WRITE ONCE RUN ANYWHERE, BUT FOR GPUS

Hung-Ying, Tai (hydai)
WasmEdge Maintainer, CNCF AMBASSADORS

Agenda

- Quick Demo - LlamaEdge Chat
- Introduction to WebAssembly, WASI-NN, and WasmEdge
- Demo - LlamaEdge Server
- Introduction to LlamaEdge
- Demo - An RAG Application, Gaianet
- The RAG Software Stack
- Demo - Run Lightweight LLM Containers
- Container Integration

- https://github.com/hydai/GOSIM2024

A Quick Demo - LlamaEdge Chat

https://github.com/LlamaEdge/LlamaEdge/tree/main/chat

WebAssembly

- Size is smaller: 1/100 the size of typical LXC images
- Faster startup time
- Near native runtime performance
- Secure by default in a sandbox and very small attack surface
- Completely portable across platforms
  - Distro: Windows, Linux, MacOS
  - Arch: amd64, arm64
- Programming language agnostic
  - Rust, C, C++, Golang, JavaScript, Kotlin, and more!
- Plays well with k8s, service mesh, distributed runtimes etc.

WebAssembly Con’t - Trade Off

- Don’t support GPU, or any specific hardware such as TPU.
- Must learn new language SDKs to create optimized services
- Common libraries need to be ported

There is no way to run AI/LLM workload with pure Wasm!!!

WASI-NN is fixing it

- Define APIs for Wasm to interact with AI/LLM models
  - load, load_by_name, load_by_name_with_config
    - Load models with the given options/configurations
  - init_execution_context
    - Initialize the execution context with the loaded model
  - set_input, get_output, get_output_single
    - Handle the IO
  - compute, compute_single
    - Do the inference job
  - unload
    - Eject models to release the RAM/VRAM usage

- https://github.com/second-state/wasmedge-wasi-nn/

WasmEdge

- A lightweight, secure, high-performance and extensible WebAssembly Runtime
- Plus lots of extensions to empower the Wasm execution environment
  - Support AI inference in llama.cpp, Intel Neural Speed, Tensorflow, OpenVINO, PyTorch etc.
  - Support networking socket and web services
  - Support databases, caches, and DOs
  - Seamlessly integrates into the existing cloud-native infra
  - Support writing wasm programs using JS

- https://github.com/WasmEdge/WasmEdge

Demo - LlamaEdge API Server

https://github.com/LlamaEdge/LlamaEdge/tree/main/api-server

Why LlamaEdge API Server?

- **Very lightweight and fast**
  - Entire runtime + app is less than 30MB
  - Runs well on Raspberry Pi and Jetson devices
  - Full native GPU and hardware accelerator support
- **Single command to install and run as an unprivileged user**
- **Can be managed and orchestrated directly by container tools and k8s**
- **Supports a wide range of LLMs, VLMs, MoE models on Hugging face**
- **Supports a wide range devices and drivers. Runs at native GPU speed**
  - Nvidia CUDA, TensorRT
  - Apple M chips with metal or MLX
  - Advanced CPUs
- **Customizable formatted responses (JSON and function calling)**
- **An efficient and extensible developer platform**
  - RAG, conversation state and function calling can all be built into the API server like OpenAI Assistant API
  - No need for a separate middleware app (e.g., LangChain)

