

PR #2843 完整报告

radixark/miles

[multi-lora v2 4/7] Publish immutable adapter snapshots to disk

合并时间: 2026-09-16 16:25

原文链接: <http://prhub.com.cn/radixark/miles/pull/2843>

执行摘要

- 一句话: 新增 LoRA 适配器不可变磁盘快照发布路径
- 推荐动作: 值得精读。三个最值得关注的设计点: 一是 joyliu-q 发现的生成器惰性求值 bug 及其修复方式 (把循环变量改为函数形参绑定, 而非依赖 `itertools.chain` 的求值时机), 这是 Python 里极易复现的陷阱; 二是 `materialize_adapter` 的集合通信契约——所有 rank 必须走同一路径、只有 writer 产出数据; 三是 `staged-push` 被整体删除后, 失败与重试语义转移到了哪里 (#2845 的 runtime 与 SGLang on-demand loading), 以及删除 `slot_lora_name` 后全仓引用是否已清理。如果所在团队还在用旧的多 LoRA push 行为, 需要先确认配置不再显式设置 `update_weights/model_path`。

功能与动机

PR body 给出的目标是「Publish Tinker sampler adapters through `WeightPublisher`」。「Every trainer rank joins the shared HF iterator's gathers; rank 0 writes the selected slot's rank-trimmed tensors, adapter configuration, and metadata into an immutable directory」, 并强调「an existing version cannot be overwritten」。旧路径要求 trainer 主动把 adapter push 给 SGLang, 依赖 update window 与 engine session, 检查点 IO 失败还会被折成结果字典静默继续。Tinker 网关需要推理引擎加载与训练一致的冻结 base, 且在没有 trainer push 的情况下也能被 server-cell 生命周期标记为 ready, 因此本 PR 把「发布」重写成一次磁盘事务, 并为引擎侧的按需加载提供不可变目录。

实现拆解

1. 新增发布者抽象

- 新增 `miles/backends/training_utils/weight_update/snapshot_publisher.py`:
`WeightPublisher.__init__` 断言 `iterator.placement.is_full_gather` (rank 0 必须持有完整 adapter), `publish_adapter(AdapterSpec, path, metadata)` 内定义 `write_shards(tmp_dir)`, 由 rank 0 把 `materialize_adapter` 的结果搬到 CPU、经 `safetensors.torch.save` 序列化, 连同 `adapter_config.json` (`self._adapter_config | {"r": ..., "lora_alpha": ...}`) 写入临时目录, 最终交给 `write_checkpoint_dir(path, write_shards, metadata=..., overwrite=False)` 原子发布。
- 关键点: 所有 rank 都必须进入 `materialize_adapter` (内部是集合通信), 只有 rank 0 落盘; `overwrite=False` 保证已存在版本不可覆盖。

2. 改造 HF 权重迭代器契约

- miles/backends/training_utils/weight_update/hf_weight_iterator/__init__.py: WeightUpdatePlacement 新增 is_full_gather property; iter_hf_weights 里的前缀拼接抽成局部函数 prefixed_units(lora_name, adapter); 新增 materialize_adapter(adapter, materialize=...), 返回裸 HF 名字的 {hf_key: tensor}, 绕开 pack_units_by_size 的传输分桶——这正是 safetensors 需要的键形态。
- 抽象钩子 _iter_hf_adapter_units 去掉 lora_name 形参、改为产出裸 hf_name, 前缀职责上移到基类; Megatron 实现 miles/backends/megatron_utils/update_weight/hf_weight_iterator.py 同步改签名。这是对后续 rollout 侧 (#2844/#2845) 的接口收口。

3. 训练侧接入与导出边界

- miles/backends/megatron_utils/lora/actor.py: _init_training_state() 里用 get_hf_weight_iterator(args, self.model, required_placement=WeightUpdatePlacement(gather_pp=True), model_name=..., quantization_config=...) 加 build_lora_sync_config(args) 构造 self.weight_publisher, 与传输层完全解耦 (无 update window、无 engine session) 。
- 新增 export_slot(slot, rank, alpha, path, metadata), 带 @with_logs 与 self._heartbeat.bump(), 直接把异常抛给 TrainerCell, 不再转成结果字典; miles/ray/train/group.py 增加同名 async RPC, 通过 _execute_slots 广播到所有 rank。

