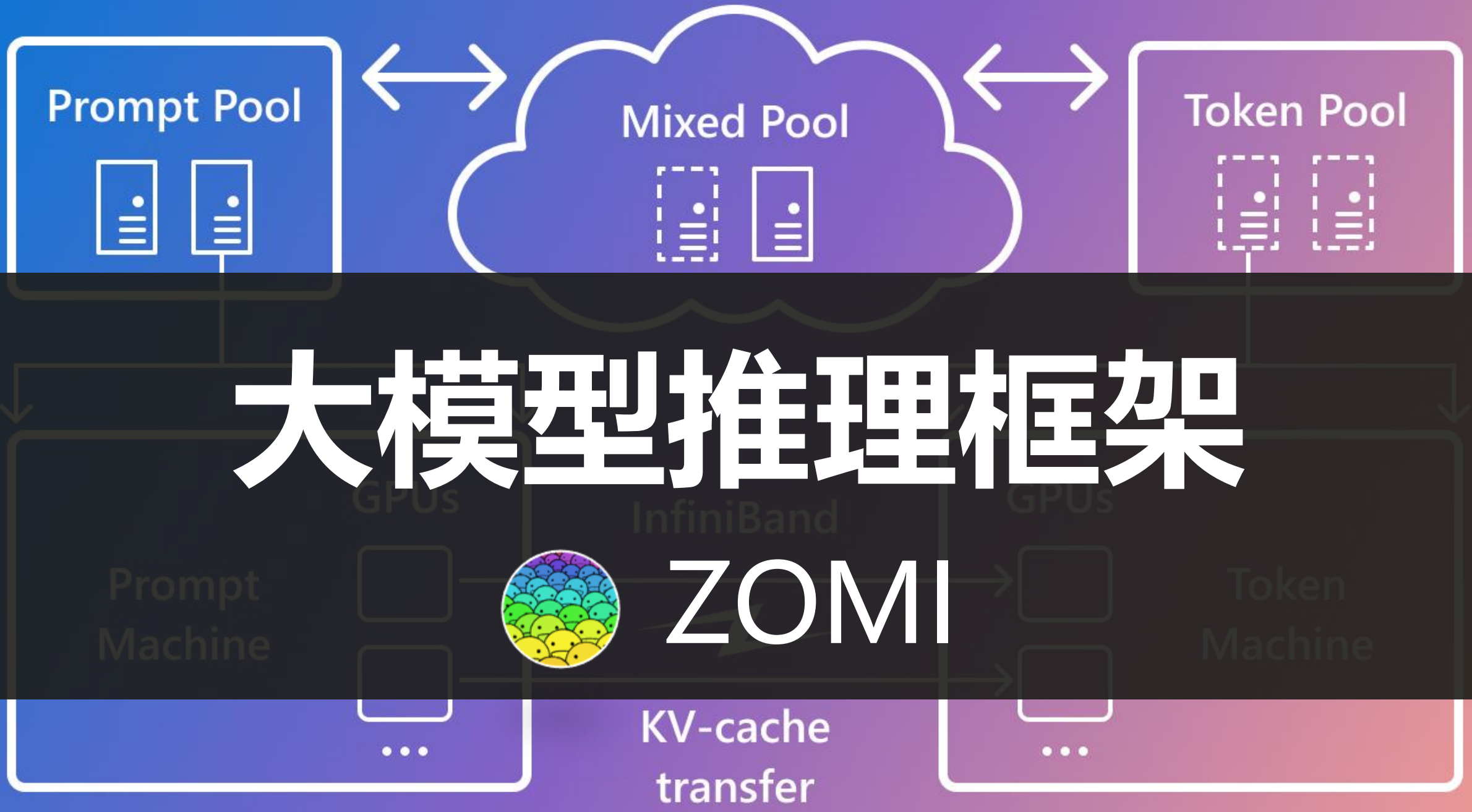
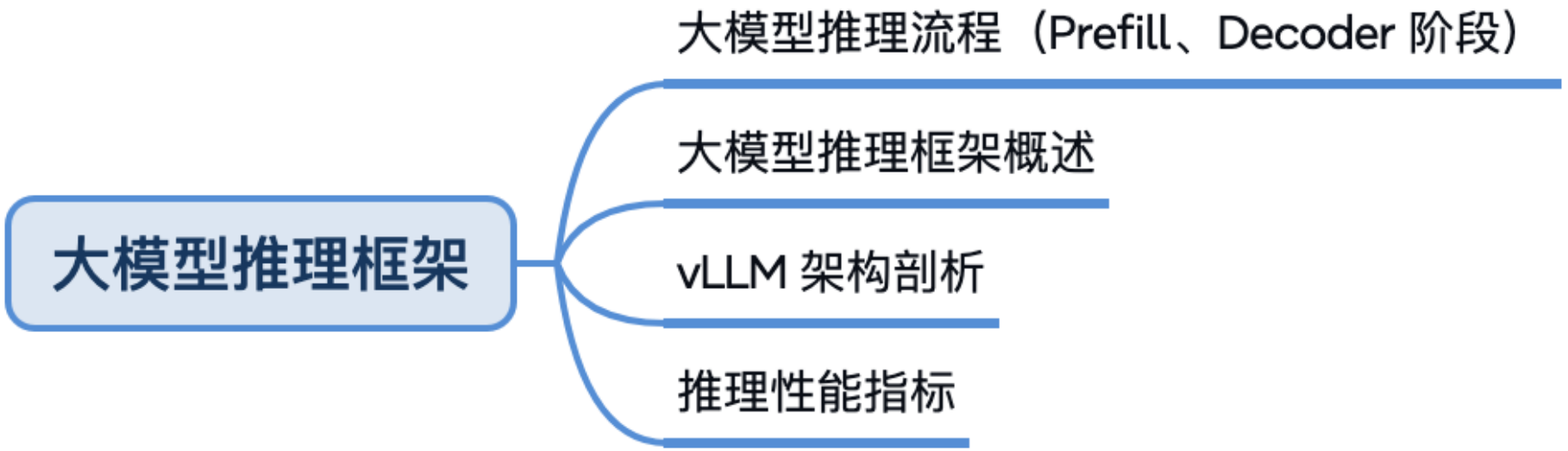


