OPEN AI + DATA FORUM @ THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

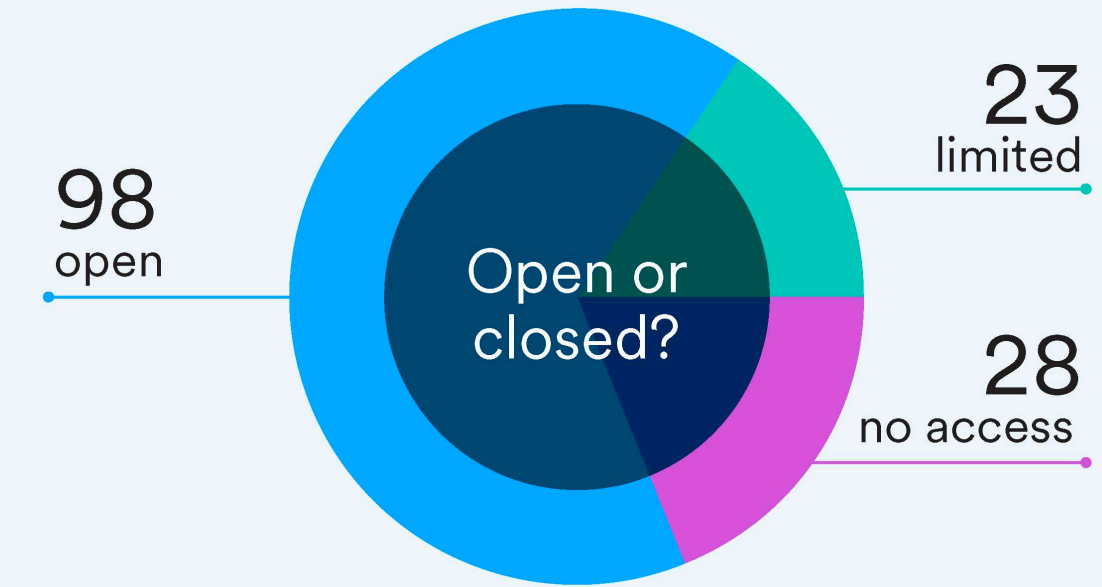
Creating Versatile AI Agents Through Wasm+Rust

Miley Fu, WasmEdge CNCF sandbox

https://github.com/WasmEdge/WasmEdge

https://github.com/LlamaEdge/LlamaEdge

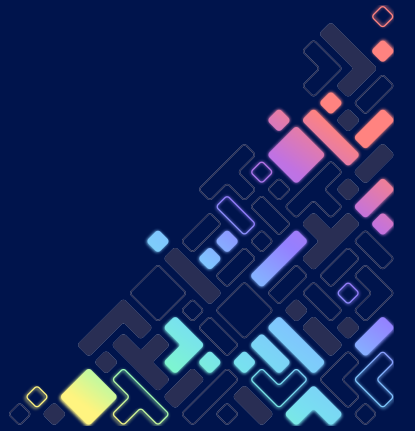
A Move Toward Open-Sourced



2023 newly released models, 65.7%
were open-source (meaning they ca
be freely used and modified by
anyone), compared with only 44.4%
2022 and 33.3% in 2021.

https://hai.stanford.edu/news/ai-index-state-ai-13-charts

Rust+Wasm: A match made in the Cloud



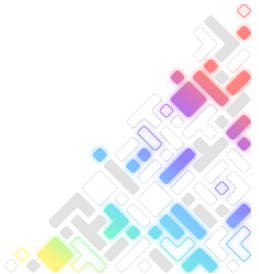
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Rust: A language gaining popularity

- According to SlashData's State of the Developer Nation 25th Edition, the number of Rust developers hit 3.5 million in the third quarter of 2023

