Slime

面向 Agentic RL 的大规模 MoE 训练框架

SGLang RL / Ant AQ 团队

github.com/THUDM/slime

OVERVIEW

内容概览

PART 1

框架概览与设计哲学

SGLang-native 架构 · Server-based Rollout

PART 2

面向 Agentic RL 的设计

自定义 Rollout · 多轮交互 · Agent Framework

PART 3

从"能训"到"能训好"

Mismatch 问题 · IS · True On-Policy · R3 · 低精度

PART 4

加速与未来规划

Speculative Decoding · TODO · Roadmap

01

框架概览与设计哲学

An SGLang-Native Post-Training Framework for RL Scaling

概览

Slime 核心设计目标

Versatile

完全可定制的 Rollout 接口
Colocated / Decoupled 部署
同步 / 异步训练

Performant

SGLang 原生推理集成
Megatron-LM 原生训练集成
全并行策略支持

Maintainable

轻量级代码库
参数无缝透传
预训练→部署平滑过渡

维度	能力
模型支持	Qwen3 · DeepSeek V3/R1 · GLM-4/4.5 · Kimi K2 · Llama 3
RL 算法	GRPO · PPO · GSPO · Reinforce++ · On-policy Distillation
并行策略	训练: TP/SP/PP/CP/EP/ETP | 推理: TP/DP Attention/EP+DeepEP
精度	BF16 + FP8 Rollout · FP8 + FP8 · INT4 QAT

设计哲学

Server-based Rollout

> *"A prevailing misconception within the RL community is the need for separate frameworks for different tasks... no one forks PyTorch just for a new dataloader."*

	Engine-Based Rollout	Server-Based Rollout (Slime)
设计方式	框架内部直接调用推理引擎	启动 SGLang Server，HTTP API 交互
Agent 兼容性	与 Agent Framework 方式相悖	天生兼容 OpenAI-compatible API
用户自由度	受限于预定义模板	注入自定义逻辑，自由交互
性能调优	框架耦合，难以独立优化	`--debug-rollout-only` 独立调优

02

面向 Agentic RL 的框架设计

从 Reasoning Task 到 Agentic Task 的范式转变

范式

RL 范式的转变

过去

Reasoning Task

单轮 Rollout

→

现在

Agentic Task

多轮 Rollout

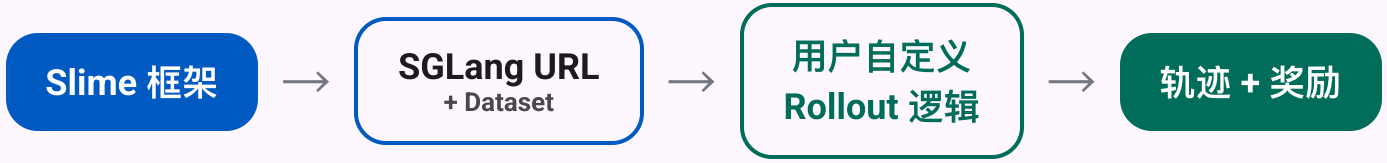
- 模型需与外部环境（工具、API、代码执行器）多轮交互
- 逐渐引入 Agent Framework / Agent Infra
- **核心问题：什么样的框架设计可以丝滑地支持各种任务?**

Slime 的回答：不预设模板，而是提供可定制的接口，让用户注入自己的逻辑

接口

自定义 Rollout 接口

不同 Agent 任务的生成逻辑和环境可能差异巨大，Slime 将自由度交给用户：



19+ 可定制接口覆盖全流程：

接口	用途
`--rollout-function-path`	替换整个 rollout 生成逻辑
`--custom-generate-function-path`	仅替换生成步骤（RAG/tool-use）
`--custom-rm-path`	自定义奖励计算
`--custom-tis-function-path`	自定义 Importance Sampling
`--custom-loss-function-path`	自定义训练损失函数

```python
async def generate_rollout(
    args,
    rollout_id,
    *,
    evaluation=False
) -> RolloutFnTrainOutput:
    # 用户返回的 Sample 需设置:
    # - tokens
    # - response_length
    # - reward
    # - truncated
    # - loss_mask (多轮任务)
    ...
```

