PyTorch DAY
CHINA 2025

Co-hosted by

BAAI 智源研究院 PyTorch

SGLang: An Efficient Open-Source Framework for Large-Scale LLM Serving

Liangsheng Yin @ LMSYS & SJTU

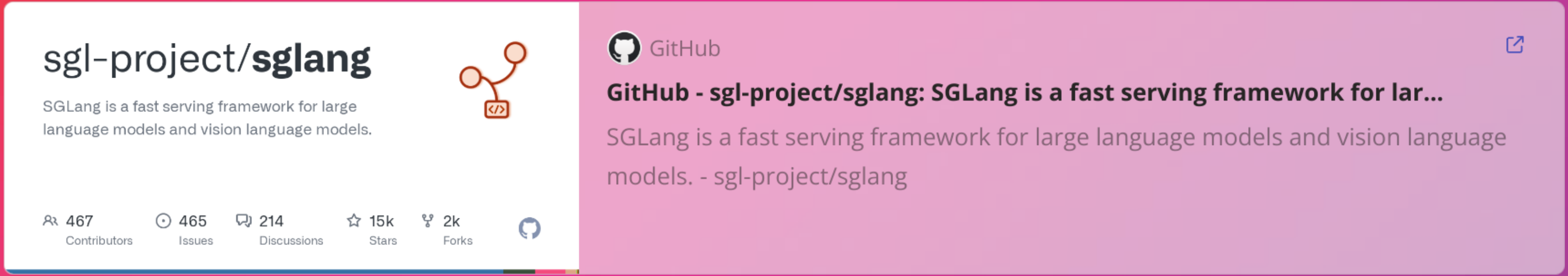
PyTorch DAY

CHINA 2025

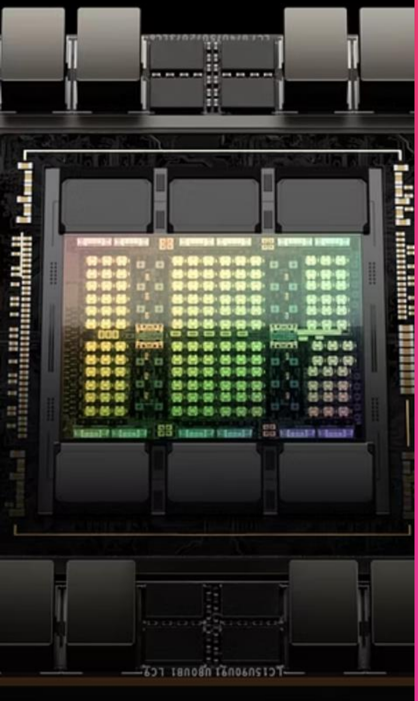
Co-hosted by

BAAI 智源研究院 PyTorch

What is SGLang



- **SGLang is a fast-serving engine for LLMs and VLMs.**
- Among fully open-source LLM Inference Engines, SGLang currently achieves **state-of-the-art (SOTA) performance, and It** is **the first open-source implementation to nearly match the throughput reported in the official DeepSeek blog** at a large scale.
- Meanwhile, its **elegant, lightweight, and customizable design** has attracted wide adoption from academics, big tech companies, and startups. **(XAI, NVIDIA, AMD, Baseten, Microsoft, Linkedin, etc.)**
- In on-policy RLHF, inference engines are crucial for efficient policy model execution, and **SGLang excels as a high-performance solution.**



Outlines

1. SGLang Milestones and Features Overview
2. Efficient Design and Implementation of PD Disaggregation
3. Large-scale EP Support and DeepSeek Blog Reproduction
4. The Ecosystem of the SGLang Community and Future Development

