seq(A,B,2) (sequences increasing by two) with 100 calls at most:
recon_trace:calls({lists, seq, fun([_,_,2]) -> ok end}, 100)

%% All calls to iolist_to_binary/1 made with a binary as an argument already
%% (a kind of tracking for useless conversions):
recon_trace:calls({erlang, iolist_to_binary,
    fun([X]) when is_binary(X) -> ok end},
10)

%% Calls to the queue module only in a given Pid, at a rate of 50 per second at most:
recon_trace:calls({queue, '_', '_'}, {50,1000}, [{pid, Pid}])

%% Print the traces with the function arity instead of literal arguments:
recon_trace:calls(TSpec, Max, [{args, arity}])

%% Matching the filter/2 functions of both dict and lists modules, across new
%% processes only:
recon_trace:calls([{dict,filter,2},{lists,filter,2}], 10, [{pid, new}])

%% Tracing the handle_call/3 functions of a given module for all new processes,
%% and those of an existing one registered with gproc:
recon_trace:calls({Mod,handle_call,3}, {1,100}, [{pid, [{via, gproc, Name}, new]}])

%% Show the result of a given function call, the important bit being the
%% return_trace() call or the {return_trace} match spec value.
recon_trace:calls({Mod,Fun,fun(_) -> return_trace() end}, Max, Opts)
recon_trace:calls({Mod,Fun,[['_', []], [{return_trace}]]}, Max, Opts)

```

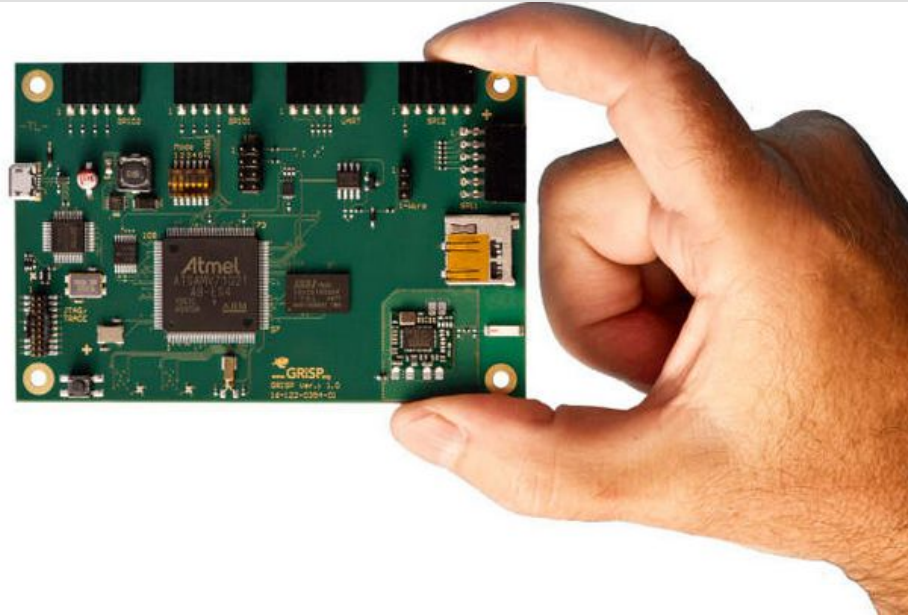

Boot into the BEAM

WIRELESS EMBEDDED SYSTEMS

GRiSP: A BASE TO BUILD ON AN ERLANG VIRTUAL MACHINE ON BARE- METAL BOARD

Create amazing Internet of Things designs without soldering or dropping down to C. Right out of the box, GRiSP-Base boots into Erlang VM running on real bare metal. It features on-board wireless networking 802.11b/g/n WLAN and connectors for standard PMod sensor and actuator modules.

More at **GRiSP.org**





Craft and deploy bulletproof embedded software

Nerves is a complete IoT platform and infrastructure for you to build and deploy maintainable embedded systems.

GET STARTED



Kry10 Secure Platform

Secure, Robust, Fast, Easy

The Kry10 Secure Platform (KSP) is a breakthrough Operating System and Support Service built on the world-class [seL4®](#), [Erlang](#), and [Elixir](#) technologies.

[LEARN MORE](#)



Programming for the BEAM

Vs

Programming in the BEAM