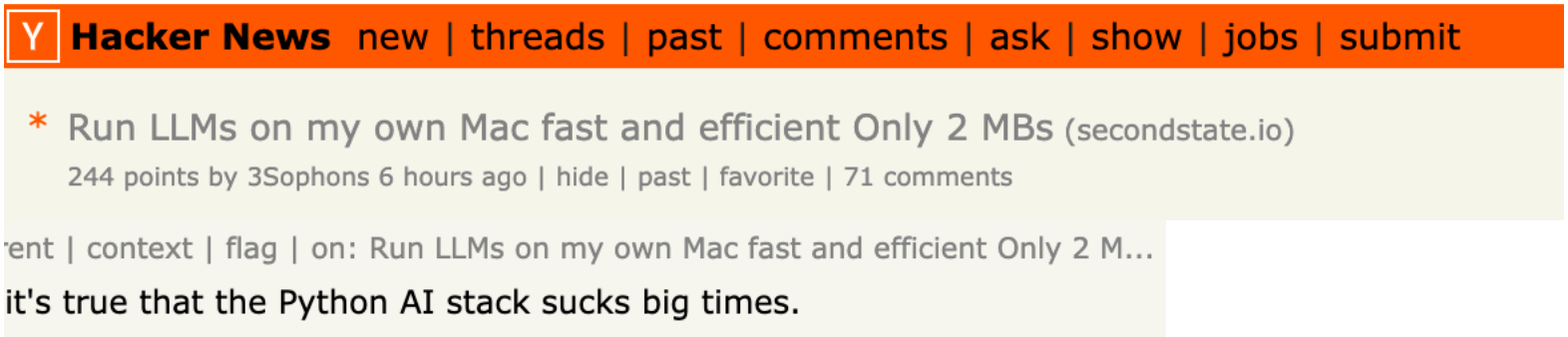
- Stackoverflow survey of 87585 devs, Rust was the language that respondents most admire. 84.66% of the developers who had used Rust saying they want to continue using it in the future

https://survey.stackoverflow.co/



Rust: The Language of Choice for AGI

- Memory safety
- Concurrency
- zero python dependency



Why Wasm on the server side?