SGLang Milestones and Features Overview

SGLang Milestones and Features

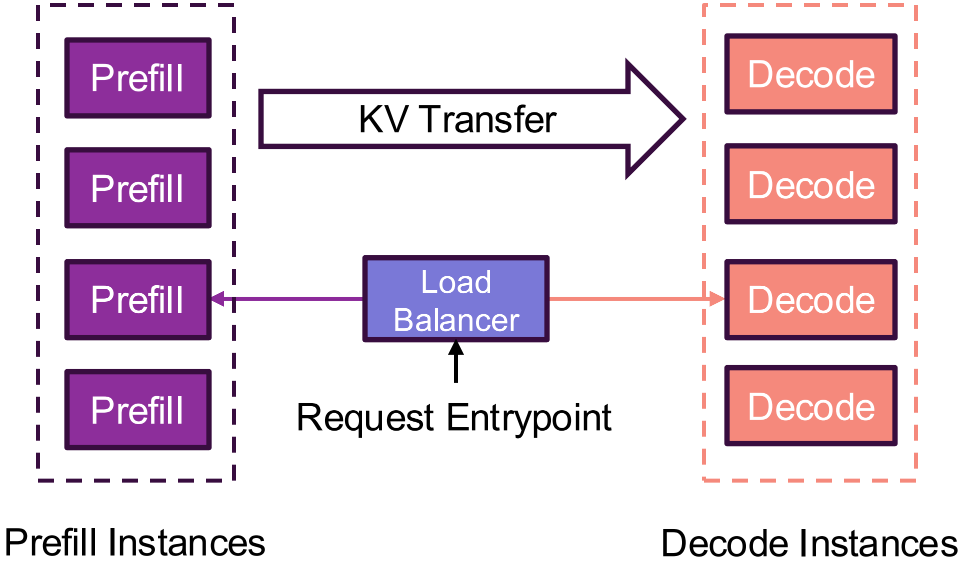
- 2023/12-2024/2: Initial Motivation, **Structured LM Programming, Prefix Caching,** and **Constrained Decoding**
- 2024/07: **Leading Performance** among inference engines on Llama3
- 2024/09: v0.3 Release, 7x Faster **DeepSeek MLA**, 1.5x Faster **torch.compile**, Multi-Image/Video LLaVA-OneVision
- 2024/12: v0.4 Release: **Zero-Overhead Batch Scheduler**, **Cache-Aware DP Router**, **X-Grammar Integration**, **The First to Serve DeepSeek V3**.
- 2025/01: SGLang provides day-one support for DeepSeek V3/R1 models on NVIDIA and AMD GPUs with DeepSeek-specific optimizations. (10+ companies!)
- 2025/05 🔥🔥🔥 First open-source implementation of DeepSeek V3/R1 **expert parallelism** with **prefill-decode disaggregation**. Achieves 52.3K in-tok/s, 22.3K out-tok/s on 96 GPUs—5× faster than vanilla TP.
- SGLang has seen extensive adoption and serves as the dominant inference engine for AMD and the default inference engine for xAI.

Efficient Design and Implementation of PD Disaggregation

Issues with Non-Disaggregation

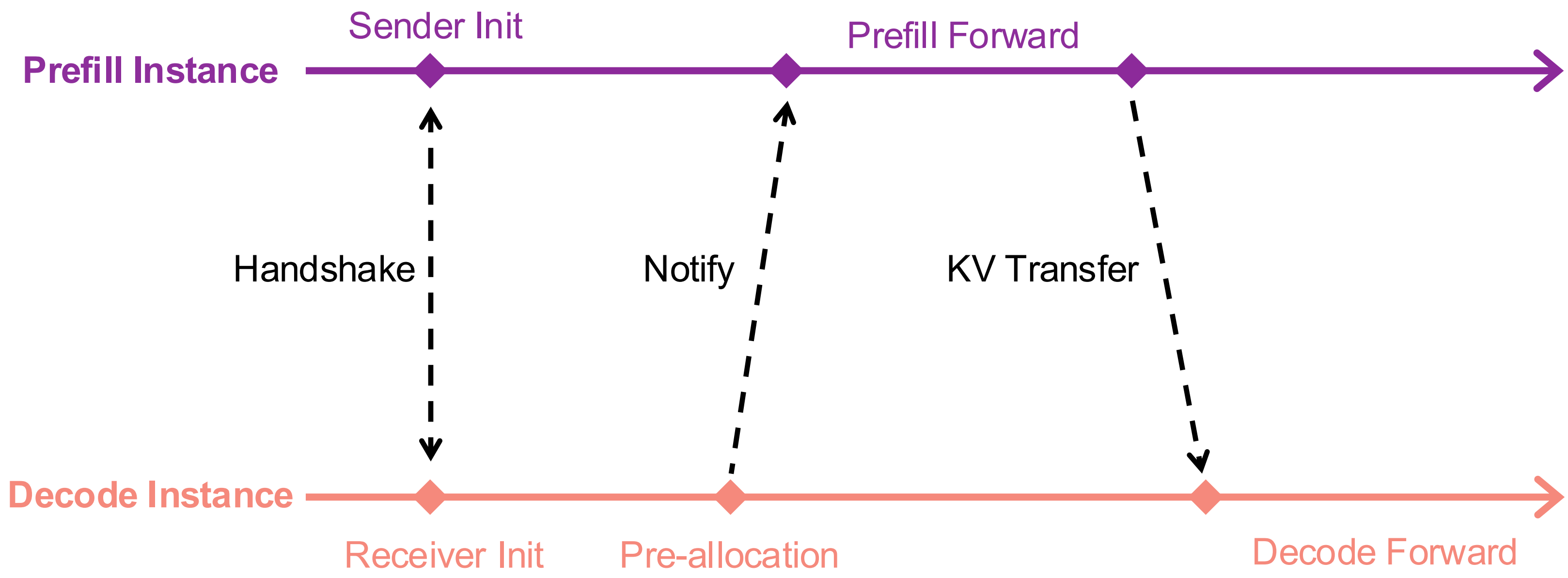
- **Prefill Interruption**: Prefill batches often preempt ongoing decode tasks, delaying token generation.
- **DP Attention Imbalance**: DP workers may handle prefill and decode simultaneously, causing load imbalance and increased latency.
- **Incompatible with DeepEP**: Prefill and decode use different dispatch modes. Without disaggregation, DeepEP cannot support both within the same communication group under DP attention.

