# NIF with Rustler

홍석민

2022년 6월 Elixir Korea Meetup



# 발표자

- C++로 게임 개발
- TypeScript/React로 프론트엔드 개발
- 모멘티 코리아 이직 후 Elixir 시작
- Elixir/Phoenix LiveView를 사용, 웹 기반 에디터 구현
- Rustler 프로젝트 기여자

### NIFs?

- Native Implemented Functions
- C로 구현된 dynamic library를 Erlang에서 호출
- 런타임에 Erlang 모듈에 연결됨

# NIF의 특징

- NIF는 Erlang 에뮬레이터 프로세스에 로드됨
- NIF를 호출할 때 context switch가 없음
- NIF에서 크래시가 발생하면 에뮬레이터 프로세스가 죽음
- 동기적, 짧은 작업 시간, side effect가 없는 함수가 NIF로 적합함

### Momenti Uses NIFs

#### 요구사항

- 동영상 편집 에디터
- 크로스 플랫폼(MacOS/Linux/iOS/WASM)에서 동작하는 코어 엔진이 필요

### Momenti Uses NIFs

#### Rust

- 쉬운 크로스 컴파일링
- 충분히 성숙해진 라이브러리와 커뮤니티
- 높은 안정성, 성능

# NIF with C

```
// lib/calc.h
int add_int(int a, int b)
{
   return a + b;
}
```

```
// nif calc.c
#include <erl_nif.h>
#include "lib/calc.h"
static ERL_NIF_TERM add_int_nif(
    ErlNifEnv *env,
    int argc,
    const ERL_NIF_TERM argv[]
    int a, b, result;
    enif_get_int(env, argv[0], &a);
    enif_get_int(env, argv[1], &b);
    result = add int(a, b);
    return enif make int(env, result);
```

# 용어

- term: Erlang에서 piece of data를 타입에 상관 없이 칭하는 단어
- env: Erlang terms들이 바운드된 environment
- enif\_get\_\*: 주어진 term으로부터 원하는 타입의
   값을 읽음
- enif\_make\_\*: C의 값을 term으로 인코딩함

```
// nif_calc.c
static ErlNifFunc nif_funcs[] = {
    // {function name, arity, function ptr}
    {"add_int", 2, add_int_nif},
};

ERL_NIF_INIT(Elixir.Calc, nif_funcs, NULL, NULL, NULL)
```

```
defmodule Calc do
  @on_load :load_nifs

def load_nifs do
    :erlang.load_nif('native/nif_calc', 0)
  end

def add_int(_a, _b), do: error()

defp error(), do: :erlang.nif_error(:nif_not_loaded)
end
```

## Build / Link

```
gcc -o lib_calc.so -c lib/calc.c

export INCLUDES=-I/{ERLANG_PATH}/usr/include
# x86
gcc $INCLUDES -shared -o nif_calc.so -fPIC nif_calc.c lib_calc
# ARM
gcc $INCLUDES -o nif_calc.so -undefined dynamic_lookup -dynami
```

iex> Calc.add(1, 2)
3

# 문제점

- 모든 인자와 반환값을 직접 인코딩/디코딩해야 함
- 바인딩 로직을 직접 작성해야 함
- 빌드 과정이 번거로움

## Rustler



https://github.com/rusterlium/rustler

### Rustler

- Rust로 NIF에 안전하게 바인딩할 수 있는 인터페이스를 제공
- Rust의 문법을 Elixir에 적절히 대응하여 *자연스러운* 바인딩 가능
- Rust에서 크래시가 발생해도 프로세스가 다운되지 않음
- 빌드가 간편함

```
#[rustler::nif]
fn add(a: i64, b: i64) -> i64 {
    a + b
}
rustler::init!("Elixir.Calc", [add]);
```

```
defmodule Calc do
  use Rustler, otp_app: :rustler_nif, crate: :calc
  @spec add(integer(), integer()) :: integer()
  def add(_a, _b), do: error()

  defp error(), do: :erlang.nif_error(:nif_not_loaded)
end
```