4. 引擎侧与配置约束

- miles/backends/sglang_utils/sglang_config.py: ModelConfig.resolve 在 is_multi_lora_enabled(args) 时断言 update_weights is not True 与 effective_model_path == args.hf_checkpoint, 随后强制 update_weights = False; 原来的 update_weights is None 分支保留给普通路径。
- miles/backends/sglang_utils/sglang_engine.py: 多 LoRA 时拒绝 load_format == "dummy"; max_loaded_loras 未显式设置时默认 2 * max_loras_per_batch (含 TODO: 动态分配), 显式 CLI/group 覆盖仍保留。

5. 清理与配套

- 删除 Tinker staged-push 路径: weight_update/updater.py 恢复为纯 base 同步 (连带单 LoRA adapter); 相关 engine-session 错误类、update-window 改动、staged-publication 专用测试一并移除, miles/utils/multi_lora.py 的 __all__ 删掉 slot_lora_name 并重写若干断言文案。
- 新增公共写入助手 save_adapter_to_disk(out_dir, config, tensors) (miles/utils/lora.py, safetensors 延迟 import, 避免不碰权重的路径背依赖) 。
- 测试: tests/fast/backends/sglang_utils/test_sglang_config.py 对 multi_lora=True/False 参数化并断言 update_weights is not multi_lora; tests/fast/backends/training_utils/weight_update/test_hf_weight_iterator.py 新增 test_each_adapter_keeps_its_own_prefix 锁住前缀回归。文档 docs/advanced/lora.md 同步更新冻结 base 与不可变目录说明。

- PR body 提示：发布器的参数化回归（进程在链接替换前后退出、随后重试）落在 #2849 的 checkpoint 测试文件里，不在本 PR 内。

关键文件：

- miles/backends/training_utils/weight_update/snapshot_publisher.py（模块 权重发布；类别 source；类型 core-logic；符号 WeightPublisher, WeightPublisher.init, WeightPublisher.publish_adapter, write_shards）：本 PR 的核心新增文件：定义 WeightPublisher，把「一个 slot 的 adapter 落成引擎可加载的不可变目录」这一件事封装成单次原子发布事务，是整条磁盘快照路线的基础。
- miles/backends/training_utils/weight_update/hf_weight_iterator/__init__.py（模块 权重迭代；类别 source；类型 core-logic；符号 WeightUpdatePlacement.is_full_gather, HfWeightIteratorBase.iter_hf_weights, HfWeightIteratorBase.prefixed_units, HfWeightIteratorBase.materialize_adapter）：迭代器契约改造的落点：新增 is_full_gather 与 materialize_adapter，前缀拼接上移到 prefixed_units，_iter_hf_adapter_units 去掉 lora_name；同时是 joyliu-q 报告的多 adapter 前缀串扰 bug 所在文件与修复位置。
- miles/backends/megatron_utils/lora/actor.py（模块 槽位训练；类别 source；类型 core-logic；符号 MultiLoRATrainRayActor._init_training_state, MultiLoRATrainRayActor.export_slot）：多 LoRA 训练 actor 的接线点：_init_training_state 里独立于传输层构造 WeightPublisher，新增 export_slot 作为唯一的导出边界并让异常穿透 TrainerCell。
- miles/backends/sglang_utils/sglang_config.py（模块 引擎配置；类别 source；类型 dependency-wiring；符号 ModelConfig.resolve）：把多 LoRA 引擎锁定为「加载冻结 base、不接受 trainer 推送」：ModelConfig.resolve 在 is_multi_lora_enabled 时强制 update_weights=False 并断言 model_path 与 hf_checkpoint 一致。
- miles/backends/sglang_utils/sglang_engine.py（模块 推理引擎；类别 source；类型 configuration；符号 _compute_server_args）：引擎启动参数侧的配套约束：多 LoRA 时拒绝 load_format=dummy，并给出 max_loaded_loras 的默认启发式，直接影响按需加载容量与显存。
- miles/utils/multi_lora.py（模块 适配器配置；类别 source；类型 core-logic；符号 slot_lora_name, AdapterSpec, validate_multi_lora_args）：删除了 __all__ 中的 slot_lora_name（引擎侧命名职责外移），并重写若干 multi-LoRA 启动断言的文案，是接口边界收敛与配置约束变化的集中体现。
- miles/backends/megatron_utils/update_weight/hf_weight_iterator.py（模块 权重导出；类别 source；类型 data-contract；符号 MegatronHfWeightIterator._iter_hf_adapter_units）：Megatron 后端的 _iter_hf_adapter_units 实现随基类契约一起改签名：去掉 lora_name、产出裸 hf_name，是前缀上移能够成立的前提。
- miles/ray/train/group.py（模块 训练编排；类别 source；类型 entrypoint；符号 TrainGroup.export_slot）：把导出能力暴露成训练组级别的 RPC：新增 export_slot，通过 _execute_slots 广播到所有 rank，是 runtime 侧调用发布器的入口。
- miles/backends/training_utils/weight_update/updater.py（模块 权重同步；类别 source；类型 refactor；符号 WeightUpdater.update_weights）：删掉 Tinker staged-push 对