PD Disaggregation Architecture Design



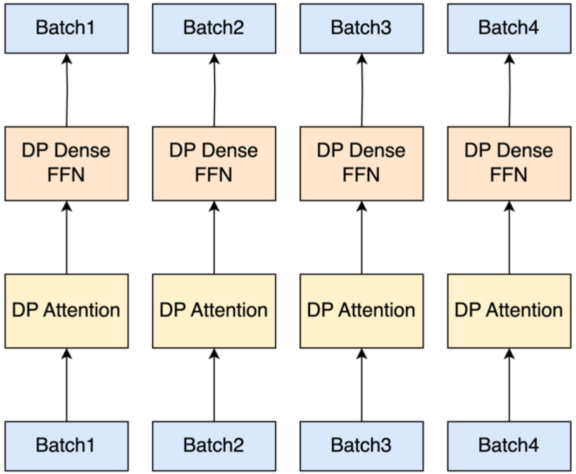
- Unified Load Balancer for both prefill and decode paths.
- LB is **decoupled** from computation logic: requests are sent to LB and then routed to a selected PD pair.
- KV transfer supports **non-blocking** and **RDMA-based** transfer.
- SGLang offers **flexible API integration** like **NIXL** and **Mooncake**.

PD Disaggregation Timestamp



Large-Scale Expert-Parallelism Support and DeepSeek Blog Reproduction

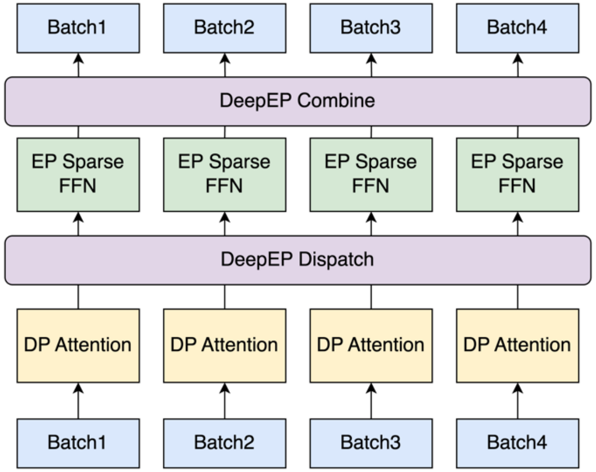
Parallelism Strategies with Dense FFN



(a) DP Dense FFN with DP Attention

- **Enhanced Scalability**: Avoids TP fragmentation on large hidden dims (e.g., 18432), ensuring better alignment and utilization.
- **Optimized Memory Efficiency**: Prefill & decode phases both benefit from low TP degrees under DP attention, reducing per-device memory.
- **Minimized Communication Overhead**: Replaces two all-reduces (in TP) with one reduce-scatter + one all-gather.

