.UBAfiuba
FACULTAD DE INGENIERÍA

Modelos Locales y uso de APIs

Docente:
Esp. Ing Abraham Rodriguez - FIUBA

Programa de la materia

1. Repaso de Transformers, Arquitectura y Tokenizers.
2. Arquitecturas de LLMs, Transformer Decoder.
3. Ecosistema actual, APIs, costos, HuggingFace y OpenAI.
4. MoEs, técnicas de prompts, evaluación de LLMs.
5. Modelos locales y uso de APIs.
6. RAG, vector DBs, chatbots y práctica.
7. Agentes, fine-tuning y práctica.
8. Generación multimodal.

Inference Servers

Una LLM preentrenada debe ser puesta y servida en modo inferencia para normalmente como modelo de chat de tipo instruct para realizar tareas mediante prompt engineering como:

- Agente conversacional
- Sistemas RAG
- Generación de texto
- Análisis de documentos

Inference Server Benchmarks

Inference Servers

Los inference server suelen abstraer varios pasos para poder servir a una LLM siendo:

- Descarga y manejo de modelos en formato eficiente (Safetensors, ggml).
- Conversión de modelos.
- Quantization.
- El uso de mejoras como FlashAttention.
- Detección y selección de hardware.

Ejemplo conversión

Lectura safetensors

Ejemplo GGML

Implementación de Inference Server

Ollama Engine

Ollama Quantization

Llama.cpp

Llama.cpp es un proyecto en C/C++ cuyo objetivo principal es permitir la inferencia de LLMs y modelos multimodales con configuración mínima y alto rendimiento en una amplia variedad de hardware, tanto **localmente como en la nube**. Consiste en:

- Implementación en C/C++ puro sin dependencias.
- Apple Silicon, optimizado mediante ARM NEON, y los frameworks Accelerate y Metal.
- Soporte para AVX, AVX2, AVX512 y AMX en arquitecturas x86.
- Quantizacion de 1.5 bits, 2 bits, 3 bits, 4 bits, 5 bits, 6 bits y 8 bits para una inferencia más rápida y menor uso de memoria.
- Kernels CUDA personalizados para ejecutar LLMs en GPUs de NVIDIA
- Soporta GPUs AMD mediante HIP y GPUs MTT de Moore Threads mediante MUSA).
- Soporte para backend Vulkan y SYCL.
- Inferencia híbrida CPU+GPU para acelerar parcialmente modelos que superen la capacidad total de VRAM.

Llama.cpp

El proyecto expone una REST API compatible con el formato OpenAI.

Existe un proyecto llamado Llama-cpp-python que expone bindings a Python.