Microservices

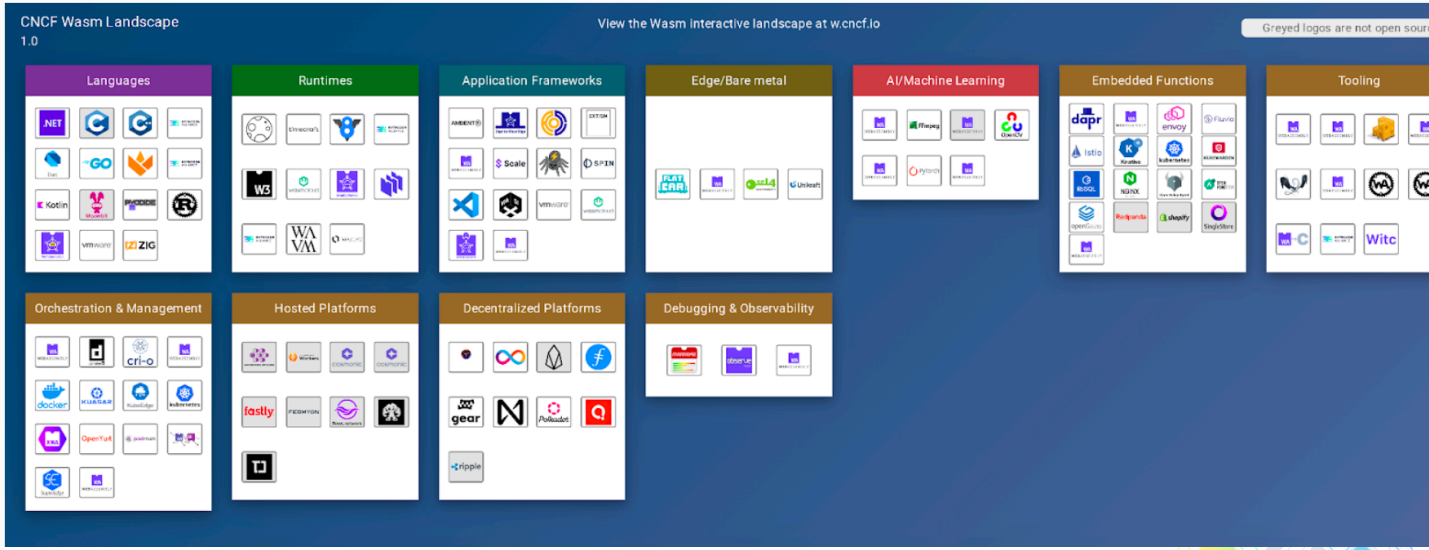
SaaS/UDF

Streaming data

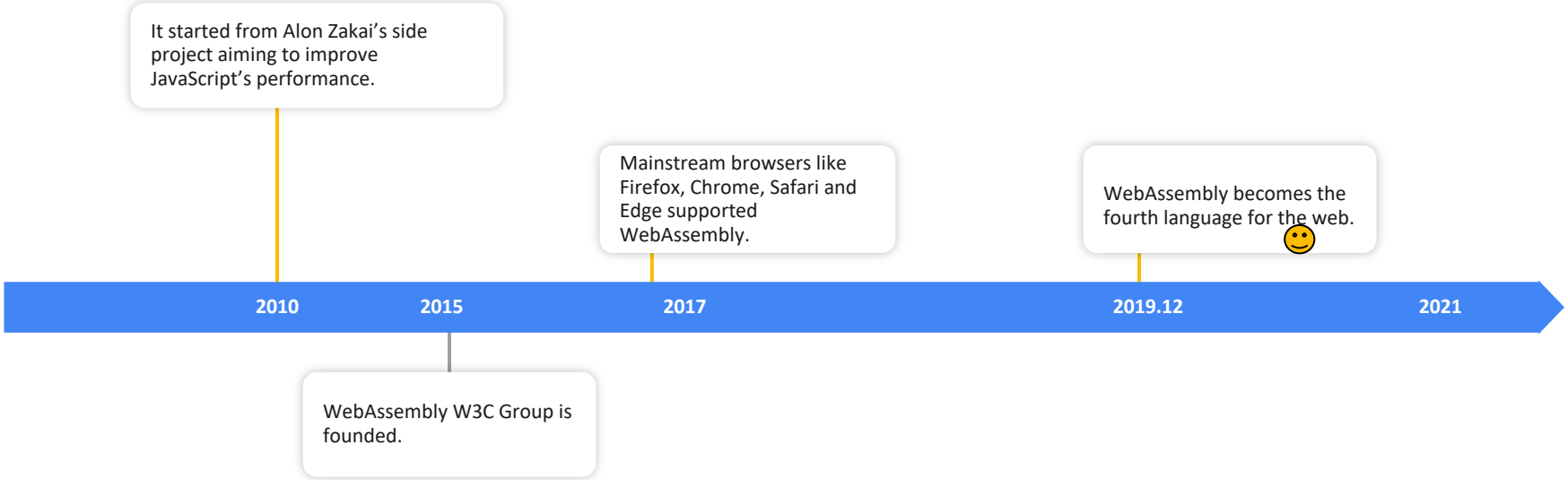
AI inference

The **initial Wasm landscape**, published in time for the WasmCon conference, includes 11 categories and 120 project or products, representing $59.4B in total economic value. The Wasm landscape is divided into two large areas: Dev (application development) and Ops (application deployment).

