RustNL GOSIM 2024 EUROPE

MOXIN

A PURE RUST EXPLORER FOR OPEN SOURCE LLMS

RustNL GOSIM 2024 EUROPE

MOXIN

A Pure Rust Explorer for Open Source LLMs

Jorge Bejar
CTO at WyeWorks

Hydai Tai
Hung-Ying, Tai (hydai)
WasmEdge Maintainer

May 6th, 2024

Agenda

- What is Moxin?
- Moxin implementation
- Demo
- Final notes

WHAT IS MOXIN?

GOSIM 2024 EUROPE

Discover, download, and run local LLMs

Search Model by Keyword

Discover

Chat

My Models

Explore

Llama-2-7B-Chat-GGUF 352 86529

Released Sep 4, 2023 (240 days ago)

Model Size | Requires | Architecture

Model Summary

Llama

View All

Resources

TheBloke

Hugging Face

Model File	Full Size	Quantization

Show All Files (12)

Llama-3-8B-Instruct-GGUF 2 2268

Released Apr 19, 2024 (13 days ago)

Model Size | Requires | Architecture

Model Downloads 0 downloading 1 paused

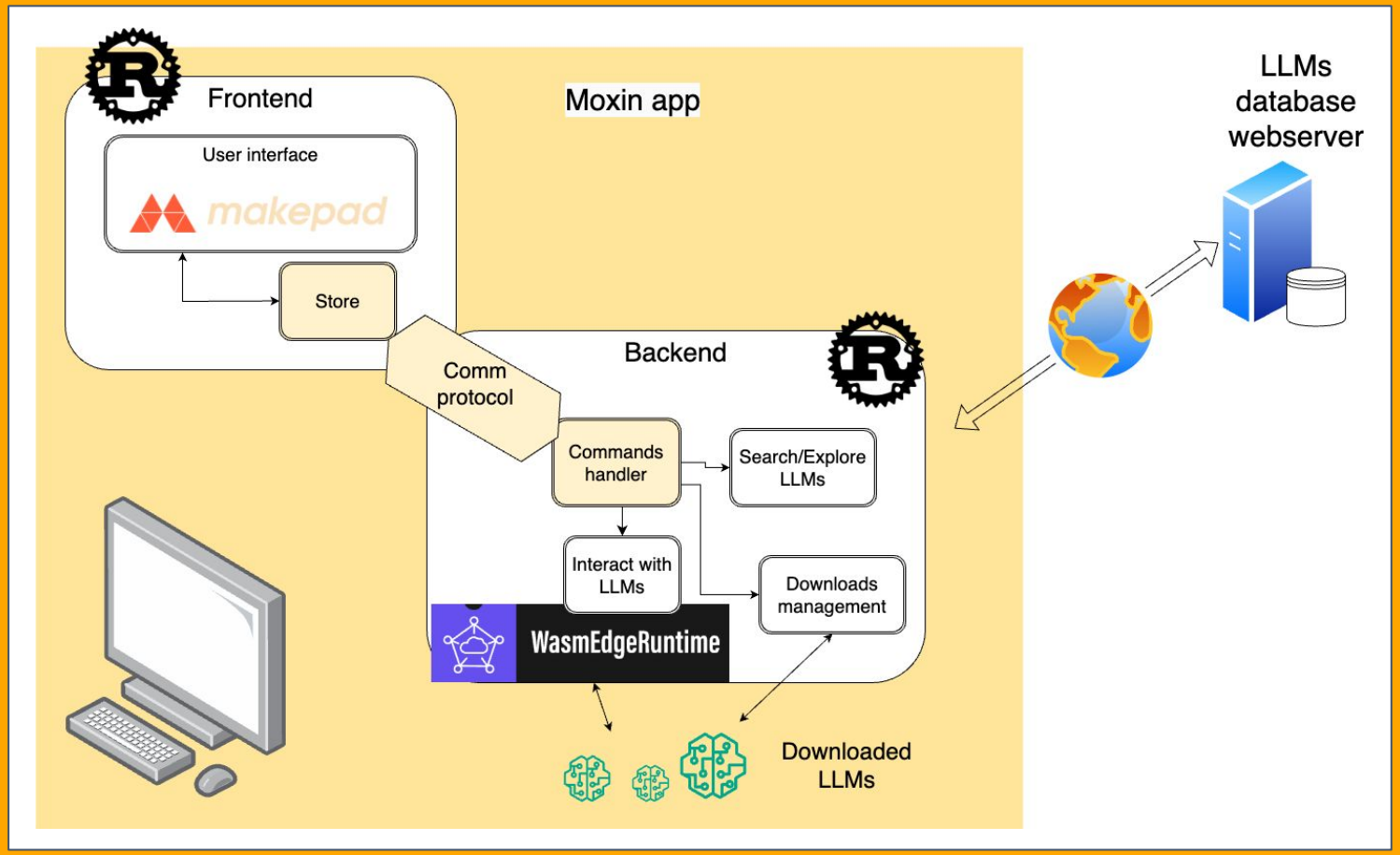
- An application to explore and experiment with open source LLMs
- Run LLMs in your own machine!
- Users can try out different prompts to pick the most appropriate model for their needs
- We just have an MVP… still growing