Parallelism Strategies with Sparse FFN (MoE)



(b) EP Sparse FFN with DP Attention

- **Scalable Model Capacity:** Expert weights are partitioned across devices using Expert Parallelism, removing memory bottlenecks.
- **Optimized Communication:** Follows a Dispatch → Expert → Combine pattern; powered by DeepEP and Two-Batch-Overlap to minimize latency and overhead.
- **Addressing Load Imbalance:** EP introduces variability in routing; EPLB and DeepEP optimize for workload distribution.

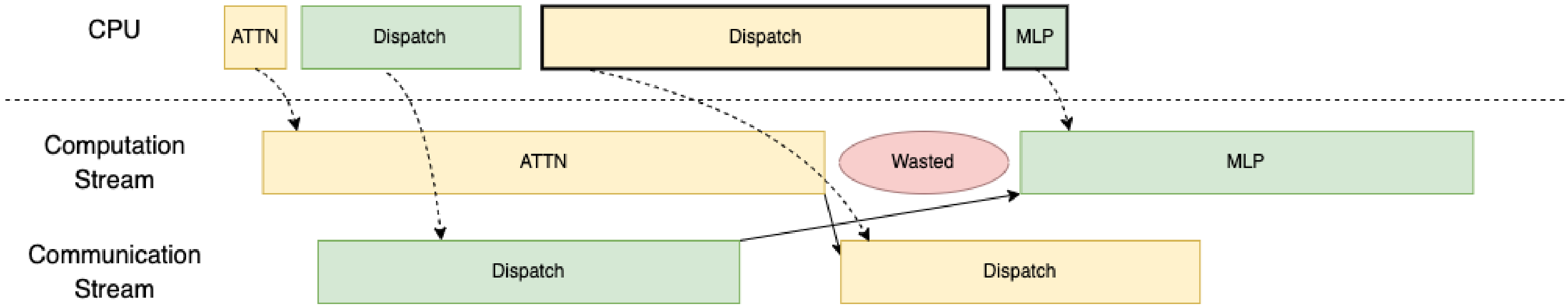
Compatibility Issue with DeepEP

Mode	Long Input	Long Output	DP Attention	CUDA Graph
Normal	✅	❌	✅	❌
Low-Latency	❌	✅	✅	✅
Auto	✅	✅	❌	✅

DeepEP holds two different dispatch mode

- **Normal**: Prefill-friendly, but no CUDA Graph.
- **Low-Latency**: Decode-friendly, supports CUDA Graph.
- **Auto**: Handles both input/output, but is incompatible with DP Attention with unified scheduling (non-disaggregation).
- PD Disaggregation resolves DeepEP Dispatch & DP Attention incompatibility.

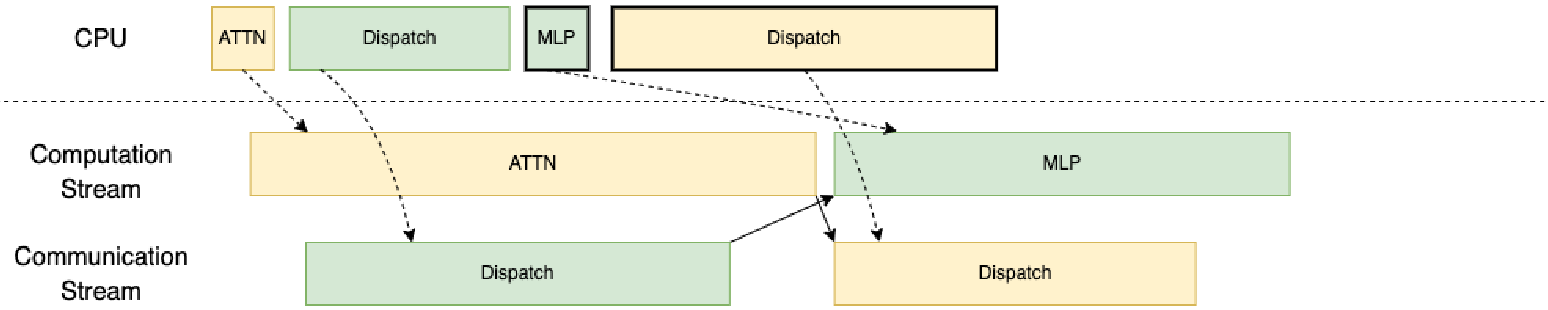
Improper Launch Order of TBO



(a) Two-batch overlap with an improper launch order

- TBO: Communication and computation are expected to be executed simultaneously.
- Dispatch brings **synchronization**, which blocks the CPU until the GPU receives metadata (required for allocating correctly sized tensors).
- **Improper launch order**, e.g. dispatch before MLP, will block the launching and leave the computation stream idle.

