

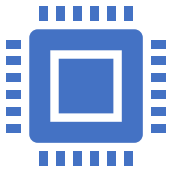
C++ in the Cloud

One NIF at a Time with Elixir

Sakshi Verma | Senior Software Engineer



What is Elixir?



Functional, concurrent language built for scalability.



Ideal for managing large-scale cloud applications.



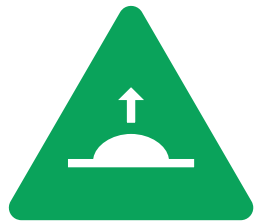
Built on the Erlang VM, ensuring resilience and fault tolerance.



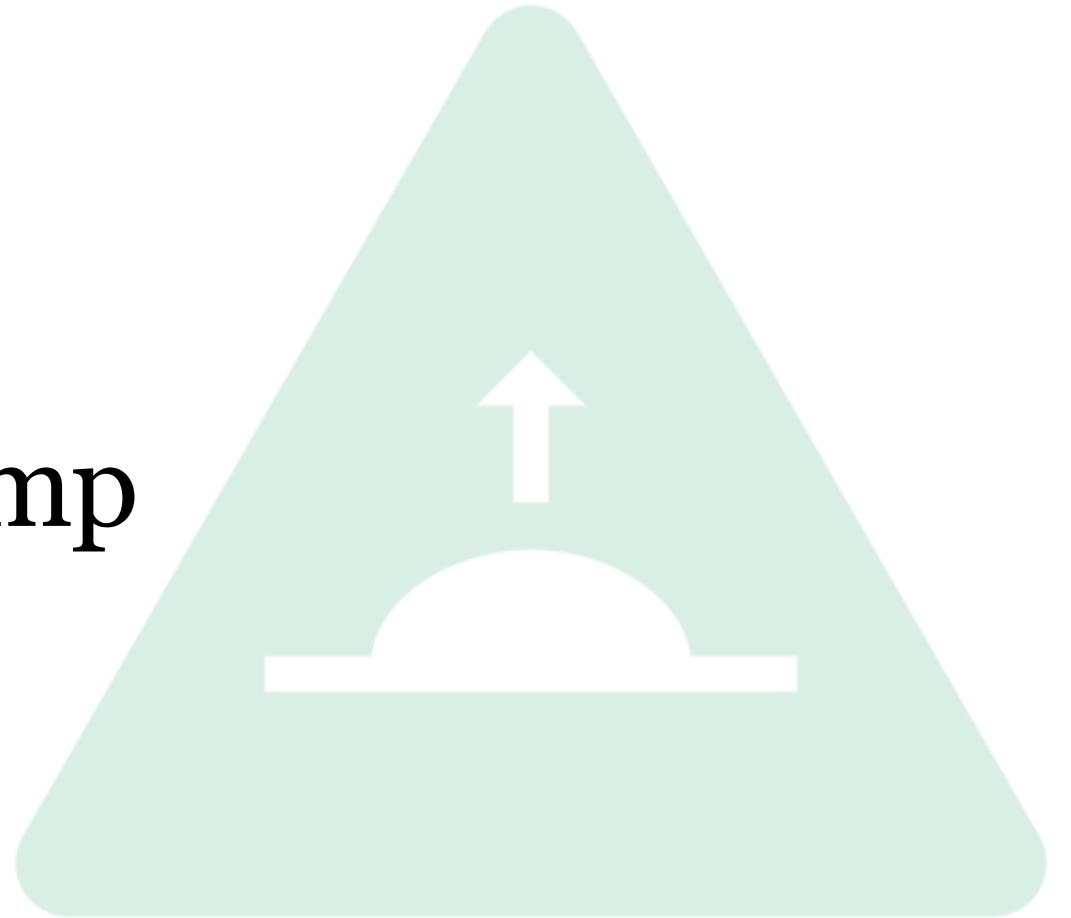


The Smooth Sailing...





...and then The Bump



Enter Native Operations



Option 1: Rewrite all the native functionality in Elixir.

Option 2: Use already existing functionality from Native Code.

A person in a red jacket is walking away from the camera on a narrow suspension bridge. The bridge has a metal grate floor and wire mesh railings. It spans a deep valley filled with dense evergreen trees. The scene is misty or foggy, with the background fading into a soft, grey light. The overall mood is quiet and somewhat mysterious.

Here comes the NIFs

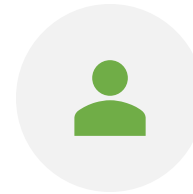
Native Implemented Functions (NIFs)



Call C/C++ from Elixir



Compiles to a shared library



Loaded into Erlang VM at startup

Why Write a NIF?

01

Erlang/Elixir ≠
CPU-heavy
tasks.

02

Direct
hardware
interaction.

03

Interop with
graphics
libraries.

04

Existing Native
libraries too
complex to
rewrite.


```
        # or object to mirror
        mirror_mod.mirror_object =
```

```
        operation == "MIRROR_X":
            mirror_mod.use_x = True
            mirror_mod.use_y = False
            mirror_mod.use_z = False
        operation == "MIRROR_Y":
            mirror_mod.use_x = False
            mirror_mod.use_y = True
            mirror_mod.use_z = False
        operation == "MIRROR_Z":
            mirror_mod.use_x = False
            mirror_mod.use_y = False
            mirror_mod.use_z = True
```

```
        # selection at the end -add
        mirror_ob.select= 1
        modifier_ob.select=1
        context.scene.objects.active
        ("Selected" + str(modifier_ob.name))
        mirror_ob.select = 0
        bpy.context.selected_objects
        data.objects[one.name].select
```

```
print("please select exactly 1 object")
```

```
-- OPERATOR CLASSES --
```

```
bpy.types.Operator):
    X mirror to the selected
    object.mirror_mirror_x"
    mirror X"
```

NIF Code: A Closer Look

```

#include <erl_nif.h>
#include <xgboost/c_api.h>

ERL_NIF_TERM xgboost_build_info(ErlNifEnv* env, int argc, const ERL_NIF_TERM argv[])
{
    char out[1024];
    int result = XGBuildInfo(out);

    if (result == 0) {
        return enif_make_string(env, out, ERL_NIF_LATIN1);
    }
    else {
        return enif_make_badarg(env);
    }
}

// NIF initialization (maps Elixir function to C++ function)
static ErlNifFunc nif_funcs[] =
{
    {"xgboost_build_info", 0, xgboost_build_info}
};

// Initialize the NIF module
ERL_NIF_INIT(Elixir.XGBoost.NIF, nif_funcs, NULL, NULL, NULL, NULL)

```

```
defmodule XGBoost.NIF do
  @on_load :on_load

  # Load the NIF when the module starts
  def on_load do
    :erlang.load_nif("./libxgboost", 0)
  end

  # Elixir function that interfaces with the C++ NIF
  def xgboost_build_info do
    raise "NIF not loaded"
  end
end
```



Content Credentials: Generated with Microsoft Copilot



Crashes affect the whole VM.

Lacks fault tolerance and isolation.

Not interrupted by the Erlang scheduler.

A black and white photograph of a vintage open-top car, likely a Jaguar XK150, shown from a front-three-quarter perspective. The car is dark-colored with a prominent white racing stripe running down the center of its hood and over the roof. The front wheel features a wire-spoke design. The text "Bon Voyage" is superimposed in a white, elegant serif font across the middle of the image, positioned over the front wheel and the lower part of the hood. The background is a soft-focus landscape with rolling hills under a bright sky.

Bon Voyage