LlamaEdge as a dev platform

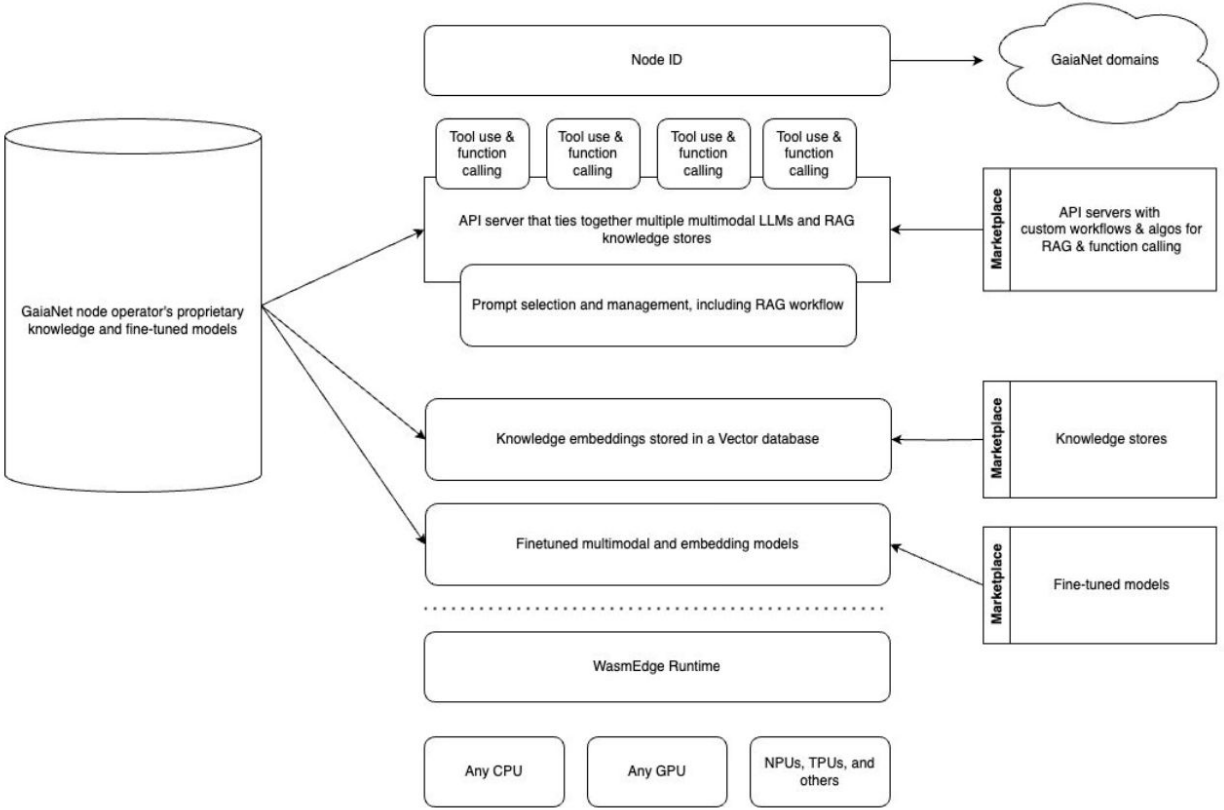
- Build a single **portable** and deployable app
  - Move code closer to model and data
  - Improve efficiency
  - Simplify development and workflow
  - Improve security
- No need for external middleware and containers to orchestrate common LLM app components
- No Python dependency (e.g., LangChain)
- Use Rust or JS to extend LlamaEdge components!
- Dev experience that matches the best of OpenAI
  - i.e., highly integrated OpenAI Assistant API

Demo - An RAG Application, Gaianet

https://github.com/GaiaNet-AI/gaianet-node/

https://llamaedge.com/docs/user-guide/server-side-rag/

LlamaEdge Software Stack



Demo - Run Lightweight LLM Containers

https://wasmedge.org/docs/zh-tw/start/build-and-run/docker_wasm_gpu/

Container integration

- Run the LlamaEdge (and GaiaNet) stack inside a Docker container
  - Requires Docker Nvidia shim
  - Requires CUDA driver in the container
- Use Docker + Wasm + CDI (container-device-interface) for GPU
  - Requires Docker to provide GPU access via CDI
  - Similar to work already done on crun
- Use the LlamaEdge WebGPU backend (WIP)
  - Requires Docker to provide WebGPU API access to containers

THANK YOU

@hydai_tw

@hydai

GOSIM 2024 EUROPE