Proper Launch Order of TBO



(b) Two-batch overlap with a proper launch order

- Proper launch order: submitting computation tasks to the GPU **before launching CPU-blocking communication**.
- Computation → Communication: enabling GPU to remain active during communication.

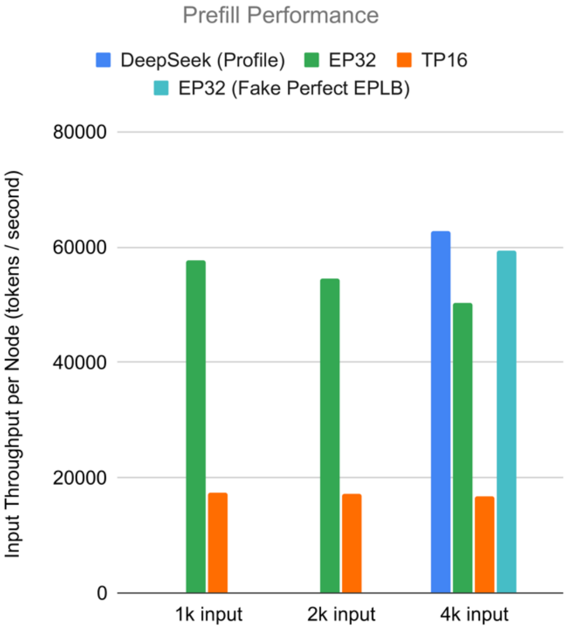
Clean Implementation: Two-Batch-Overlap

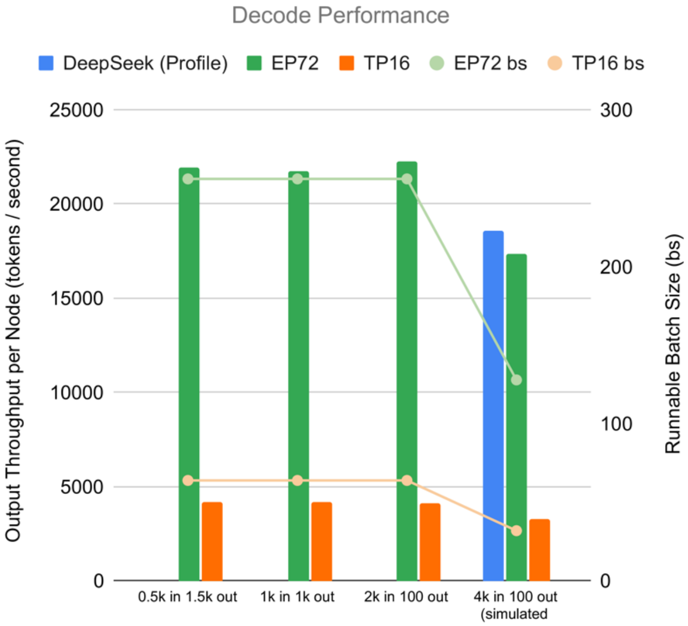
```
operations = [
    self._forward_attn,
    YieldOperation(),  # Pause execution for other micro-batches
    self._forward_dispatch,
    self._forward_mlp,
    YieldOperation(),  # Another pause point
    self._forward_combine,
]

# Process a single micro-batch without duplicating code
def _forward_attn(self, state):
    state.hidden_states = self.self_attn(state.hidden_states, ...)
```

- Abstracted execution via **operation list** + **yield points**, enabling cooperative scheduling.
- Eliminates code duplication and reduces the need for variable post-fixes.
- Efficiently manages partial completion at layer boundaries.

Throughput Performance





Throughputs of prefill (P) and decode (D) phases are evaluated independently, assuming unlimited resources for the non-tested phase to isolate and maximize the load on the tested nodes—mirroring the setup used by DeepSeek.