大模型推理框架

ZOMI

大模型推理


大模型推理框架
大模型推理流程（Prefill、Decoder 阶段）
大模型推理框架概述
vLLM 架构剖析
推理性能指标

目录

Question

1. 到底是大模型推理框架？还是大模型推理引擎？还是大模型推理服务呢？

Question

1. 大模型推理加速的目的是什么？用什么目标牵引设计一个好的大模型推理框架？

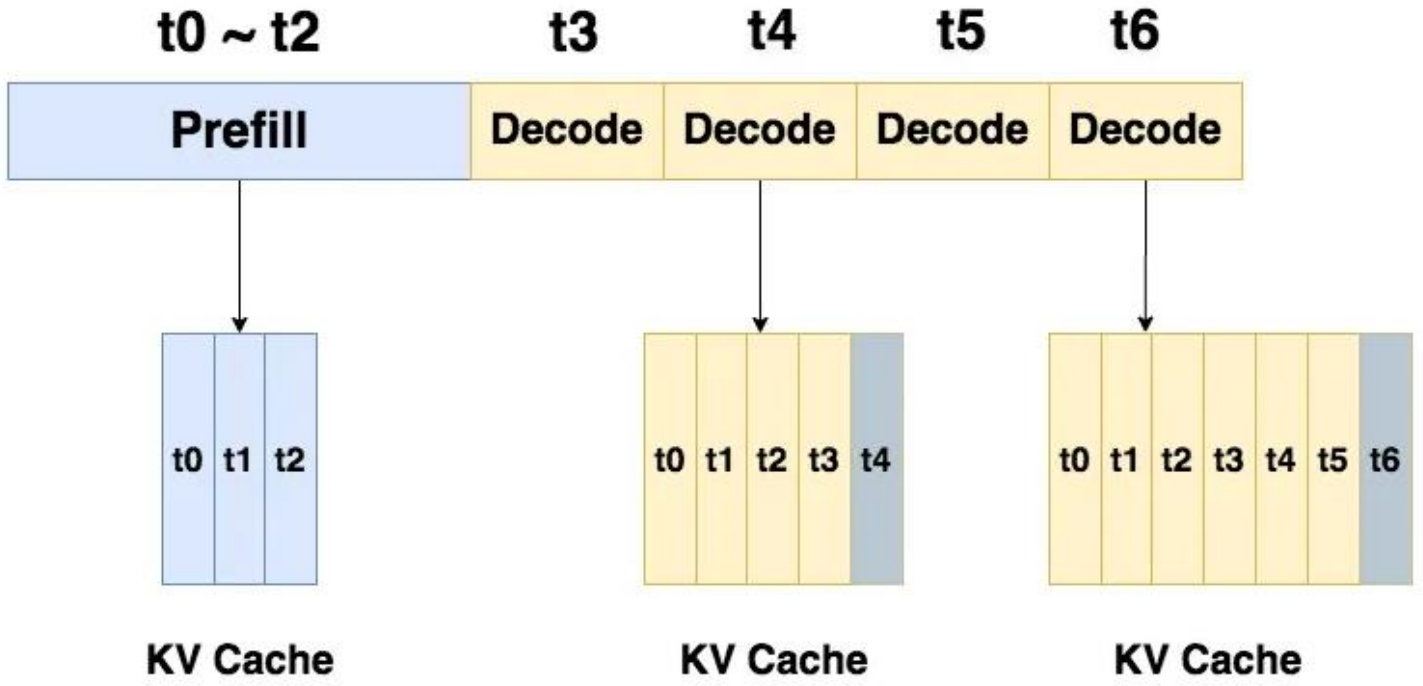
2. 大模型推理加速目标：高吞吐、低延迟。

01

有哪些大模型推理框架

LLM 推理框架

开源大模型推理框架从 2024 年由 “非常粗糙，但是能用” 阶段迈入到了 “好用，稍微有那么点粗糙” 的阶段，正在快速发展



Question

1. 为什么说从长远的一个视角看如今的开源引擎实际上都还是比较粗糙?

- 社区充满活力，快速探索中
- LLM 新技术更新太快，推理框架承受太多
- AI 推理框架要支持日新月异新模型和新硬件
- 可能涉模型结构、计算策略、调度策略、存储策略、硬件加速等

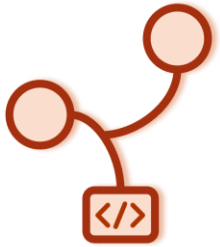
主流推理框架现状

TGI

Text Generation Inference