```

3  defmodule HelloTest do
4    use ExUnit.Case, async: true
5
6    test "it works" do
7      assert Hello.world() == "hello w
8    end
9  end
10
11  ExUnit.run()

```

Evaluated

Finished in 0.00 seconds (0.00s as
1 test, 0 failures

Randomized with seed 398770

%{excluded: 0, failures: 0, skippe

```

[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
...
]

```

>

```

1  require Axon
2
3  model =
4    Axon.input({nil, 28, 28})
5    |> Axon.flatten()
6    |> Axon.dense(128, activation: :s
7    |> Axon.dense(10, activation: :so
8

```

Evaluated

```

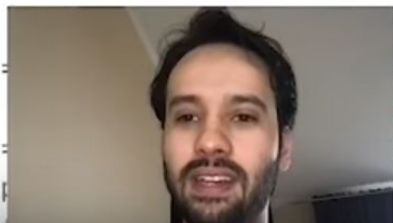
=====
Layer
=====

```

```

input_4 ( inp
flatten_5 (

```



Let's Grow and Improve the **System**

One More Thing

EFLFE

Elixir Flavoured Lisp Flavoured Erlang



Business in the Front



Aliens in the Back

An LFE to Elixir Transpiler

github.com/marianoguerra/efe#eflfe-elixir-flavoured-lisp-flavoured-erlang

e fe pp-lfe file.conf my-code.lfe

```

(defmodule ping_pong
  (export ...)
  (behaviour gen_server))      ; Just indicates intent

(defun start_link ()
  (gen_server:start_link
   #(local ping_pong) 'ping_pong '() '()))

;; Client API

(defun ping ()
  (gen_server:call 'ping_pong 'ping))

;; Gen_server callbacks

(defrecord state
  (pings 0))

(defun init (args)
  `(ok ,(make-state pings 0)))

(defun handle_call (req from state)
  (let* ((new-count (+ (state-pings state) 1))
        (new-state (set-state-pings state new-count)))
    `(reply #(pong ,new-count) ,new-state)))

(defun handle_cast (msg state)
  `(noreply ,state))

(defun handle_info (info state)
  `(noreply ,state))

(defun terminate (reason state)
  'ok)

(defun code_change (old-vers state extra)
  `(ok ,state))

```

```

defmodule :ping_pong do
  use Bitwise
  @behaviour :gen_server
  def start_link() do
    :gen_server.start_link({:local, :ping_pong}, :ping_pong, [], [])
  end

  def ping() do
    :gen_server.call(:ping_pong, :ping)
  end

  require Record
  Record.defrecord(:r_state, :state, pings: 0)

  def init(args_0) do
    {:ok, r_state(pings: 0)}
  end

  def handle_call(req_0, from_0, state_0) do
    new_count_0 = r_state(state_0, :pings) + 1

    (
      new_state_0 = r_state(state_0, pings: new_count_0)
      {:reply, {:pong, new_count_0}, new_state_0}
    )
  end

  def handle_cast(msg_0, state_0) do
    {:noreply, state_0}
  end

  def handle_info(info_0, state_0) do
    {:noreply, state_0}
  end

  def terminate(reason_0, state_0) do
    :ok
  end

  def code_change(old_vers_0, state_0, extra_0) do
    {:ok, state_0}
  end

  def unquote(:"LFE-EXPAND-EXPORTED-MACRO")(_, _, _) do
    :no
  end
end

```

Thanks

Robert Virding  @rvirding
Mariano Guerra  @warianoguerra