### How it works

- Rust의 매크로를 사용
- 바인딩 코드를 자동 생성

### cargo-expand

https://github.com/dtolnay/cargo-expand

• Rust의 매크로를 expand하여 최종 결과를 보여주는 도구

```
pub struct add;
impl rustler::Nif for add {
    const NAME: *const u8 = "add\u{0}".as_ptr() as *const u8;
    const ARITY: u32 = 2u32;
    const FLAGS: u32 = rustler::SchedulerFlags::Normal as u32;
    const RAW_FUNC: unsafe extern "C" fn(
        nif_env: rustler::codegen_runtime::NIF_ENV,
        argc: rustler::codegen_runtime::c_int,
        argv: *const rustler::codegen_runtime::NIF_TERM,
    ) -> rustler::codegen_runtime::NIF_TERM = {
        ...
    };
}
```

```
let result: std::thread::Result< > = std::panic::catch unwind(
    let a: i64 = match args[Ousize].decode() {
        Ok(value) => value,
        Err(err) => return Err(err),
    };
    let b: i64 = match args[lusize].decode() {
        Ok(value) => value,
        Err(err) => return Err(err),
    };
    fn add(a: i64, b: i64) -> i64 {
        a + b
    Ok(add(a, b))
});
```

### Rustler <-> Elixir

- Elixir와 Rust의 자료형을 *자연스럽게* 이어줌
- 유저 정의 타입에 매크로를 사용하여 변환 가능
- Rust의 매크로가 encode/decode 로직을 작성해줌

#### Enum

#### Struct <-> Map

```
#[derive(NifMap)]
pub struct Test {
    a: i32,
    b: f64,
}
// %{a: integer(), b: float()}
```

#### **Struct <-> Struct**

```
#[derive(NifStruct)]
#[module = "SomeModule.Test"]
pub struct Test {
    a: i32,
    b: f64,
}
// %SomeModule.Test{a: integer(), b: float()}
```

# 함수 입력 및 출력 단순화

- 함수의 인자는 decode 가능한 타입이면 전부 가능
- 함수의 첫 번째 인자에 선택적으로 env를 사용 가능
- 함수의 반환 타입은 encode 가능한 타입이면 전부 가능
- 함수의 반환 타입으로 NifResult 허용

```
#[nif]
fn send_message(env: Env, pid: LocalPid, message: String) {
    let msg = message.encode(env);
    env.send(&pid, msg);
@spec send_message(pid(), binary()) :: no_return()
def send_message( pid, msg), do: error()
         iex(1)> Calc.send_message(self(), "hello")
         {}
         iex(2)> receive do
         ...(2) value -> IO.puts(value)
         ...(2)> end
         hello
         :ok
```

```
#[nif]
fn add(a: f64, b: f64) -> f64 {
    a + b
#[nif]
fn div(a: f64, b: f64) -> NifResult<f64> {
    if b == 0.0 {
        // :div_by_zero = div(1.0, 0.0)
        Err(Error::Atom("div_by_zero"))
    } else {
        Ok(a / b) // 10.0 = div(1.0, 0.1)
```

# 시행착오

### 긴 빌드 시간

- Rust는 의존성이 더 많음
- 빌드 캐시가 되지 않는 CI 환경에서 두드러짐

rustler\_precompiled

https://github.com/philss/rustler\_precompiled

• 바인딩 crate를 GitHub Releases로부터 바로 가져올 수 있는 라이브러리

```
defmodule Example.Native do
  version = Mix.Project.config()[:version]

use RustlerPrecompiled,
  otp_app: :rustler_precompilation_example,
  crate: "example",
  base_url:
    "https://github.com/philss/rustler_precompilation_exampl
  version: version

# When your NIF is loaded, it will override this function.
  def add(_a, _b), do: :erlang.nif_error(:nif_not_loaded)
end
```

### 단점

- 배포 파이프라인에 커스터마이징이 힘듦
- 아직 개발 초기 단계임

#### 직접 구축하자!

- Rustler의 load\_from 및 skip\_compilation? 플래그로 사전에 빌드한 라이브러리를 사용 가능
- Crate 배포 시 실행할 플랫폼들을 대상으로 빌드
- GitHub Releases에 빌드된 파일 추가
- Elixir 레포지토리에 PR을 자동생성하여, 빌드된 파일을 푸시



8	Replace build script Release precompiled NIFs #19: Commit e6845b2 pushed by SeokminHong	0.0.0-a20
(!)	Fix script Release precompiled NIFs #18: Commit adc3de4 pushed by SeokminHong	0.0.0-a19
8	Rollback again Release precompiled NIFs #17: Commit a23834d pushed by SeokminHong	0.0.0-a18
(!)	Rollback to use actions-rs/cargo and setup Cross.toml Release precompiled NIFs #16: Commit 38dc483 pushed by SeokminHong	0.0.0-a17
8	Do not use actions-rs/cargo Release precompiled NIFs #15: Commit 2df0f06 pushed by SeokminHong	0.0.0-a16
8	Remove quotes Release precompiled NIFs #14: Commit 91113c8 pushed by SeokminHong	0.0.0-a15
8	Export RUSTFLAGS Release precompiled NIFs #13: Commit a3553e0 pushed by SeokminHong	0.0.0-a14
8	Fix CI script Release precompiled NIFs #12: Commit 1326b4c pushed by SeokminHong	0.0.0-a13
8	Add flags by manually Release precompiled NIFs #11: Commit e85f90b pushed by SeokminHong	0.0.0-a12
8	Fix command Release precompiled NIFs #10: Commit 2ce0601 pushed by SeokminHong	0.0.0-a11
8	Add cross compilation configure for alpine Release precompiled NIFs #9: Commit c6af63c pushed by SeokminHong	0.0.0-a10