Documentation

sglang



SGL

VLLM

LLM

LLMDeploy

LMDeploy

厂商自配的推理框架

MindIE

https://www.hiascend.com/software/mindie

TensorRL-LLM

02

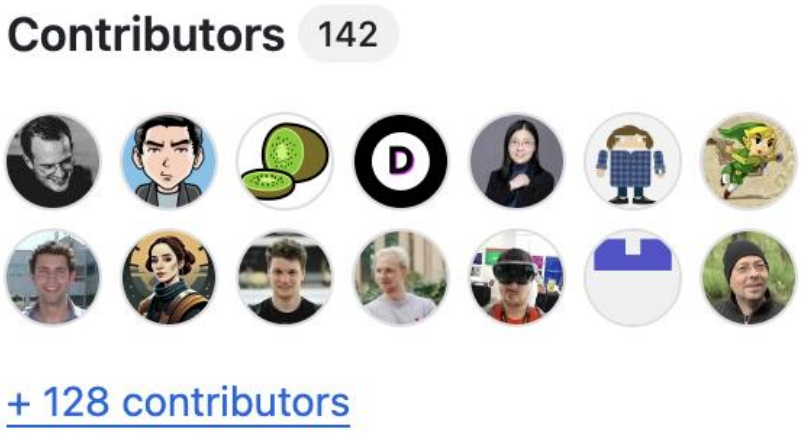
Huggingface TGI

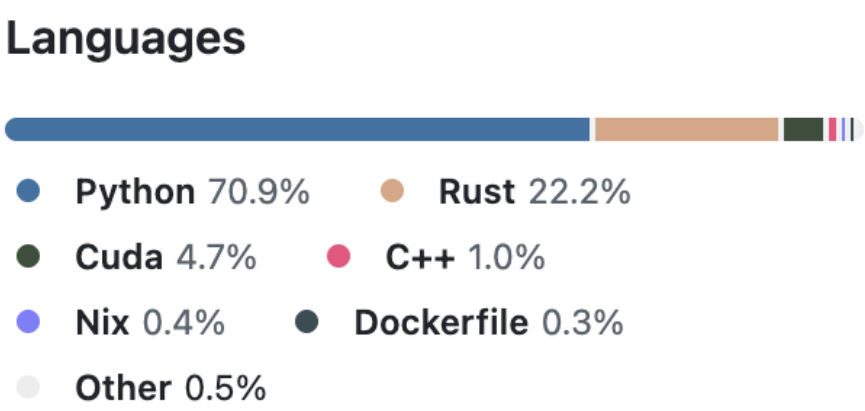
TGI

text-generation-inference Public

Watch 102 Fork 1.1k Star 9.1k

https://github.com/huggingface/text-generation-inference





TGI

1. tgi 宣称使用 pagedattn，吞吐性能一般，预分配显存时浪费严重， batch_size 无法增长
2. tgi 的 cpu 和 gpu 调度串行模型，导致 cpu 调度计算时候 gpu 闲置，导致吞吐性能变差
3. tgi 使用 rust 来实现调度逻辑，导致广大 python 开发者效果无法快速上手
4. 开发人员投入不够，版本更新太慢，大模型推理新功能上线慢