多轮

单轮 → 多轮 Rollout & Multi Agent Training

单轮/多轮的生成本质上没有任何区别！

- 框架本身不区分单轮与多轮——区别仅在 rollout function 内部逻辑
- 用户将多轮交互的 token 序列拼接，正确设置 `loss_mask` 即可
- 对于多轮/Agentic 场景，推荐使用 **PD Disaggregation**（Prefill-Decode 分离），优化频繁 Prefill 的性能

Multi Agent Training

蚂蚁集团 AQ Team — MrlX

- 多个 Agent 各自与 SGLang Server 交互生成轨迹
- 协调训练中医生模型 reward 显著增加
- 患者 reward 在训练过程中同步上升

Agent Framework RL

- 现有 Agent 框架基于 OpenAI Base URL，与 Slime **天然吻合**
- 已有 **strands-agents** 集成示例，可迁移至任何框架

03

从"能训"到"能训好"

解决 Training-Inference Mismatch 问题

问题

Mismatch 问题全景

Agentic RL 引入外部 observation（工具返回值、环境反馈），使 mismatch 影响更显著，训练不稳定甚至崩溃。

Retokenization Mismatch

多轮交互中文本重编码导致 token 序列不一致

南京 | 市长 | 江大桥 ≠ 南京市 | 长江大桥

Training-Inference Mismatch

SGLang 输出的概率分布 ≠ Megatron/FSDP 输出的概率分布

三大产生原因：

原因	严重程度
算子不确定性 batch size / tile size 不同	中
数据类型差异 BF16 训练 + FP8 推理	高
MoE 专家差异 训推激活不同专家	极高

Slime 的解决方案矩阵：

1. Importance Sampling
2. True On-Policy
3. 低精度统一
4. Routing Replay

方案

Retokenization Mismatch → SlimeRouter Radix-tree Cache

SlimeRouter 透明拦截文本请求，自动缓存 token IDs，消除重编码 mismatch。

1. 拦截 text-based 请求，tokenize 并将轨迹（text, token IDs, logprobs, loss masks）以文本前缀为 key 存入 radix tree
2. **最长前缀匹配**复用已缓存 token 序列（支持 GRPO 多轨迹共享前缀）
3. Rollout 结束后 `/retrieve_from_text` 返回精确 token 序列 + 对齐 metadata
4. 定期清理过期节点，控制内存使用

SlimeRouter vs SGLang Gateway

	SlimeRouter	SGLang GW
实现	Python/FastAPI	Rust
设计	Passthrough	固定 schema
R3 支持	✅	❌
Radix Cache	✅	❌
高性能路由	–	✅

方案 1

Importance Sampling (IS) — 算法层面缓解

从算法层面减少 mismatch 对训练稳定性的影响。

框架需要支持：

- **不同 IS 计算粒度**：Token level · Sequence level · Geometry level
- **不同处理策略**：限制权重大小 · 裁剪梯度

Slime 的设计：

- 提供 training logp、inference logp 等数据
- 用户通过 `--custom-tis-function-path` 注入自定义 IS 算法
- 内置 `--use-tis` 提供 Truncated IS 开箱即用

```python
# 自定义 IS 示例
# examples/train_infer_mismatch_helper/mis.py

def compute_mis_weights_with_cp(
    train_logp,      # 训练 log prob
    inference_logp,  # 推理 log prob
    loss_mask,
    ...
):
    # 计算 importance weights
    # 返回加权后的损失
    return weighted_loss
```

还可通过 `--custom-pg-loss-reducer-function-path` 自定义 pg_loss 归约方式（如 Dr.GRPO）。

方案 2

True On-Policy — 算子级对齐，前向/反向算子一致

IS 是算法层面的缓解，能否从**系统层面**彻底对齐 training engine 和 inference engine?