Wasm landscape *September 6, 2023*



How it started



How it's going

WebAssembly vs Linux Container

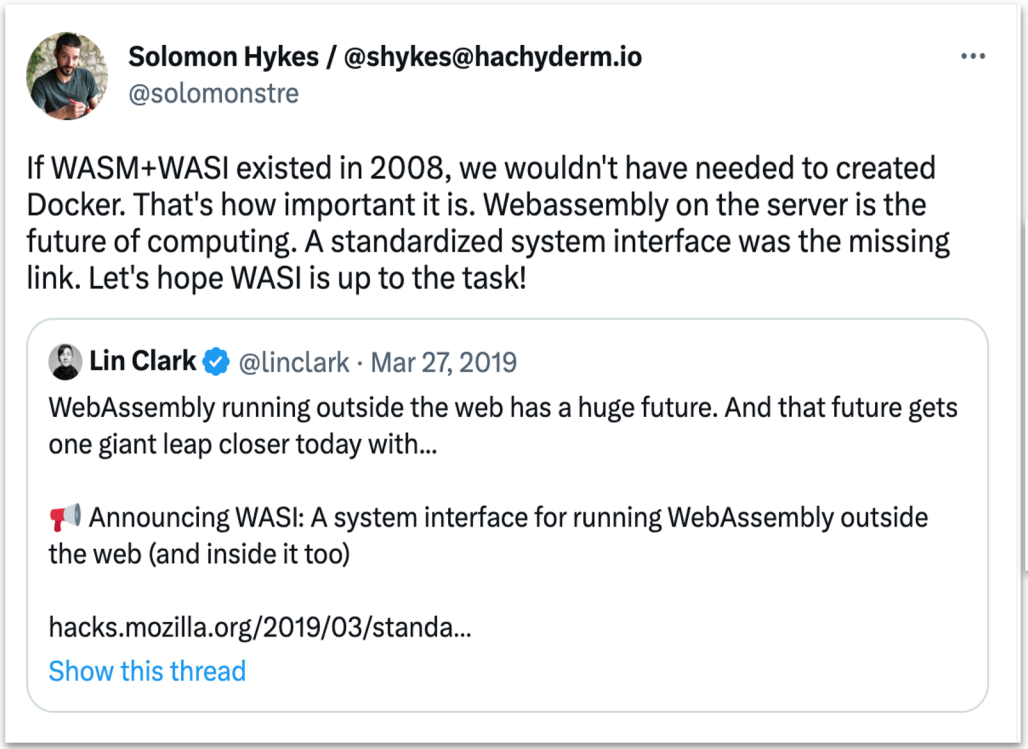
WebAssembly is 100x faster at cold start
WebAssembly is 10% to 50% faster at execution time
AOT compiler could be faster than native
WebAssembly has a much smaller footprint
WebAssembly has a modern security model
WebAssembly is composible

Wasm	Performance	Container
Several MBs	DISK FOOTPRINT	Several GBs
milliseconds	STARTUP PERFORMANCE	seconds
AOT is within 10% of native client	RUNTIME PERFORMANCE	10% to 20% loss to native client

Wasm	Host OS and Apps	Container
Linux, Mac OS X, Windows, RTOS	HOST OS	Linux, Mac OS X, Windows
Yes	MICROKERNEL AS HOST	No
Yes	EMBEDDED OR REAL-TIME OS AS HOST	No
C, Rust, GO, Python, Java etc.	EMBED IN HOST APPLICATIONS	Not embeddable at language SDK level
Access via imported functions from shared libraries	ACCESS HOST OS RESOURCES	Must be supported by Docker itself
Through host functions in shared libraries in the host OS	GPU, TPU, AND SPECIALIZED HARDWARE	Specialized Docker version

https://wasmedge.org/wasm_linux_container/

Don’t take our word for it



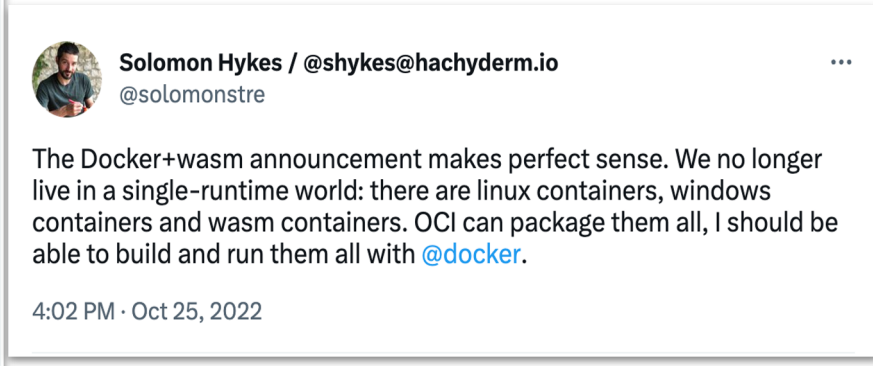
Solomon Hykes / @shykes@hachyderm.io
@solomonstre

If WASM+WASI existed in 2008, we wouldn't have needed to created Docker. That's how important it is. Webassembly on the server is the future of computing. A standardized system interface was the missing link. Let's hope WASI is up to the task!

> **Lin Clark** @linclark · Mar 27, 2019
> WebAssembly running outside the web has a huge future. And that future gets one giant leap closer today with...
>
> 📢 Announcing WASI: A system interface for running WebAssembly outside the web (and inside it too)
>
> hacks.mozilla.org/2019/03/standa...
> Show this thread

WasmEdge was started in 2019

Solomon Hykes was Docker’s founding CTO and the inventor of Linux containers.