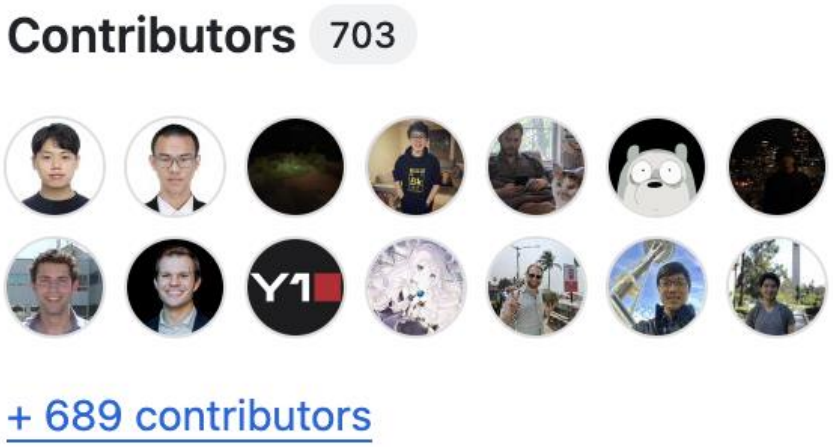

02
vLLM

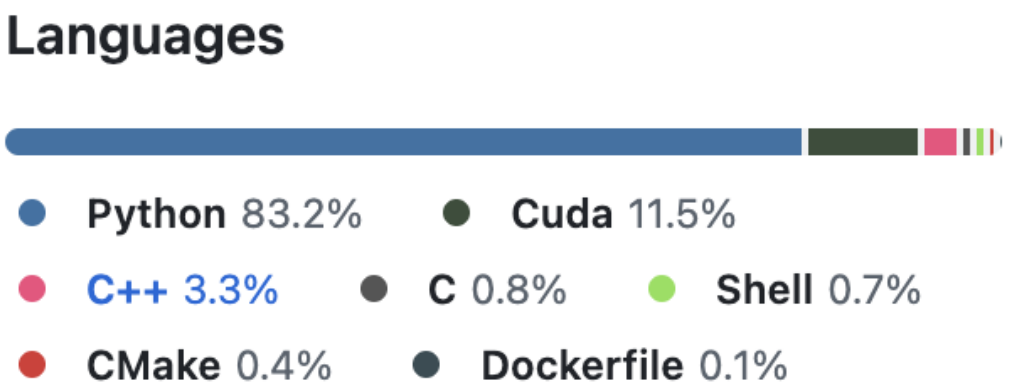
VLLM

vllm Public

Sponsor | Watch 249 | Fork 4.6k | Star 30.6k

https://github.com/vllm-project/vllm





VLLM

原本只作为 pagedattn 开源实现，但发展到今天已经成为 llm 推理框架的标杆了

优势：

1. UCB，有着大量且稳定的开发者，作者基本为在读博士生，github 上 Contributors 最多，vllm 开发人员投入最高。因此 vllm 对模型支持和硬件支持都是最完善，以及各种功能也往往是最齐全的
2. 社区活跃度最高，github 上 issue 和 pr 都很多。大量 paper 都是以 vllm 作为 baseline 来开发 demo，因此各种新技术的引入 vllm 是具有更大优势的
3. 基础的各种优化以及进阶的权重量化、kv压缩、speculate decode、chunked prefill、promot cache、constrainted decoding等功能都是完备的

VLLM

原本只作为 pagedattn 开源实现，但发展到今天已经成为 llm 推理框架的标杆了

缺点：

1. v0.6.0 前 vllm 中 cpu 和 gpu 调度串行，导致 cpu 计算时候 gpu 闲置，让吞吐变差
2. 合入功能太多，vllm 高负载下 gpu 利用率甚至会降低到50%左右~
3. 代码臃肿复杂，导致定制性的二次开发变得非常困难