Expert Parallelism Load Balancer

Real-World Serving Challenges

- **Imbalance worsens at scale** with expert usage skews causing idle GPU time.

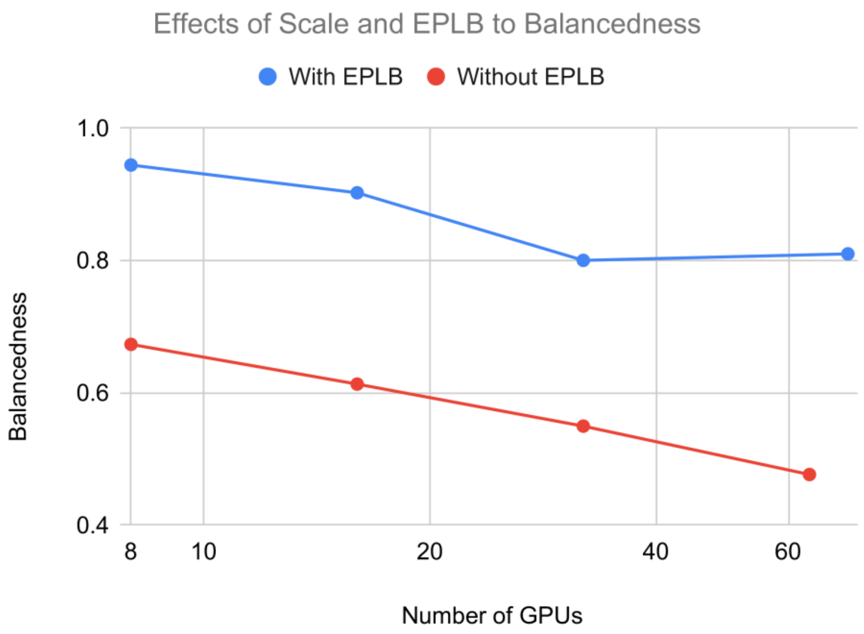
Strategies to Improve Balance

- **Larger Batch Sizes**: Reduces randomness in expert routing; enabled via cluster scaling or Multi-Token Prediction (MTP).
- **Periodic Rebalancing**: Adapts to input shifts over time; requires low-cost expert weight reloads.

SGLang Implementation

- Exchange expert weights with torch P2P operations.

Effects of Scale and EPLB to Balancedness



- **Balancedness:** the ratio between mean computation time and maximum computation time for a MoE layer among GPUs.
- Balancedness decreases when the system scales with the number of nodes.
- Enabling EPLB significantly improves the balancedness.

The Ecosystem of the SGLang Community and Future Development

Future Work of Large-Scale Serving

- **Latency Optimization:** TTFT remains at 2–5s; ITL around 100ms — needs tuning for real-time use cases.
- **Sequence Length Constraints**: Limited by the current 96-GPU setup; longer sequences require more hardware.
- **MTP & DP Integration**: Multi-Token Prediction is not fully integrated with DP attention, reducing efficiency.
- **EPLB Generalization**: Current tests use in-distribution data; future work should include distribution shift scenarios.
- **Flexible TP Sizes**: DeepSeek-V3 benefits from small but strong TP — SGLang currently supports only full TP or DP.
- **Blackwell GPU Support**: Presently supports NVIDIA Hopper only; Blackwell compatibility is in development.

About SGLang Team

- SGLang Team is incubated by LMSYS Org
- Major Maintainers: Lianmin Zheng, Ying Sheng, Liangsheng Yin, Yineng Zhang, Ke Bao, Byron Hsu, Chenyang Zhao, Zhiqiang Xie, Jingyi Chen, Xiaoyu Zhang, Baizhou Zhang, Yi Zhang, Jiexin Liang, Chang Su, Hai Xiao.
- Contributors: 400+

Community Adoptions

xAI NVIDIA AMD Google Cloud ORACLE

LinkedIn CURSOR VOLTAGE PARK Atlas Cloud Data Crunch

baseten NEBIUS Novita innomatrix RunPod

Stanford Berkeley UNIVERSITY OF CALIFORNIA Ucla ETCHED Jam & Tea STUDIOS Hyperbolic

Alibaba Cloud 美团 Meituan ANT GROUP 01.AI Tencent Cloud