Solomon Hykes / @shykes@hachyderm.io
@solomonstre

The Docker+wasm announcement makes perfect sense. We no longer live in a single-runtime world: there are linux containers, windows containers and wasm containers. OCI can package them all, I should be able to build and run them all with @docker.

4:02 PM · Oct 25, 2022

Docker’s landmark partnership with WasmEdge in 2022 brought “Wasm containers” to 10 million Docker Desktop users.

WebGPU and Wasm: abstract away and unify these underlying AI hardware and software stacks

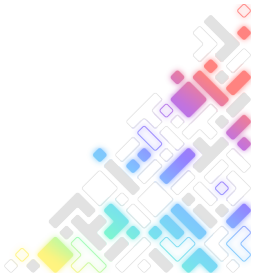
New ways to support cross-platform GPU / AI workloads in container ecosystems, specifically with Docker's support for the WebGPU standard.

WebGPU allows container applications to access the host GPUs and other AI accelerators through a portable API. There is no more need to build Docker images for GPU vendors and their proprietary drivers.

https://sched.co/1c4SM

Why Wasm for AI

- Write once, run everywhere
- Zero python dependency
- Cloud-ready apps. Supported in Docker, containerd, Podman, Kubernetes etc.
- Fast
- Secure: Wasm sandbox is more secure than native binary.
- Polyglot: Support 20+ languages on the frontend

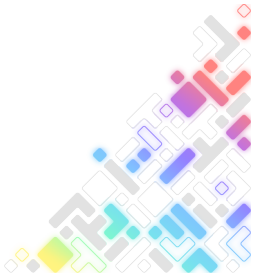


Compare with Python

- Simplicity
  - Self-contained binary app
  - No complex dependencies
  - Easily copy across different devices
  - Ready for cloud deployment
  - Easy to modify and make changes
- 0.1% of the overall size
- 100,00x faster for compute

Compare with C++

- Secure containerized app
  - Ready for cloud deployment
- Single cross-platform binary
  - Automagically take advantage of local hardware accelerators
- Modern languages
  - Rust, Go, Zig etc.

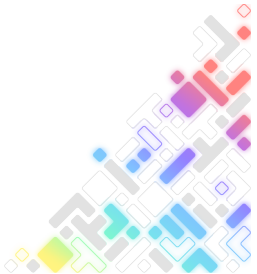


The Power of Portability

Cross-Platform Compatibility:

- run on any platform that has a compatible runtime
- once compiled to Wasm, can be deployed across various environments without modification.

Simplifies Deployment: Developers can build and train their AI models on their preferred platforms and easily deploy them across different hardware and operating systems



How?

wasi-nn

A Bytecode Alliance project

High-level bindings for writing wasi-nn applications

CI passing | crates.io v0.6.0 | npm v0.3.0

Introduction

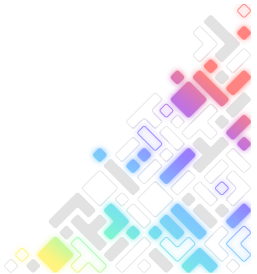
This project provides high-level wasi-nn bindings for Rust and AssemblyScript. The basic idea: write your machine learning application in a high-level language using these bindings, compile it to WebAssembly, and run it in a WebAssembly runtime that supports the wasi-nn proposal, such as Wasmtime and WasmEdge.

- Wasi neural network
- Integrate llama.cpp as a new backend
- Support major AI frameworks
  - OpenVINO
  - TensorFlow
  - Pytorch
- llama.cpp is the inference of LLaMA model in pure C/C++

Benefits of Wasm for AI

- Faster compared to Python.

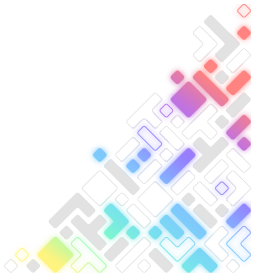
- Portable, more secure and lightweight compared with Python and other native solutions. WasmEdge runtime + portable app is 30M compared with 4G Python and 300MB llama.cpp
- Wasm sandbox is more secure than native binary.
- Container-ready. Supported in Docker, containerd, Podman,and Kubernetes.