03

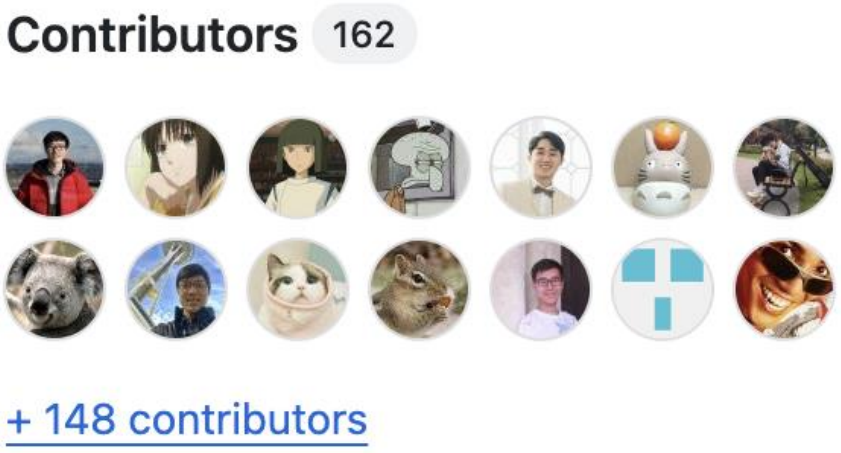
SGLang

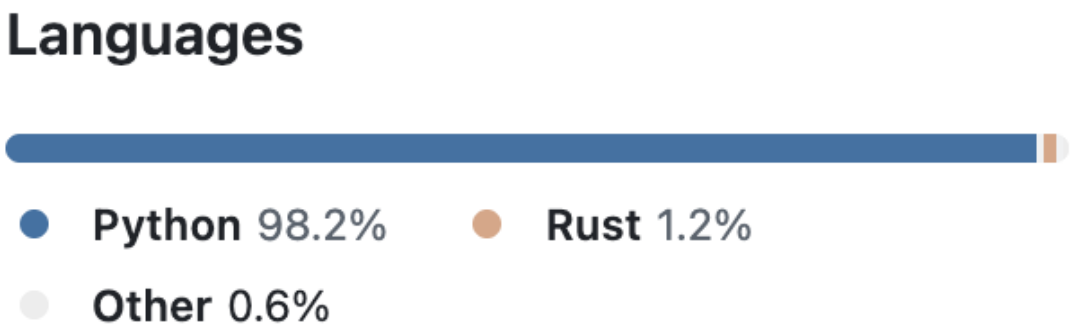
sglang

sglang Public

Watch 58 Fork 522 Star 6.2k

https://github.com/sgl-project/sglang





sglang

借鉴 lightllm 推理框架，优势：

1. sglang 吞吐性能最优，通过多进程 zmp 传输中间数据来 cover 掉 cpu 开销，高负载下 gpu 利用率可以到80%以上
2. sglang 代码可拓展性很高，主流功能都有支持的情况下，代码比 vllm 清晰简单很多，对于二次开发来说是很重要的
3. sglang 开源维护者积极地回复 issue。而且开发节奏快，有些功能还不太完善地方下一个版本基本马上就更新了

04

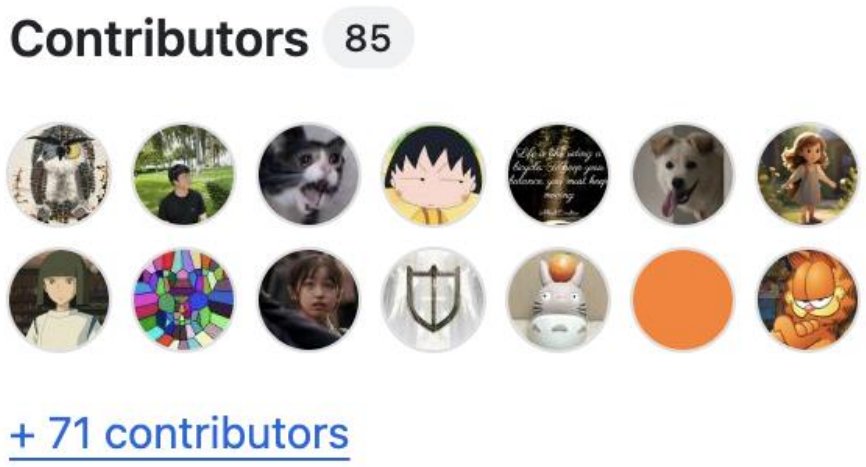
LMDeploy

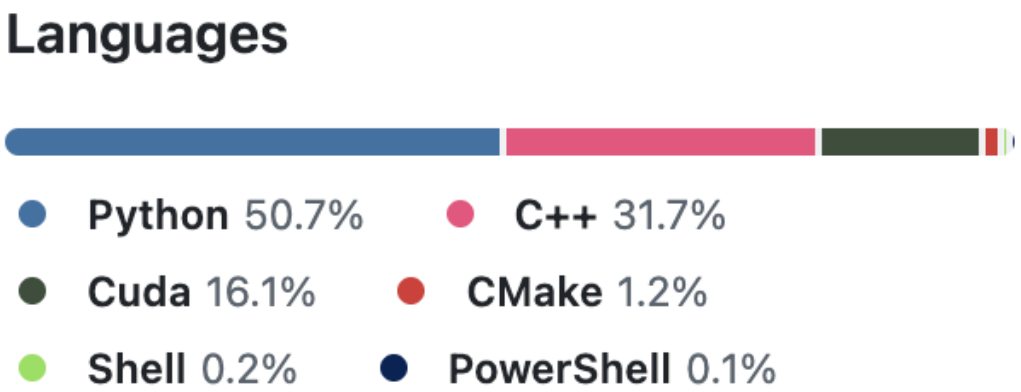
lmdepoloy

lmdeploy Public

Watch 38 Fork 428 Star 4.7k

https://github.com/InternLM/lmdeploy





lmdepoloy

上海人工智能实验室团队开发，优势：

1. 相比 vllm 和 sglang 的 python 实现，lmdeploy 调度和执行 runtime 代码使用了 C++ 实现
2. cpu 调度策略优，高负载下 gpu 利用率稳定在 95%
3. 对多模态模型支持很好，支持大量的多模态模型
4. 对国内GPU厂商的硬件支持较好