✅ Dense Model

SGLang Deterministic Rollout + FSDP Backend
→ 算子级前向/反向一致，**bitwise reproducibility**

✅ VLM

True On-Policy with Qwen3-VL on FSDP
→ 训推对齐扩展至视觉语言模型

实现配置：

```
# SGLang 确定性推理
--sglang-enable-deterministic-inference
--sglang-attention-backend flashinfer

# Megatron 确定性模式
--deterministic-mode

# 环境变量
NCCL_ALGO=Ring
NVTE_ALLOW_NONDETERMINISTIC_ALGO=0
CUBLAS_WORKSPACE_CONFIG=:4096:8
```

需卸载 flash_attn_3。验证方式：多次运行检查 log prob 完全一致。

方案 3

低精度统一 — FP8 / INT4 QAT 全流程

训推使用相同低精度，从精度统一角度消除 mismatch。MoE 规模越大，BF16+FP8 差异越明显。

模式 1: FP8 Rollout + BF16 Train

`convert_hf_to_fp8.py`

Block size 128×128

设置 `--hf-checkpoint` 即可

模式 2: FP8 Rollout + FP8 Train

`--fp8-format e4m3 --fp8-recipe blockwise`

TransformerEngine FP8 GEMM

embedding/lm_head 保持原精度

模式 3: INT4 QAT

STE fake quantization

group_size: Qwen3=128, K2=32

INT4 推理大幅提升 throughput

模型	INT4 QAT 脚本	规模
Moonlight-16B-A3B	`run-moonlight-16B-A3B-int4.sh`	单机
Qwen3-30B-A3B	`run-qwen3-30B-A3B-int4.sh`	多机
Qwen3-235B-A22B	`run-qwen3-235B-A22B-int4.sh`	8 nodes
Kimi-K2-Thinking	`run-kimi-k2-Thinking-int4.sh`	32 nodes

方案 4

Rollout Routing Replay (R3) — MoE 专属方案

确保训练时使用与推理时完全相同的专家路由决策。**DeepSeek 3.2 的训练使用了该技术。**

SGLang 侧实现：

- `--enable-return-routed-experts`
- `RoutedExpertsCapturer` 捕获 topk_ids
- 返回张量 `[seq_len-1, num_layers, top_k]`

两种 Routing Replay 模式：

参数	说明
`--use-routing-replay`	前向-反向一致（不依赖 rollout 数据）
`--use-rollout-routing-replay`	R3: 使用 rollout 路由数据（需 SlimeRouter）

04

加速与未来规划

Speculative Decoding in RL · Roadmap

加速

Speculative Decoding in RL

Draft model 快速生成候选 token，target model 批量验证。**关键创新：Online SFT for Draft Model**。

- RL 训练中 draft/target 分布漂移 → 验证通过率下降
- Slime 支持训练中**同步更新 MTP layers**
- Accepted length 持续高水平，长期收益稳定

~35%
训练 Spec vs 不开启 Spec

~14%
训练 Spec vs 冻结 Spec

25%
训练后期提升达到

配置参数：

```
# 推理加速 (EAGLE)
--sglang-speculative-algorithm EAGLE
--sglang-speculative-num-steps 3
--sglang-speculative-eagle-topk 1
--sglang-speculative-num-draft-tokens 4

# Online MTP 训练
--mtp-num-layers 1
--enable-mtp-training
--mtp-loss-scaling-factor 0.2
```

适用于 GLM-4.7、DeepSeek-V3/R1 等有 MTP layers 的模型。也支持 SpecForge 独立 draft model。

未来

TODO & 未来规划

Agent 侧

- 多任务 RL 训练支持
- 更方便的 config 传递
- 平衡不同 task 速度
- 更好的监控指标
- VLA RL

Infra 侧

- Slime Router for OpenAI Endpoint
- True On-Policy（MoE Model）

官方路线图关键词：

大规模 MoE 最优 RL 策略 · 更广泛的后训练工作流 · PyTorch 原生训练后端降低门槛

Thanks!