#### Alpine build

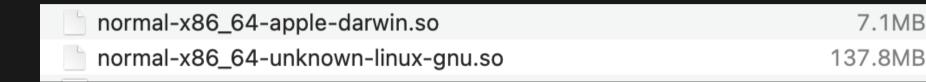
- GitHub CI의 Linux 환경은 Ubuntu만 제공
- Ubuntu에서 Rust의 기본 툴체인으로 Alpine 타깃 빌드를 하면 glibc를 사용함
- Alpine에는 glibc가 없기 때문에 실행 불가능

#### cross

#### https://github.com/cross-rs/cross

- Rust의 cross compiliation 툴
- 원하는 타깃의 Docker daemon을 띄워 그 안에서 빌드
- 🗸

#### **Ubuntu binary size**



- Ubuntu에서 빌드할 때 바이너리 크기가 지나치게 큼
- Ubuntu에서 Rust 공유 라이브러리 빌드 시, 디버그 심볼이 들어가는 버그
- https://users.rust-lang.org/t/binary-is-waybigger-on-linux-than-on-macos/14814/3

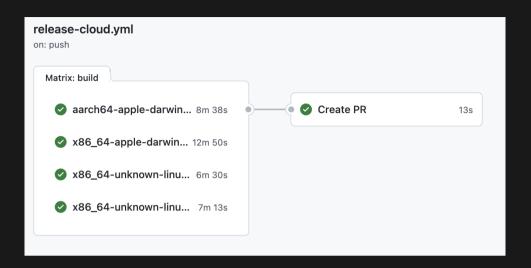
#### strip

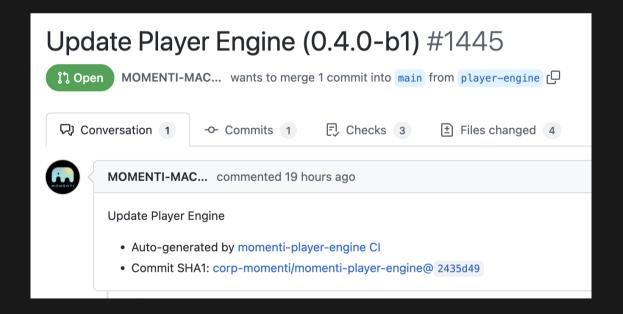
stripped-x86_64-unknown-linux-gnu.so	5.3MB
normal-x86_64-apple-darwin.so	7.1MB
normal-x86_64-unknown-linux-gnu.so	137.8MB

```
$ strip -S ${LIB_FINAL_NAME}
```

• 빌드 후 디버그 심볼을 강제로 제거







#### 비동기!

- NIF는 기본적으로 동기적 함수를 호출하는 것을 전제함
- 코어 엔진은 비동기 호출 위주인데 가능할까?

```
async fn some_task(task_ended: Callback<SomeTask>) {
    job().await;
    task_ended.invoke(payload);
}
```



```
#[nif]
fn start_some_task(pid: LocalPid) {
    thread::spawn(move | | {
        runtime.block on(async {
            some task(Callback{ pid }).await
        });
    });
#[cfg(feature = "elixir")]
impl Callback<SomeTask> {
    fn invoke(&self, payload: SomeTaskPayload) {
        OwnedEnv::new().send_and_clear(self.pid, move | env | {
            ("some task", payload).encode(env)
        });
```

```
Example.start_some_task(self())
receive do
   {"some_task", payload} -> payload
end
```



### References

- Erlang NIFs Document: https://www.erlang.org/doc/tutorial/nif.html
- erl\_nif manual: https://www.erlang.org/doc/man/erl\_nif.html
- Elixir and NIF: a study case: https://dev.to/rodrigocaldeira/elixir-and-nif-astudy-case-2npp

# 감사합니다

예제 코드 및 구체적인 구현은

다음 저장소에서 보실 수 있습니다.

https://github.com/SeokminHong/nif-elixir-example

PDF:

https://github.com/SeokminHong/presentations/raw/m