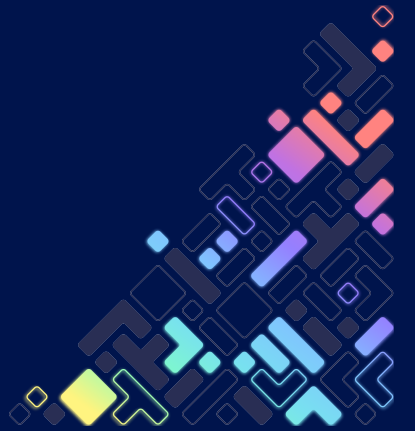
Manage it under containerd / Docker / k8s

Video demo: https://www.youtube.com/watch?v=1qjtxZWXh5w

Try it yourself: https://github.com/second-state/runwasi#llama2-example



Wasm+Rust for AI Agents



THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

LlamaEdge: Run customized and fine-tuned LLMs locally or on the edge

Quick start without any argument

```
bash <(curl -sSfL
'https://raw.githubusercontent.com/LlamaEdge/LlamaEdge/main/run-llm.sh')
```

- It will download and start the Gemma-2b model automatically. Open http://127.0.0.1:8080 in your browser and start chatting right away!

```
bash <(curl -sSfL
'https://raw.githubusercontent.com/LlamaEdge/LlamaEdge/main/run-llm.sh') -
-model llama-2-7b-chat
```

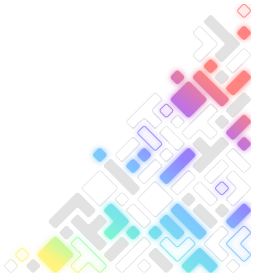

My Machine vs Nvidia Jetson Orin

MacBook Pro

13-inch, 2020, Four Thunderbolt 3 ports

Processor	2 GHz Quad-Core Intel Core i5
Graphics	Intel Iris Plus Graphics 1536 MB
Memory	16 GB 3733 MHz LPDDR4X
Serial number	C02DF0BFML7L
macOS	Sonoma 14.4.1

64GB RAM

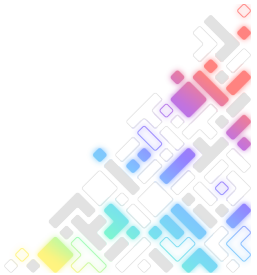


```
wasmedge --dir .:. --nn-preload
default:GGML:AUTO:Llama-2-7b-chat-hf-Q5_K_M.gguf
llama-chat.wasm -p llama-2-chat
```

- wasmedge - Starts the WasmEdge runtime secure sandbox
- --dir - Specifies the model file directory
- --nn-preload default:GGML:AUTO: - Loads the WasmEdge ML plugin (ggml backend) and enables automatic hardware acceleration

And run the 314B Grok!

CPU only, Intel(R) Core(TM) i7-10700
CPU @ 2.90GHz with 128 GB RAM




hydai grok
✓ 1955 16:09:00
1[|0.7%] 3[0.0%] 5[0.0%] 7[0.0%]
2[0.0%] 4[|0.7%] 6[0.0%] 8[0.0%]
Mem[|||||1.95G/126G] Tasks: 103, 211 thr; 1
Swp[0K/0K] Load average: 2.27 1.9
Uptime: 5 days, 22:17:
T RES SHR S CPU% MEM% TIME+ Command
4 8652 5632 S 0.0 0.0 0:00.48 sshd: hydai
8 7472 4864 S 0.0 0.0 0:00.38 -zsh
6 4096 3584 S 0.0 0.0 0:00.00 tmux
4 8524 5504 S 0.0 0.0 0:00.01 sshd: hydai
2 7480 4992 S 0.0 0.0 0:00.11 -zsh
8 4864 3584 R 0.7 0.0 0:01.19 htop
0 9216 7552 S 0.0 0.0 0:00.16 /lib/systemd/syst
M 7280 1664 S 0.0 0.0 0:00.00 (sd-pam)
0 4416 2944 S 0.0 0.0 0:05.21 tmux
0 8944 5504 S 0.0 0.0 0:09.62 -zsh
0 1 zsh | 16:09 | 28 Mar hydai neihu-2
F1Help F2Setup F3SearchF4FilterF5List F6SortByF7Nice