共享 updater 的侵入，恢复为「只做 base 同步（外加单 LoRA adapter）」的干净路径，是 nanjiangwill 多条 review 意见的直接回应。

- miles/utils/lora.py（模块 适配器落盘；类别 source；类型 core-logic；符号 save_adapter_to_disk）：新增通用适配器写盘助手 save_adapter_to_disk，供其他路径（如测试 fake 与后续 runtime）复用，避免磁盘格式写多份。
- tests/fast/backends/training_utils/weight_update/test_hf_weight_iterator.py（模块 迭代器测试；类别 test；类型 test-coverage；符号 test_each_adapter_keeps_its_own_prefix）：新增 test_each_adapter_keeps_its_own_prefix，把 review 中发现的惰性求值前缀串扰固化为回归测试，同时同步 _iter_hf_adapter_units 的桩实现。
- tests/fast/backends/sglang_utils/test_sglang_config.py（模块 配置测试；类别 test；类型 test-coverage；符号 test_prefill_num_servers_counts_engines_not_gpus, test_no_offload_flag_means_no_group_needs_offload）：对 multi_lora 参数化并在两个既有用例里断言 update_weights is not multi_lora，锁住引擎配置的冻结 base 语义。
- docs/advanced/lora.md（模块 适配器文档；类别 docs；类型 documentation）：同步说明冻结 base、不可变 sampler 目录与按需加载，帮助用户理解配置侧行为变化。

关键符号：WeightPublisher.init, WeightPublisher.publish_adapter, HfWeightIteratorBase.materialize_adapter, HfWeightIteratorBase.iter_hf_weights, HfWeightIteratorBase._iter_hf_adapter_units, MultiLoRATrainRayActor._init_training_state, MultiLoRATrainRayActor.export_slot, TrainGroup.export_slot, save_adapter_to_disk, ModelConfig.resolve

关键源码片段

miles/backends/training_utils/weight_update/snapshot_publisher.py

本 PR 的核心新增文件：定义 WeightPublisher，把「一个 slot 的 adapter 落成引擎可加载的不可变目录」这一件事封装成单次原子发布事务，是整条磁盘快照路线的基础。

```
# 新增文件：miles/backends/training_utils/weight_update/snapshot_publisher.py
import json
from pathlib import Path

import safetensors.torch
import torch
import torch.distributed as dist

from miles.backends.training_utils.checkpoint_io import write_checkpoint_dir
from miles.backends.training_utils.weight_update.hf_weight_iterator import
HfWeightIteratorBase
from miles.utils.multi_lora import AdapterSpec

class WeightPublisher:
    def __init__(self, iterator: HfWeightIteratorBase, adapter_config: dict) -> None:
        # 发布前提：rank 0 必须持有完整 adapter。PP/TP/EP 任一维没有 gather 时，
        # rank 0 手里只有分片，写出的 safetensors 会缺权重，所以在构造期就 fail fast。
```

```

assert iterator.placement.is_full_gather, "publishing requires the full adapter on rank 0"
self._iterator = iterator
self._adapter_config = adapter_config

```

```
@torch.no_grad()
```

```
def publish_adapter(self, adapter: AdapterSpec, path: str, metadata: dict | None = None) ->
None:
```

```

# 所有 rank 都必须进入 materialize_adapter, 因为里面的 gather 是集合通信;
# 只有 rank 0 (is_writer) 把 tensor 搬到 CPU 并落盘, 其余 rank 走 materialize=False
# 分支: 仍然参与通信, 但返回空字典。
is_writer = dist.get_rank() == 0

