Beyond Elixir and BEAM

integrating with other languages and ecosystems

Membrane

Mateusz Front

Elixir Community Kraków

Motivation

- Performance
- Using non-BEAM tools/libraries

NIFs

- Native implemented functions
- C callbacks that are called by BEAM
- Defined with erl_nif library
- Linked dynamically (.so/.dll)

erl_nif

- Powerful
- Hard to use
- Lot's of boilerplate
- No build system

Rust

- Modern
- Safe
- Rustler

Rustler

- Good & popular
- Can be used from Erlang
- Supports precompilation
- Not really suitable for calling C/C++

Zig

- Meant to be a bridge to C
- Build system & C compiler
- Reasonable stdlib
- Zigler

Zigler

- Simple & straightforward
- Another language, in beta
- No support for Windows
- Calling C++ not that simple

Unifex

- Pure C or C++
- Spec-like interface DSL

Unifex

- Nothing beyond C & C++
- Simple build system (Bundlex)
- 1.0

NIFs - limitations

- Can crash the whole BEAM
- Long-running NIFs need special care
- Overhead

Dealing with long-running NIFs

- Splitting into smaller NIFs
- Yielding
- Dirty NIFs
- Manual threading

Avoiding BEAM crashes

- A separate BEAM node
- Lightweight node

Warning

Use this functionality with extreme care.

A native function is executed as a direct extension of the native code of the VM. Execution is not made in a safe environment. The VM **cannot** provide the same services as provided when executing Erlang code, such as pre-emptive scheduling or memory protection. If the native function does not behave well, the whole VM will misbehave.

- A native function that crashes will crash the whole VM.
- An erroneously implemented native function can cause a VM internal state inconsistency, which can cause a crash of the VM, or miscellaneous misbehaviors of the VM at any point after the call to the native function.
- A native function doing `lengthy work` before returning degrades responsiveness of the VM, and can cause miscellaneous strange behaviors. Such strange behaviors include, but are not limited to, extreme memory usage, and bad load balancing between schedulers. Strange behaviors that can occur because of lengthy work can also vary between Erlang/OTP releases.

Lightweight nodes

- Don't require BEAM
- Can connect to a cluster
- Communicate like any other node
- But it's 'hidden' - won't be listed in `Node.list/0`
- Usually C nodes

Erl interface

- ei_connect.h
- ei.h
- erl_interface.h [*]

Erl interface

OTP 23

Erlang/OTP 23 is a new major release with new features, improvements as well as a few incompatibilities.