Deploying Llama 2 /Any HuggingFace GGUF Models

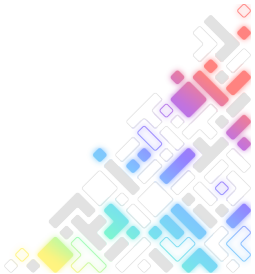
- an example LlamaEdge command with Llama-2-7b as an example
- wasmedge --dir .:. --nn-preload default:GGML:AUTO:llama-2-7b-chat.Q5_K_M.gguf llama-chat.wasm --prompt-template llama-2-chat

https://www.secondstate.io/articles/convert-pytorch-to-gguf/

LlamaEdge: A Dev Platform

https://llamaedge.com/

- Build a single **portable** and deployable app
  - Move code closer to model and data
  - Improve efficiency
  - Simplify development and workflow
  - Improve security
- No Python dependency
- Use Rust or JS to extend LlamaEdge components
- Dev experience that matches the best of OpenAI
  - i.e., highly integrated OpenAI Assistant API



Next steps

- Create a chatbot using the chatbot-ui framework
  - https://second-state.github.io/chatbot-ui/
- Create a RAG application using LangChain or LlamaIndex
  - https://www.langchain.com/
  - https://www.llamaindex.ai/
- Create RAG Discord, Telegram, Slack chatbots on flows.network
  - https://flows.network/ <- This uses Rust and Wasm! 

Make your own API server refering to the existing Components

End user

- Gaianet: a open source knowledge based LLM+ RAG server
https://github.com/GaiaNet-AI

- Each GaiaNet node provides a specialized API service that encapsulates a unique combination of
  - a specialized and fine-tuned LLM (e.g., an LLM that excels in answering questions about the Rust programming language)
  - a domain-specific knowledge base (e.g., knowledge about the WasmEdge project)
  - an inference app that manages the context and history of conversations (e.g., RAG and MemGPT prompt injection)
  - compute resources required to run the LLM app as an API service (e.g., a Nvidia GPU or a Mac M3 device)
- The GaiaNet node API service is fully compatible with the OpenAI JSON spec, and hence each GaiaNet node can function as an alternative backend for OpenAI-based frontends or agents.