```

```
def write_shards(tmp_dir: Path):
```

```

    tensors = {
        name: tensor.detach().contiguous().cpu()
        for name, tensor in self._iterator.materialize_adapter(adapter, materialize=is_writer).
        items()
    }

```

```
    data = safetensors.torch.save(tensors) if is_writer else None
```

```
if is_writer:
```

```

    # r / lora_alpha 直接取自 AdapterSpec, 保证落盘的缩放系数与训练侧 slot 配置一致,
    # 而不是依赖训练启动参数里可能被改写过的默认值。
    config = self._adapter_config | {"r": adapter.rank, "lora_alpha": adapter.alpha}
    (tmp_dir / "adapter_config.json").write_text(json.dumps(config))
    (tmp_dir / "adapter_model.safetensors").write_bytes(data)

```

```

# overwrite=False 配合临时目录 + 原子改名: 目录要么是完整的新版本, 要么保持旧版本,
# 引擎侧永远读不到写了一半的 adapter, 且已存在的版本号无法被覆盖。
write_checkpoint_dir(path, write_shards, metadata=metadata, overwrite=False)

```

miles/backends/training_utils/weight_update/hf_weight_iterator/__init__.py

迭代器契约改造的落点: 新增 is_full_gather 与 materialize_adapter, 前缀拼接上移到 prefixed_units, _iter_hf_adapter_units 去掉 lora_name; 同时是 joyliu-q 报告的多 adapter 前缀串扰 bug 所在文件与修复位置。

```
# miles/backends/training_utils/weight_update/hf_weight_iterator/__init__.py (改造后)
```

```
@dataclasses.dataclass(frozen=True)
```

```
class WeightUpdatePlacement:
```

```

    gather_pp: bool
    gather_tp: bool = True
    gather_ep: bool = True

```

```
@property
```

```
def is_full_gather(self) -> bool:
```

```

    # WeightPublisher 用它判断「rank 0 手上的 adapter 是否完整」;
    # 未来若某后端只做部分 gather, 构造期就会断言失败而不是写出残缺权重。
    return self.gather_pp and self.gather_tp and self.gather_ep

```

```

class HfWeightIteratorBase(ABC):
    def iter_hf_weights(self, weights, *, include_base=True, adapters=(), materialize=True):
        def prefixed_units(lora_name, adapter):
            # 关键修复：把 lora_name 作为函数形参绑定，而不是在 for 循环里用生成器表达式
            # 捕获循环变量。生成器是惰性求值的，等到下游消费时 lora_name 已经是最后一个值，
            # 多个 adapter 会全部顶着同一个前缀（joyliu-q 在 review 中给出的复现就是这个）。
            for unit in self._iter_hf_adapter_units(adapter, materialize=materialize):
                yield [(f"{lora_name}:{name}", tensor) for name, tensor in unit]

        hf_param_units = self._iter_hf_param_units(weights, materialize=materialize) if include_base else iter(())
        for lora_name, adapter in adapters:
            hf_param_units = itertools.chain(hf_param_units, prefixed_units(lora_name, adapter))
        atomic_update_groups = self._hf_atomic_update_groups() if include_base and materialize else []
        hf_param_units = assemble_atomic_update_groups(hf_param_units, atomic_update_groups)
        yield from pack_units_by_size(hf_param_units, self.args.update_weight_buffer_size)

    def materialize_adapter(self, adapter, *, materialize: bool = True) -> dict[str, torch.Tensor]:
        """One adapter as ``{hf_key: tensor}``, skipping the transport bucketing.
        Collective: ``materialize=False`` joins the gathers, returns {}."""
        # 绕过 pack_units_by_size 的分桶，直接得到裸 HF 名字的字典，
        # 正好对应 adapter_model.safetensors 的键名；
        # non-writer rank 传 materialize=False：参与集合通信但返回空字典。
        return {
            name: tensor
            for unit in self._iter_hf_adapter_units(adapter, materialize=materialize)
            for name, tensor in unit
        }

    @abstractmethod
    def _iter_hf_adapter_units(self, adapter, *, materialize: bool):
        # 契约变化：不再接收 lora_name、不再负责加前缀，只产出 rank-trimmed 的裸 hf_key。
        # 前缀拼接连同 adapter 归属判断一起收归基类，rollout 侧的命名由调用方决定。
        raise NotImplementedError