Langchain Docs

LlamaIndex Docs

Build on Android using Termux

Running Llama-2 on AWS with Llama.cpp

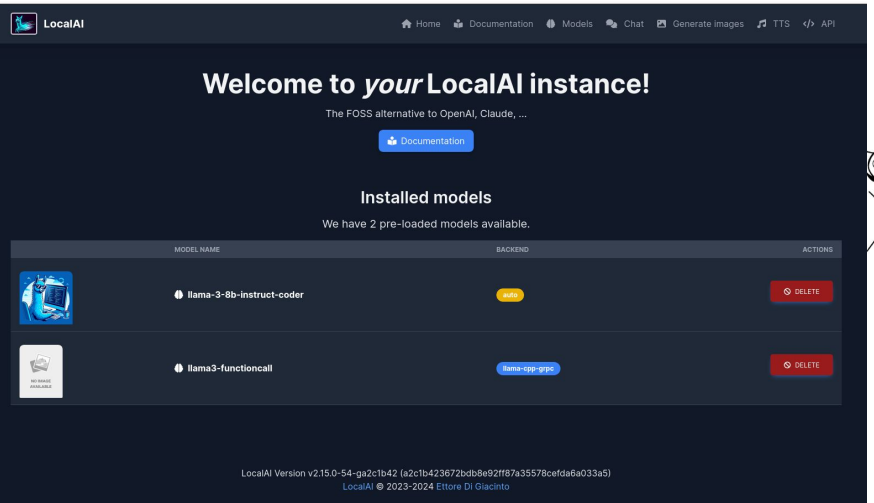
Llama.cpp LM Studio

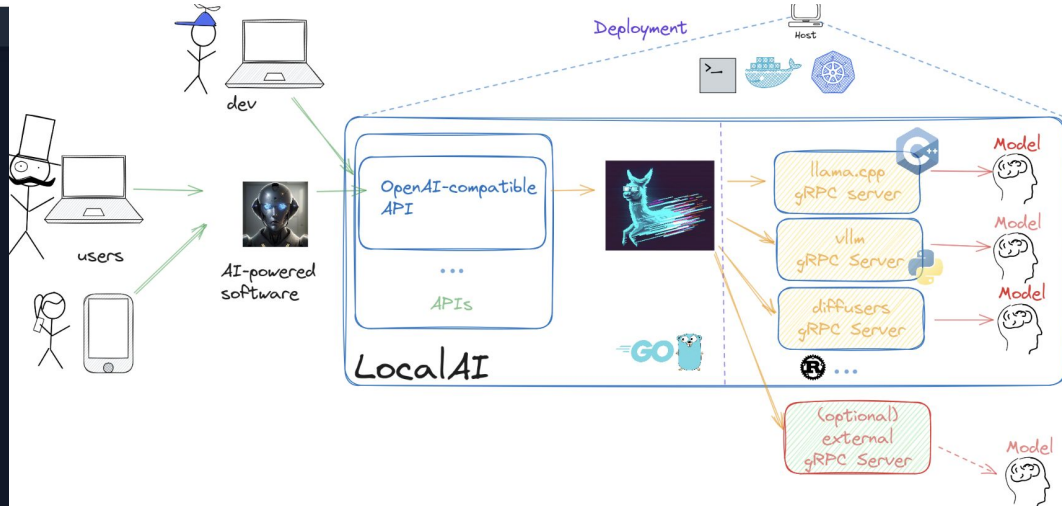
LM Studio es un Wrapper de Llama.cpp provee una GUI para ejecutar modelos locales, permite realizar RAGs locales al agregar documentos.

LocalAI

LocalAI es un proyecto open-source hecho en Go que implementa múltiples plataformas como backend siendo: Llama.cpp, vLLM, entre otras, el objetivo es ofrecer una REST API, GUI y una suite de modelos para ser utilizados de manera gratuita y privada.

Es capaz de integrar modelos de texto, multimodales y diffusion.





Ollama

Ollama es un proyecto open-source realizado en Go que consiste en un **wrapper de Llama.cpp** y enfatiza el uso de LLMs de **manera local** mediante un CLI simple, por ejemplo el comando en terminal descarga y optimiza el modelo Mistral-7B desde huggingface:

- ***ollama run mistral***

Ollama expone una REST API propia y una experimental con el formato de OpenAI.

Compatible con GPUs NVIDIA y AMD

Langchain provee documentación para integrar Ollama.

Scaling Local Inference to the Cloud

How to Deploy local LLM using Ollama Server and Ollama Web UI on Amazon EC2

Running Ollama on Raspberry Pi

vLLM

vLLM es un proyecto hecho en Python presentado en el paper "Efficient Memory Management for Large Language Model Serving with PagedAttention", implementa:

- Gestión eficiente de la memoria de keys y values con PagedAttention.
- Quantization: GPTQ, AWQ, INT4, INT8 y FP8.
- Kernels CUDA optimizados, incluyendo integración con FlashAttention y FlashInfer.
- Chunked prefill
- Integración fluida con modelos populares de Hugging Face
- Diversos algoritmos de decodificación como parallel sampling y beam search.
- Salidas en streaming.
- REST API compatible con OpenAI.
- Soporte para GPUs NVIDIA, CPUs y GPUs AMD, CPUs Intel, TPU.
- Prefix caching.
- Soporte multi-LoRA.

vLLM Serving

vLLM está listo para servir en ambientes productivos, ofrece múltiples guías para configurarlo:

Docker Deployment

Kubernetes Deployment

Deployment with Nginx Load Balancer

Production Metrics

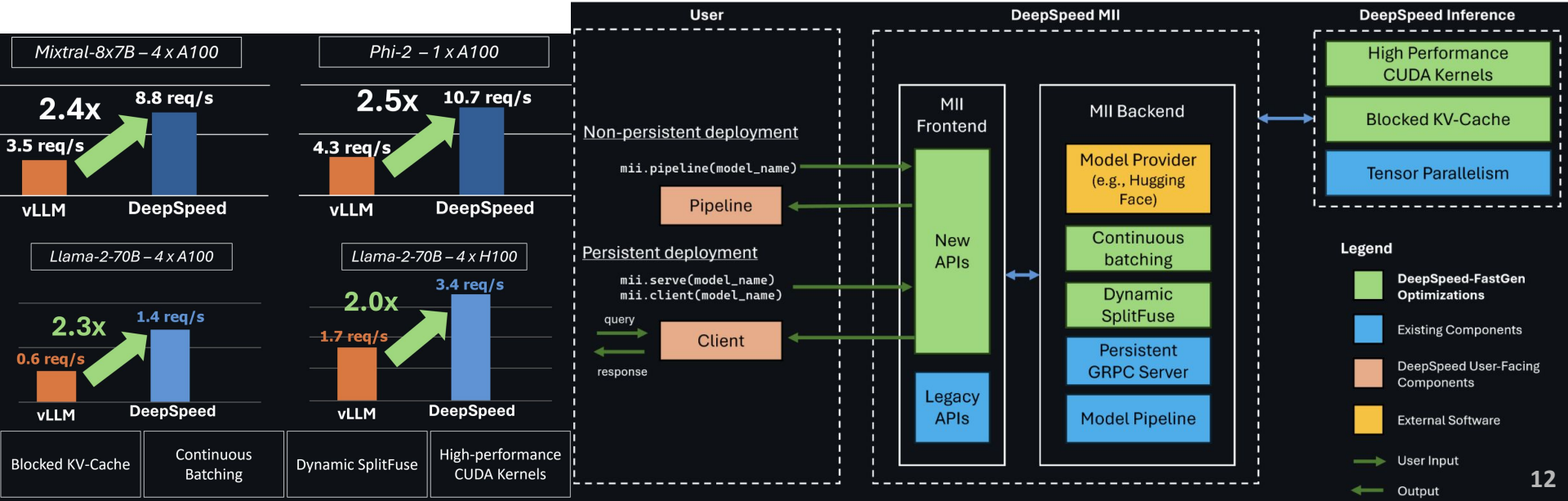
Triton Deployment

BentoML Deployment

Supported Models

DeepSpeed-MII

DeepSpeed-MII es un proyecto de Microsoft realizado en Python que promete ofrecer mayor rendimiento que vLLM, gracias a 4 Tecnologias. Esta basado en DeepSpeed un wrapper ligero de Pytorch que provee optimizaciones para LLMs.



NVIDIA Triton Inference Server

Triton, es un software open-source de la plataforma de NVIDIA que permite deploy cualquier modelo de múltiples frameworks como : TensorRT, Pytorch, ONNX, entre otros.

Ejecución concurrente de modelos
Pipelines de modelos usando Ensembling o Scripting de Lógica de Negocio (BLS)
APIs REST y GRPC.
Una API en C y una API en Java permiten la aplicación para casos de edge computing y otros usos en proceso.
Métricas y estadísticas del servidor y recursos.

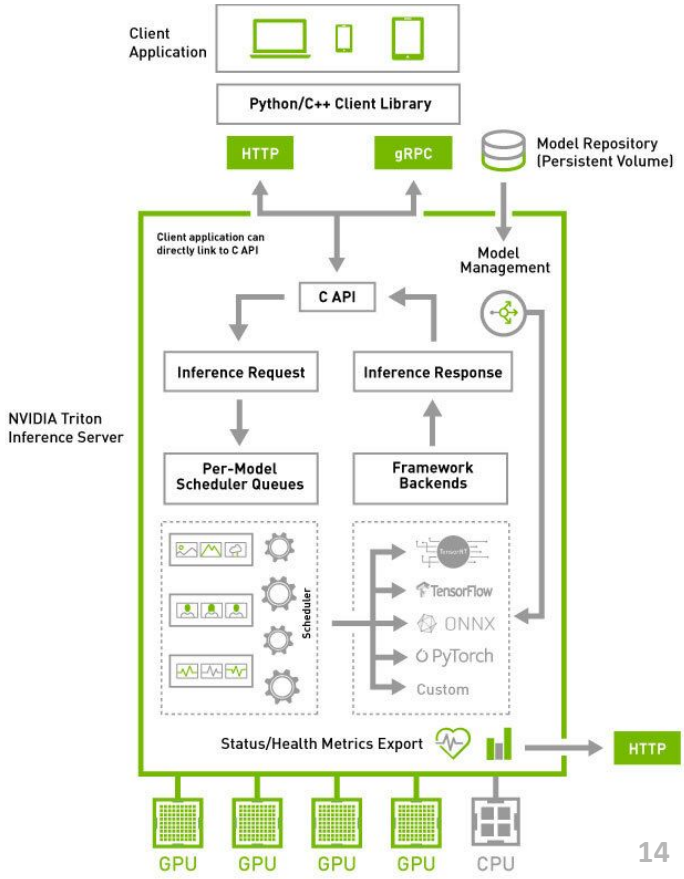
NVIDIA Triton Inference Server

[Deploying with Triton]

[Triton Container]

[Triton Tutorials]

[Kubernetes Deployment]



Jetson-inference

NVIDIA Jetson cuenta con una plataforma para realizar inferencia on Edge, para lograr configurar el sistema embebido se debe seguir la guia de jetson-inference.

Text-generation

Gemma.cpp

Gemma.cpp es un proyecto open-source de Google, similar a Llama.cpp, sin embargo está focalizado en la familia de modelos Gemma.

Cuenta con Python Bindings.

No expone una REST API.

ROCm (AMD)

AMD es soportado por varios inference servers como vLLM, para ello véase la documentación.

ROCm inference optimization.

Making AMD GPUs competitive for LLM inference

Práctica

Preguntas?