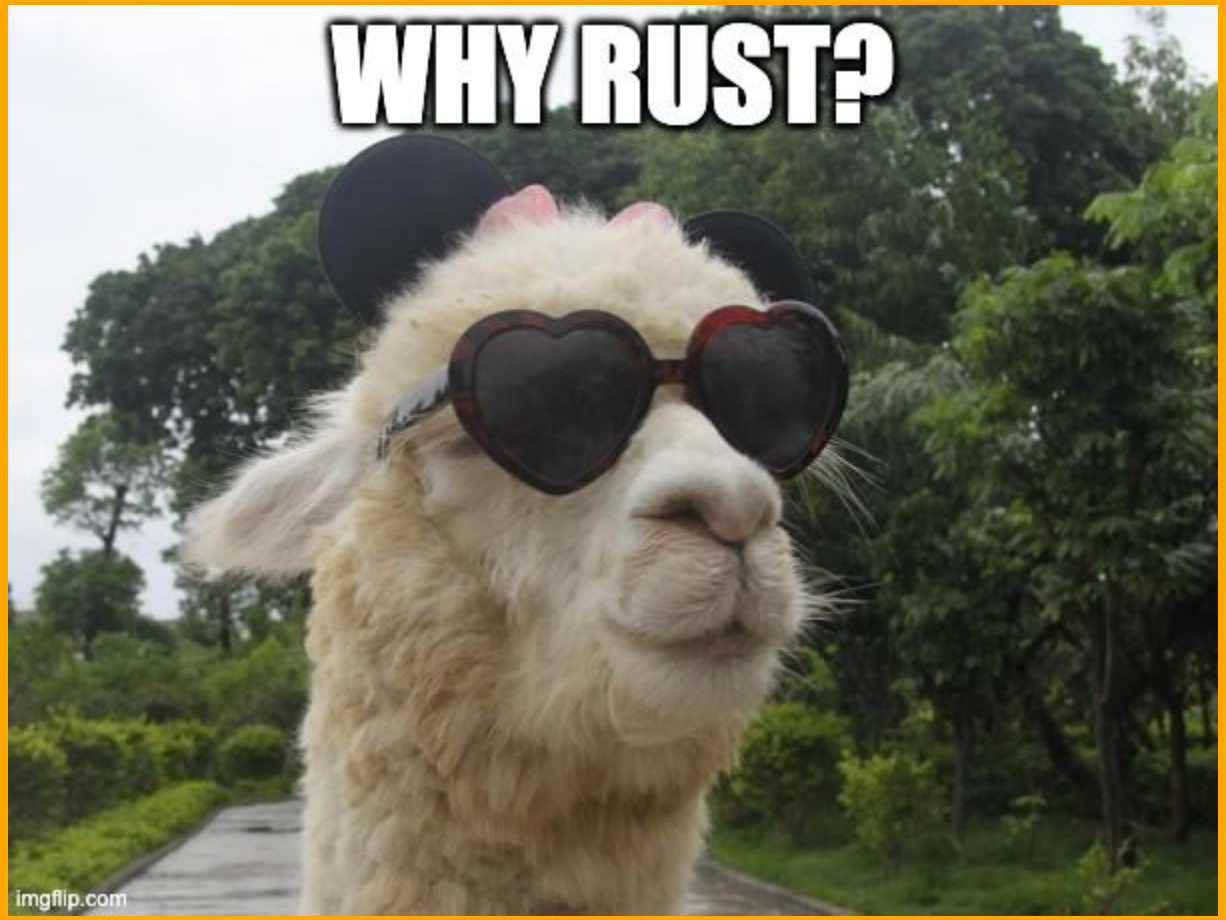

demo time
imgflip.com

- An application to explore and experiment with open source LLMs
- Run LLMs in your own machine!
- Users can try out different prompts to pick the most appropriate model for their needs
- We just have an MVP… still growing
- It's **Open Source**!

https://github.com/project-robius/moxin

ARCHITECTURE




WHY RUST?
imgflip.com

Why Rust?

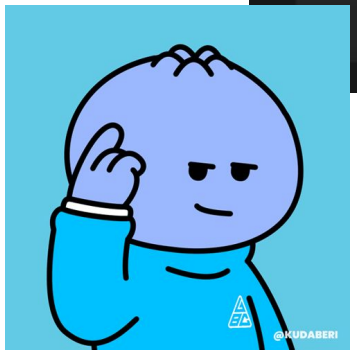
- Performance, safety, cross platform compilation
- Interoperability
- Productivity
- Availability of crates for many problems and domains

We will discuss later: why Rust for the frontend?

Why Rust?

- Perform
- Interope
- Producti

In other languages simple things are easy and complex things are possible, in Rust simple things are possible and complex things are EASY.



