# What is vLLM?

This project is from UC Berkeley’s students, who have a passion to optimize serving performance in LLMs. Many systems spend a lot of resources on serving LLMs. However, it has a poor response time when using a simple method to deploy it. As a result, vLLM’s team proposes a new method to solve this issue by using the OS’s virtual memory design, which could improve LLM serving performance around 24 times while using half the memory of the GPU compared with the traditional method. To integrate into your system, vLLM provides a simple interface that lets machine learning engineers (MLE) develop it via a Python interface, which you could integrate into your system without using fancy packages or dependencies.

# What is Flash Attention?

Flash Attention is a method to improve the efficiency of transformer models, in particular [large language models (LLMs)](https://www.hopsworks.ai/dictionary/llms-large-language-models), helping reduce both [model training](http://www.hopsworks.ai/dictionary/model-training) time and inference latency. Inference latency is, in particular, a challenge for LLMs, and flash attention has become a key technique that enables your LLM applications to respond faster.

vllm (Very Large Language Models) is a specialized library designed to efficiently handle and run large-scale language models. It is particularly beneficial for managing memory usage and optimizing inference performance for models that require significant computational resources.

# Benefits of Using vllm

1. Efficient Memory Utilization

vllm allows for efficient utilization of GPU memory, making it possible to run very large models on hardware with limited resources.

2. High-Performance Inference

The library is optimized for high-performance inference, enabling faster generation of text while maintaining high-quality outputs.

3. Easy Integration

vllm integrates seamlessly with popular libraries such as Hugging Face's transformers, allowing users to leverage pre-trained models and fine-tuning capabilities.

4. Customizability

Users can easily customize sampling parameters and other configurations to suit specific use cases, enabling fine control over the model's behavior.

# How to integrate it in code

**Importing vllm**

To use vllm, you need to import the necessary classes and functions from the library:

from vllm import LLM, SamplingParams

# Initialization and Model Loading

In the ReviewSnippets class, vllm is used to load and configure the language model. Here’s how it is done:

class ReviewSnippets:

def \_\_init\_\_(self):

self.df\_pid = None

self.features\_df = None

self.llm = None

self.sampling\_params = None

self.tokenizer = None

load\_dotenv("/home/ankur/projects/llm\_test/Akshay/.env")

def load\_model(self):

model\_name = "/home/ankur/projects/llm\_test/Akshay/fine\_tuning/llama3-8b\_bvr\_v3"

hf\_auth = os.getenv("HF\_AUTH")

Initialize the LLM with the model name and GPU memory utilization parameter

self.llm = LLM(model=model\_name,

gpu\_memory\_utilization=0.9,

max\_model\_len=2048)

Set up sampling parameters for text generation

self.sampling\_params = SamplingParams(

temperature=0.7,

top\_p=0.95,

max\_tokens=2048,

)

Load the tokenizer

self.tokenizer = AutoTokenizer.from\_pretrained(model\_name, token=hf\_auth)

self.tokenizer.eos\_token = ""

# Explanation of Key Components

1. LLM Class

The LLM class from vllm is used to load and manage the language model. It takes several parameters:

- model: The path or name of the model to be loaded.

- gpu\_memory\_utilization: Specifies the fraction of GPU memory to be used, ensuring efficient memory management.

- max\_model\_len: Sets the maximum length of the model's input.

2. SamplingParams Class

The SamplingParams class allows setting parameters for text generation:

- temperature: Controls the randomness of predictions by scaling the logits before applying softmax.

- top\_p: Implements nucleus sampling, where the model considers only the smallest set of tokens whose cumulative probability is greater than p.

- max\_tokens: Limits the number of tokens generated.

Memory Efficiency

By setting gpu\_memory\_utilization to 0.9, the ReviewSnippets class ensures that the model uses GPU memory efficiently without overloading the hardware, which is crucial for handling large models.

Custom Sampling

The SamplingParams class provides fine control over the text generation process, allowing for high-quality and contextually appropriate snippets to be generated from the reviews.

Seamless Integration

Using vllm alongside Hugging Face’s transformers allows the ReviewSnippets class to leverage state-of-the-art models and tokenizers, enhancing the performance and accuracy of generated text.

Conclusion

vllm is a powerful library for managing very large language models, offering benefits like efficient memory usage, high-performance inference, and easy integration with existing libraries. The ReviewSnippets class demonstrates how vllm can be utilized to load and configure a large language model, customize text generation parameters, and optimize memory usage, making it a valuable tool for developing applications that require advanced language model capabilities.