lmdepoloy

开发人员太少，功能比 vllm 和 sglang 来说还是少了挺多重要的功能

纯 python 实现 LLM 推理框架够用，推理框架引入 C++ 和 rust 是否利大于弊不好说

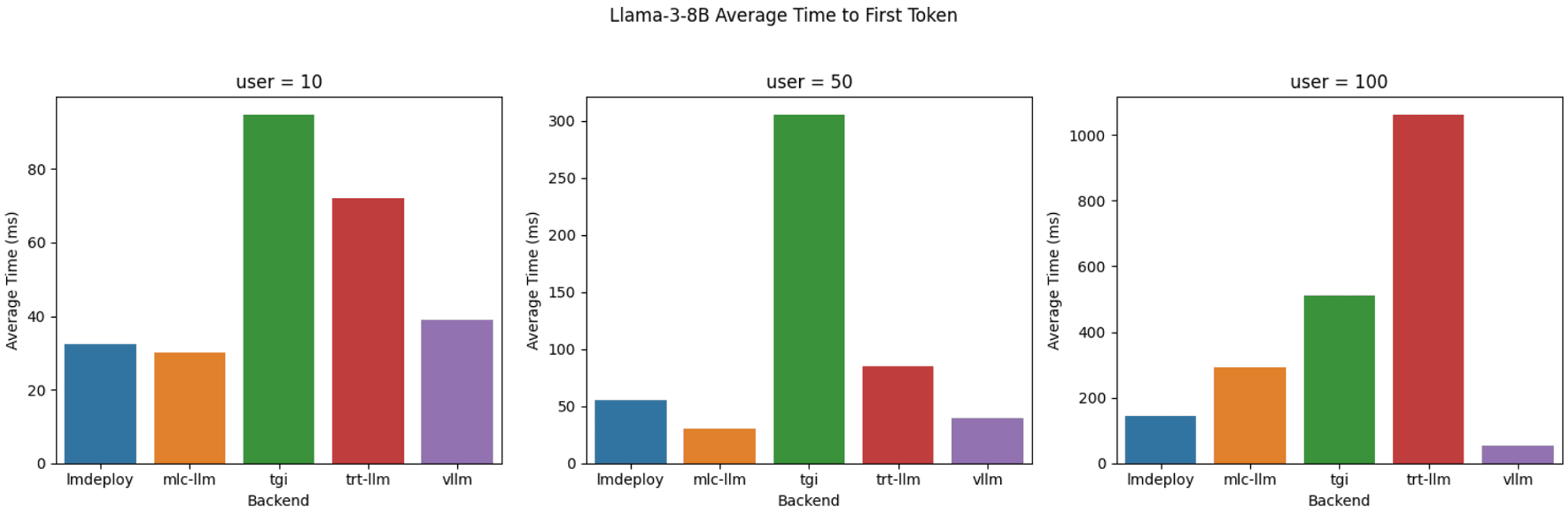
AI 场景 python 开发效率比其他语言优势大太多

05

性能对比

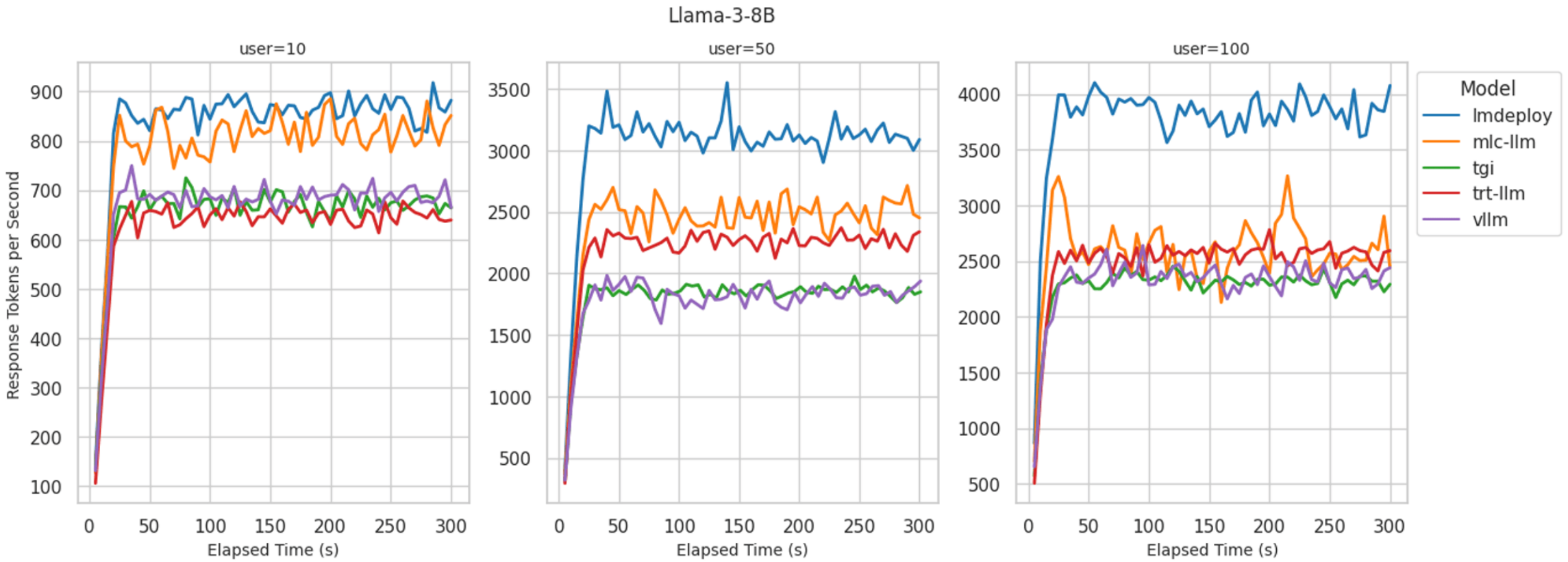
性能分析

Llama 3 8B: Time to First Token (TTFT) of Different Backends



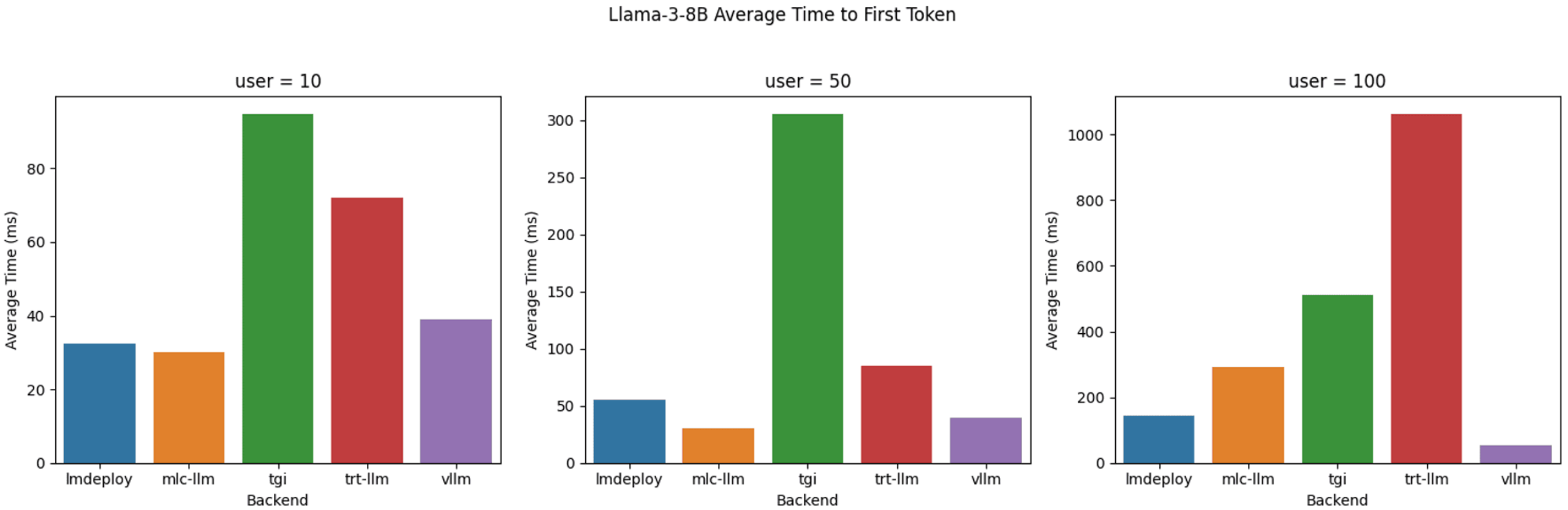
性能分析

Llama 3 8B: Token Generation Rate of Different Backends



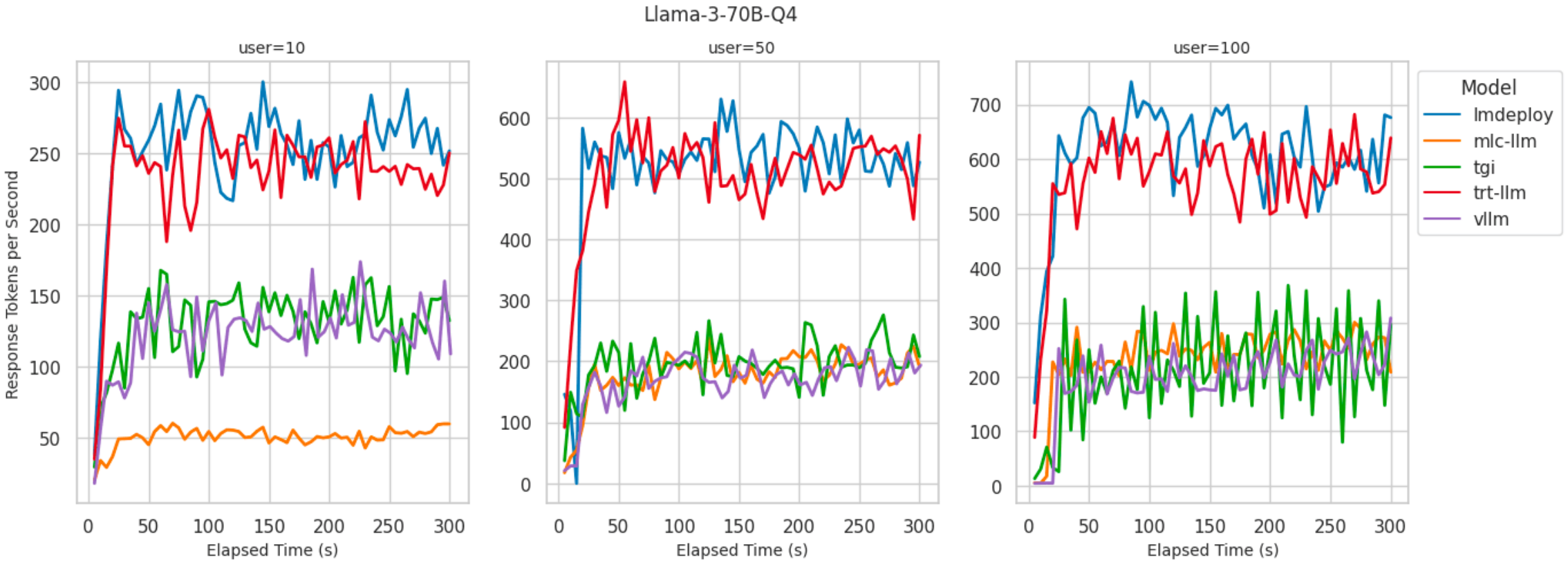
性能分析

Llama 3 70B Q4: Time to First Token (TTFT) of Different Backends



性能分析

Llama 3 70B Q4: Token Generate Rate of Different Backends



06

小结与思考

LLM 推理框架分析

	LMDeploy	TensorRT-LLM	vLLM	MLC-LLM	TGI