Cited from:
https://waszczyk.com/rustic-introduction-into-substrate-framework-syntax-and-design-patterns

RustNL GOSIM 2024 EUROPE

MOXIN BACKEND

Hung-Ying, Tai (hydai)
WasmEdge Maintainer

Moxin Backend - The Rust Part

- Main Loop:
  - Load the pre-built Wasm file (chat_ui.wasm, a black box, will explain later)
  - Retrieve a request from the front end
  - Dispatch the request into command handler
- Command Handler:
  - Model Management:
    - List and Search Models
    - Download, Pause, Cancel, and Delete Models
  - Model Interaction:
    - Load and Eject Models
    - Run and Stop the Chat Completion
    - Start and Halt a Local LLM Server (TODO)

- LoadModel:
  - Create a Model instance with the given model
  - Spawn a thread to run the Wasm Application with the given configuration
  - In the spawned thread:
    - Use WasmEdge SDK to create a standalone Wasm runner
    - Setup the configuration from the request:
      - Read `context size(n_ctx)`, `gpu layers(n_gpu_layers)`, `prompt template` and more options from the given information
      - Set the corresponding options into the Wasm runner
    - Register three special backend host functions into the Wasm runner to handle the IO
      - **get_input**: Allow wasm app to receive input from the backend
      - **push_token**: Allow wasm app to send output to the backend
      - **return_token_error**: When error occurs, use this function to return the error code instead of putting an output token.
  - Enter the entry point of the Wasm application.

Moxin Backend - The Wasm Part

- The black box Wasm is modified from llamaedge/chat: https://github.com/L-jasmine/LlamaEdge/tree/chat_ui/chat_ui
- It’s a command line interface application.
- That’s why we need to hook the IO with the previous 3 host functions.
- The execution flow:
  - Parse the options and initialize the model
  - Enter the main loop
    - Call `get_input` to retrieve the input from the backend
    - Build the prompt with input and prompt template
    - Run the compute function to ask model to generate tokens
    - When the token is generated, call `push_token` to return to the backend
    - If an error happens during the computation, call `return_token_error` instead

MOXIN FRONTEND

Applications development with Rust?

- Ecosystem is a bit rough yet :(
  - Not clear what tools are production-ready or recommended
  - Lack of examples and proper documentation
  - Crates with overlapping features, hard to integrate them

Applications development with Rust?

- Ecosystem is a bit rough yet
- … but there is hope!

ROBIUS

makepad

https://robius.rs

ROBIUS

Robius is a fully open-source, decentralized, community-driven effort to enable multi-platform application development in Rust.

Project Robius

Community for Multi-platform Application Development in Rust

201 followers https://robius.rs

README.md

Robius: Multi-Platform App Dev in Rust

view Robius Book | view Code Examples | Matrix Chat | 中文

Robius is a fully open-source, decentralized, community-driven effort to enable multi-platform application development in Rust.

nk

(15MB)

ng group: a welcoming, public space to collect and discuss resources
ce in Rust.

ntributing?
Robius space.

We believe that the Rust programming language is the right choice for the next generation of application developers, but that the

makepad

Blazingly Fast Cross Platform Rust UIs

Makepad is an in-development shader based live designable OSS UI-Framework

Check out our Demos

Join our Discord Server!

Makepad framework

- It’s a framework including a collection of highly-performant widgets and minimal, zero/low-overhead platform abstractions.
- Unique approach for UI development combining retained and immediate mode.
- Rapid development cycle: very fast compile times due to a custom minimal dependency set, plus a custom DSL for live design that enables hot reloading of UI elements.

FRONTEND - MAKEPAD

Communication frontend - backend

- Relies on `std::sync::mpsc::channel`
- Works well for synchronous and asynchronous commands
- Designed to be re-implemented for distributed or web applications
  - The Makepad frontend could also be deployed in other platforms without much rework.

A practical case with Moxin

CLOSING NOTES

What is in the Roadmap?

- Go beyond plain-text conversations (image, video, charts)
  - Get the most from Makepad!
- Integrate with cloud APIs
- Agents orchestration to accomplish complex tasks
- Multiplatform application

Our long term goals

- We aim to build an explorer for the AI.
- Engage with the social community.
- Fully integrate with the federated Matrix ecosystem

GOSIM 2024 EUROPE

github.com/project-robius/moxin

README | Apache-2.0 license

Moxin: a Rust AI LLM client built atop Robius

Moxin is an AI LLM client written in Rust to demonstrate the functionality of the Robius, a framework for multi-platform application development in Rust.

> ⚠️ Moxin is just getting started and is not yet fully functional.

The following table shows which host systems can currently be used to build Robrix for which target platforms.

Host OS	Target Platform	Builds?	Runs?
macOS	macOS	✅	✅
Linux	ubuntu(x86_64-unknown-linux-gnu)	✅	?

Building and Running

First, install Rust.

Then, install the required WasmEdge WASM runtime:

No packages published
Publish your first package

Contributors 8

Languages

Rust 100.0%

Suggested workflows
Based on your tech stack

Rust
Build and test a Rust
Cargo.

SLSA Generic generator

https://github.com/project-robius/moxin

THANK YOU

More information available at:

@jmbejar

jorge-bejar

GOSIM 2024 EUROPE