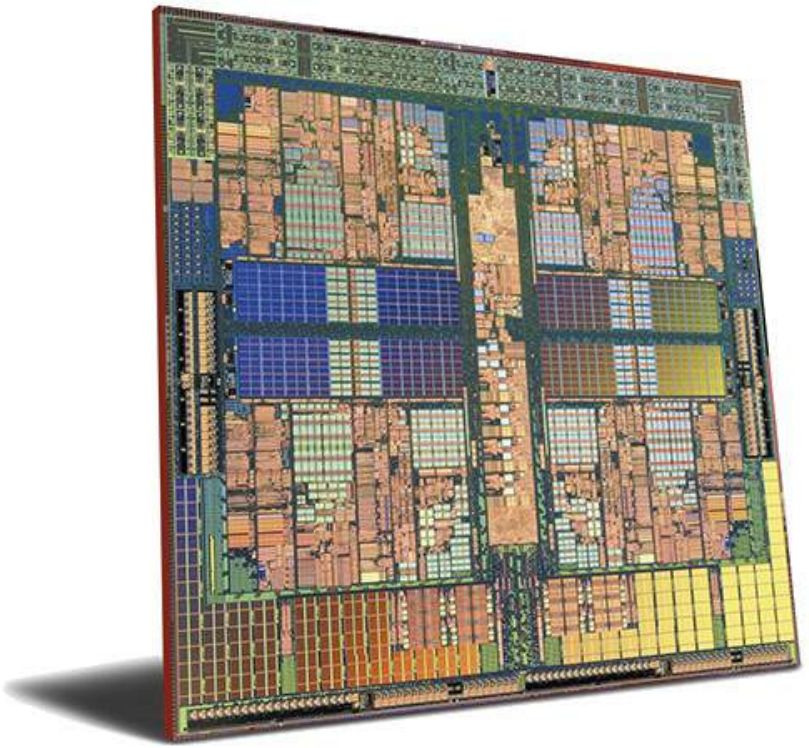
Rendering the World

3d reconstruction on the BEAM

What is Elixir really good at?



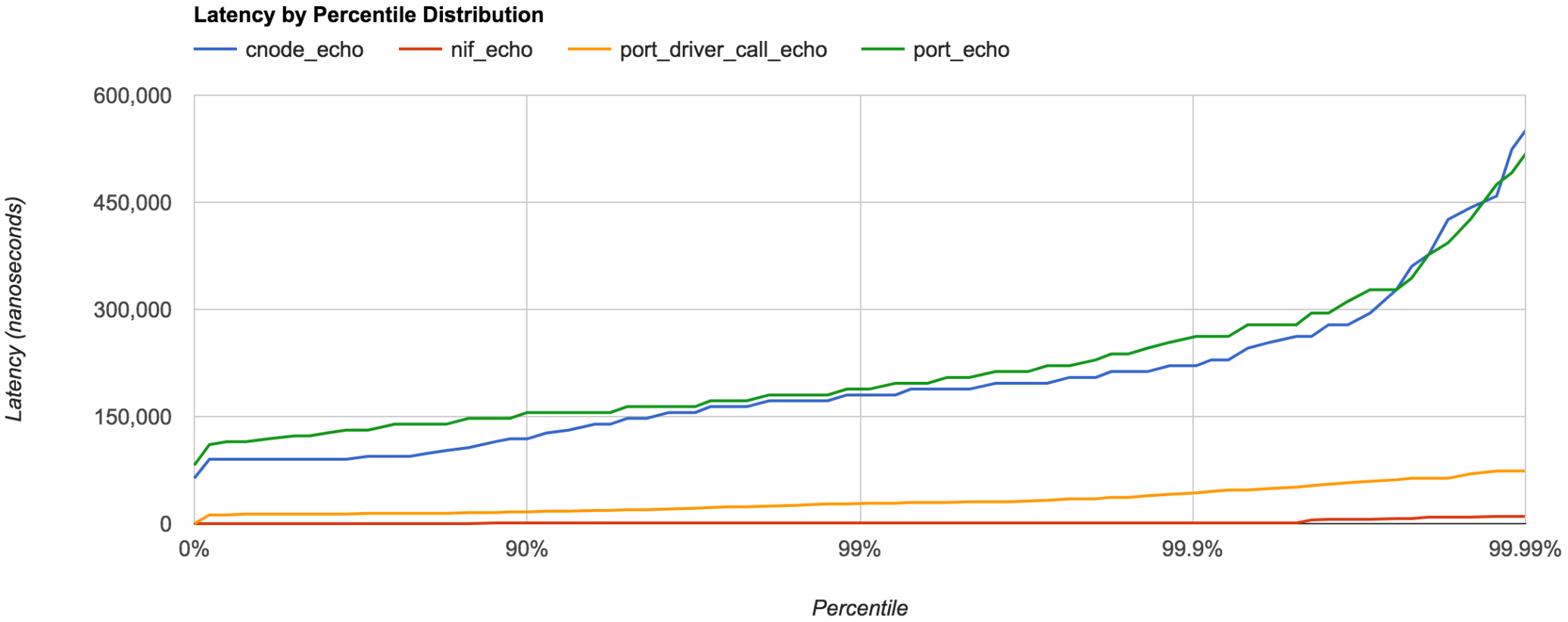
The Challenge

- Reconstructing 1 km^2 takes ~3k CPU hours
- Individual processes may run from minutes to days

- One machine working on a job is too slow
- Crashes are very costly
- This is a large C/C++ codebase with some Python

Native Bindings

https://potatosalad.io/2017/08/05/latency-of-native-functions-for-erlang-and-elixir



nifpp

Single-header library → https://github.com/goertzenator/nifpp

- Templated get / make replaces enif_[get|make]_*()
- STL containers: tuple, vector, array, list, deque, set, unordered_set, multiset, map, and unordered_map. Even nested!
- A resource pointer type so that any type can be easily used as a NIF resource; similar to a std::shared_ptr that the BEAM can hold references to.

So easy!

```
template<typename TK, typename TV>
nifpp::TERM mapflip_test(ErlNifEnv* env, ERL_NIF_TERM term)
{
    std::map<TK,TV> inmap;
    std::map<TV,TK> outmap;
    get_throws(env, term, inmap);
    for (auto i = inmap.begin(); i =/= inmap.end(); i++)
    {
        outmap[i->second] = i->first;
    }
    return make(env, outmap);
}
```

SafeExecEnv

```
defmodule MyApp do
  @moduledoc false

  use Application

  def start(_type, _args) do
    children = [SafeExecEnv]
    opts = [strategy: :one_for_one, name: MyApp.Sup]
    Supervisor.start_link(children, opts)
  end
End

SafeExecEnv.exec(fn → 2 * 3 end)
SafeExecEnv.exec(module, func, args)
```

Clustering

Automatically organizing self-healing clusters:

- libcluster
- epmdless
- erlang_node_discovery
- lbm_kv
- RaftFleet

Hardware resources as workers

Hardware resources as workers

A worker per

- CPU core (Big / Little)
- GPU
- I/O thread

Configuration …

Thank you! Questions?

Aaron Seigo <aaron@nomoko.world>