Quantization	Supports 4-bit AWQ, 8-bit quantization, and 4-bit KV quantization.	Supports quantization via modelopt, and note that quantized data types are not implemented for all the models.	Users need to quantize models with AutoAWQ or find pre-quantized models on HF. Performance is under-optimized.	Supports 3-bit and 4-bit group quantization. AWQ quantization support is still experimental.	Supports AWQ, GPTQ, and bits-and-bytes quantization.
Models	About 20 models supported by TurboMind engine.	30+ models supported	30+ models supported	20+ models supported	20+ models supported
Hardware	Only optimized for Nvidia CUDA	Only supports Nvidia CUDA	Nvidia CUDA, AMD ROCm, AWS Neuron, CPU	Nvidia CUDA, AMD ROCm, Metal, Android, IOS, WebGPU	Nvidia CUDA, AMD ROCm, Intel Gaudi, AWS Inferentia

Question

1. 这么多 LLM 推理框架应该怎么选?

- 不具备二次开发能力，想要快速部署开箱即用，vllm 是最好选择
- 国内 GPU 厂商硬件或者是部署多模态模型，lmdeploy 是不错选择
- 对性能有极致追求，团队具备很强二次开发能力，sglang 最好选择

Thank you

把AI系统带入每个开发者、每个家庭、每个组织，构建万物互联的智能世界

Bring AI System to every person, home and organization for a fully connected, intelligent world.

ZOMI

GitHub https://github.com/chenzomi12/AIFoundation

1. https://www.omrimallis.com/posts/understanding-how-llm-inference-works-with-llama-cpp/
2. https://www.databricks.com/blog/llm-inference-performance-engineering-best-practices
3. SARATHI: Efficient LLM Inference by Piggybacking Decodes with Chunked Prefills