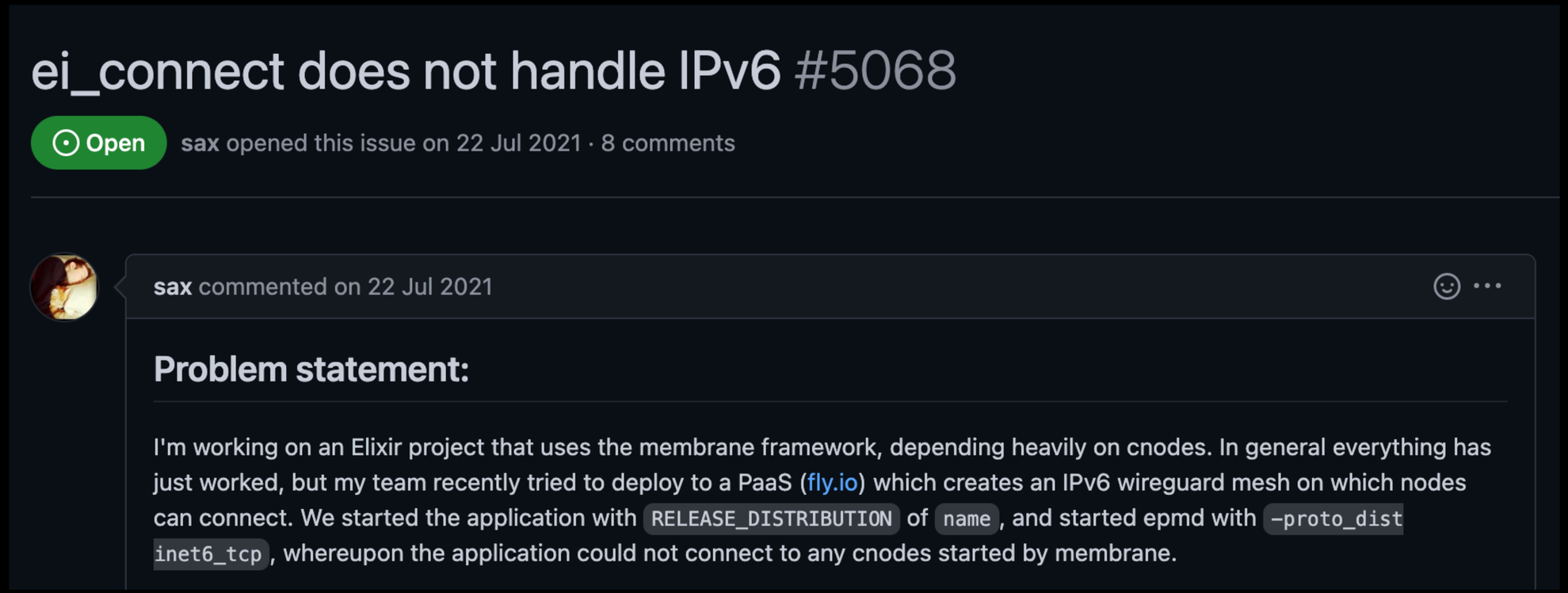
Potential Incompatibilities

- **SSL**:
  - Support for SSL 3.0 is completely removed.
  - TLS 1.3 is added to the list of default supported versions.
- `erl_interface`: Removed the deprecated parts of `erl_interface` (`erl_interface.h` and essentially all C functions with prefix `erl_`).

Lightweight nodes - limitations

- Overhead
- Distribution needed

Lightweight nodes - limitations



ei_connect does not handle IPv6 #5068

Open sax opened this issue on 22 Jul 2021 · 8 comments

sax commented on 22 Jul 2021

Problem statement:

I'm working on an Elixir project that uses the membrane framework, depending heavily on cnodes. In general everything has just worked, but my team recently tried to deploy to a PaaS (fly.io) which creates an IPv6 wireguard mesh on which nodes can connect. We started the application with `RELEASE_DISTRIBUTION` of `name`, and started epmd with `-proto_dist inet6_tcp`, whereupon the application could not connect to any cnodes started by membrane.

https://github.com/erlang/otp/issues/5068

Ports

- Separate OS processes
- Communication over standard I/O
- Port module
- System.cmd

Ports - limitations

- Overhead
- Problems with third-party programs
- No built-in support for Erlang terms

Third-party ports

- Zombie processes
- No back pressure

Third-party ports - libraries

Library	Characteristics	Handles back pressure?	Avoids zombie processes?
rambo	simple, developer friendly	no	yes
ex_cmd	back pressure, Stream abstraction	yes	yes
erlexec	in Erlang, allows fine-grained control	no	yes
exile	no middleware	yes	Unless BEAM crashes
porcelain [*]	not maintained, known bugs	no	?

External Term Format

- Binary format for serialising Erlang terms
- Any Erlang term can be encoded
- :erlang.term_to_binary & :erlang.binary_to_term
- Used by Erlang distribution
- Specified in Erlang docs

External Term Format - libraries

- erl_interface
- erlpack - javascript, python, go

Port drivers

kennethL Erlang Core Team 1d

Actually port drivers is the old solution and our intention is that NIFs can be used instead of port drivers for all use cases. We don’t recommend writing new code with the port driver concept use NIFs instead. In the not so near future we might remove support for port drivers.

Solution 6 Reply

Luerl

README.md

Luerl - an implementation of Lua in Erlang

The migration from Lua 5.2 to 5.3 is very much Work-In-Progress. Please test it but there are as yet no guratantees.

Luerl is an implementation of standard Lua 5.3 written in Erlang/OTP.

Lua is a powerful, efficient, lightweight, embeddable scripting language common in games, IoT devices, AI bots, machine learning and scientific computing research.

It supports procedural, object-oriented, functional, data-driven, reactive, organizational programming and data description.

Being an extension language, Lua has no notion of a "main" program: it works as a library embedded in a host simple called the embedding program. The host program can invoke functions to execute a piece of Lua code, can write and read Lua variables, and can call Erlang functions by Lua code.

Through the use of Erlang functions, Luerl can be augmented to cope with a wide range of different domains, creating a customized language sharing a syntactical framework.

Thanks!

- Andrew Bennett's talk on dirty NIFs
  youtube.com/watch?v=FYQcn9zcZVA
- erl_nif erlang.org/doc/man/erl_nif.html
- rustler github.com/rusterlium/rustler
- zigler github.com/ityonemo/zigler
- unifex github.com/membraneframework/unifex
- erl_interface
  erlang.org/doc/apps/erl_interface/ei_users_guide.html
- Port hexdocs.pm/elixir/1.13/Port.html
- rambo github.com/jayjun/rambo
- ex_cmd github.com/akash-akya/ex_cmd
- erl_exec github.com/saleyn/erlexec
- exile github.com/saleyn/exile
- erlpack github.com/discord/erlpack
- luerl github.com/rvirding/luerl