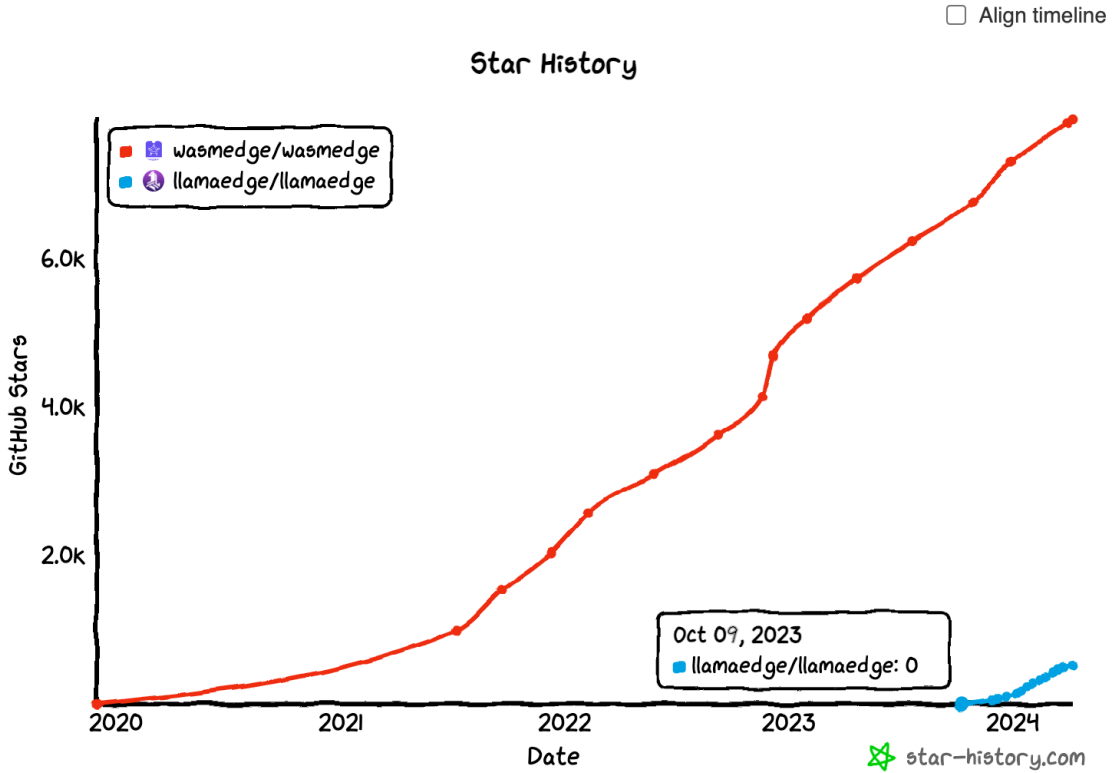
WasmEdge+LlamaEdge

WasmEdge: The lightweight and cross platform AI runtime

https://github.com/WasmEdge/WasmEdge

LlamaEdge: The developer platform for LLM apps

https://github.com/LlamaEdge/LlamaEdge



Calling contributors

is:issue is:open label:"LFX Mentorship"

Clear current search query, filters, and sorts

29 Open ✓ 22 Closed Author Label Pro

- LFX Workspace: Integrate MLX as a new WASI-NN backend [LFX Mentorship] [question]
  #3266 opened last month by guptaaryan16
- LFX Workspace: Integrate Intel Extension for Transformers as a new WASI-NN backend [enhancement] [LFX Mentorship]
  #3260 opened on Mar 6 by grorge123
- LFX Workspace: Integrate burn.rs as a new WASI-NN backend [enhancement] [LFX Mentorship]
  #3252 opened on Mar 4 by derekwin
- LFX Mentorship (Mar-May, 2024): Integrate burn.rs as a new WASI-NN backend [c-WASI-NN] [enhancement] [LFX Mentorship]
  #3172 opened on Jan 23 by hydai
- LFX Mentorship (Mar-May, 2024): Integrate whisper.cpp as a new WASI-NN backend [c-WASI-NN] [enhancement] [LFX Mentorship]
  #3170 opened on Jan 23 by hydai
- LFX Mentorship (Mar-May, 2024): Integrate Intel Extension for Transformers as a new WASI-NN backend [c-WASI-NN] [enhancement] [LFX Mentorship]
  #3169 opened on Jan 23 by hydai
- LFX Mentorship (Mar-May, 2024): Integrate MLX as a new WASI-NN backend [enhancement] [LFX Mentorship]

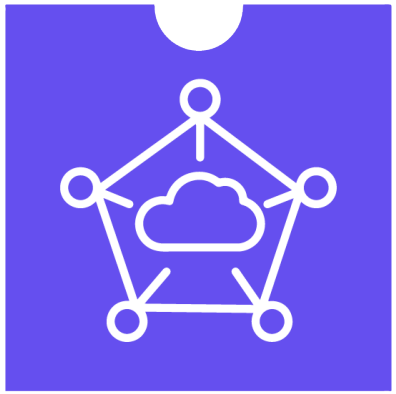
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

LFX Mentorship

Google Summer of Code

37 Open ✓ 57 Closed Author Label

- bug: Different output when running a same binary [bug] [c-AOT] [fuzz-different-behavior] [good first issue]
  #3063 opened 2 weeks ago by XinyuShe
- bug: `fd_pwrite` after setting `fdflags::append` writes from the wrong offset [bug] [c-WASI] [good first issu] [platform-linux]
  #3062 opened 2 weeks ago by yagehu
- Print hard link file number error. [bug] [c-WASI] [good first issue] [platform-windows]
  #3054 opened 3 weeks ago by Userzxcvbvnm
- Fail to set file time ? [bug] [c-WASI] [good first issue] [platform-windows]
  #3052 opened 3 weeks ago by Userzxcvbvnm
- A difference was found in the printing time of a file? [bug] [c-WASI] [good first issue] [platform-windows]



WasmEdgeRuntime

https://github.com/WasmEdge/WasmEdge

CLOUD NATIVE
COMPUTING FOUNDATION

- WASI-NN: https://github.com/bytecodealliance/wasi-nn
- WasmEdge-WASI-NN: https://github.com/second-state/wasmedge-wasi-nn
- LlamaEdge: https://llamaedge.com/ https://github.com/LlamaEdge/LlamaEdge

Stay in Touch! @mileyfu

https://europe2024.gosim.org/

https://foundation.rust-lang.org/events/rust-china-conf/

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

NORTH AMERICA