```

miles/backends/megatron_utils/lora/actor.py

多 LoRA 训练 actor 的接线点：_init_training_state 里独立于传输层构造 WeightPublisher，新增 export_slot 作为唯一的导出边界并让异常穿透 TrainerCell。

```

# miles/backends/megatron_utils/lora/actor.py (多 LoRA 训练 actor)
class MultiLoRATrainRayActor(MegatronTrainRayActor):
    def _init_training_state(self) -> None:
        args = self.args
        self.slot_optimizers: dict[int, SlotOptimizer] = {}
        # 发布者与传输层解耦：只依赖 HF 权重迭代器 + LoRA 配置，
        # 不再有 update window / engine session，因此也不需要再在推送热路径上缓存注册名。
        iterator = get_hf_weight_iterator(
            args,

```

```

self.model,
required_placement=WeightUpdatePlacement(gather_pp=True),
model_name=type(self.hf_config).__name__.lower() if args.model_name is None else
args.model_name,
quantization_config=getattr(self.hf_config, "quantization_config", None),
)
self.weight_publisher = WeightPublisher(iterator, build_lora_sync_config(args))

@with_logs
def export_slot(self, slot: int, rank: int, alpha: float, path: str, metadata: dict | None = None)
-> None:
    """Write the slot's adapter as an engine-loadable dir."""
    self._heartbeat.bump()
    # export_slot 自己拥有导出边界：检查点 IO 失败直接抛异常穿过 TrainerCell,
    # 不再像旧的 staged-push 路径那样被折叠成结果字典后静默继续。
    self.weight_publisher.publish_adapter(AdapterSpec(slot=slot, rank=rank, alpha=alpha),
    path, metadata=metadata)

```

评论区精华

1) joyliu-q 在 `hf_weight_iterator/_init_.py` 上给出最小复现，指出生成器惰性求值会让多个 adapter 共享最后一个 `lora_name`：消费发生在循环之后，输出 `['B:w1', 'B:w2', 'B:w1', 'B:w2']` 而不是 `['A:w1', 'A:w2', 'B:w1', 'B:w2']`，并附 Goloopvar-preview 作为更详细解释。最终版本把拼接抽成局部函数 `prefixed_units(lora_name, adapter)`（形参绑定），并补 `test_each_adapter_keeps_its_own_prefix` 回归。2) nanjiangwill 在 `updater.py` 上指出 `_registered_adapters` 在 staged stream 与 commit 完成前就写入缓存：失败重试会命中缓存跳过注册、只发 begin/end；复用已发布名字还会跳过 `defer_publish` 的新鲜度校验，建议不要缓存、让 SGLang 自己保证唯一性并补失败重试 / 重名测试。最终版本选择整体删除 Tinker staged-push 路径。3) nanjiangwill 在 `actor.py` 的 `push_slot` 上指出 `wait_futures()` 只捕获传输层异常，SGLang 可以返回 HTTP 200 但 body 为 `success: false`，这些结果在广播成功前被丢弃，建议走 `check_weight_sync_results(..., is_lora=True)`。该 push 路径同样被删除。4) nanjiangwill 在 `group.py` 的非阻塞生命周期问题：失败热推送的恢复假设 abort 让引擎回到干净状态，但 `EngineWeightUpdateSession.__exit__()` 会吞掉 abort 失败且 `_updatable_engines()` 仍标记 ready，可能让后续同名磁盘回填失败；建议补 abort 失败测试或明确该场景需要替换引擎。最终版本移除了 `_updatable_engines/update window`，恢复语义转移到 #2845 的 runtime，本 PR 未给出引擎替换策略。5) yushengsu-thu 建议对照 sglang 的 #38165 与 miles 的 #3127 (immutable adapter versions alongside generation) 的实现方式；随后给出 APPROVED。

- 生成器惰性求值导致多 adapter 前缀串扰 (correctness): 修复方式是把拼接抽成局部函数 `prefixed_units(lora_name, adapter)`（形参绑定），并在 `TestIterHfWeightsTemplate` 增加 `test_each_adapter_keeps_its_own_prefix` 回归。
- staged 发布名缓存导致失败重试跳过注册 (correctness): 最终版本整体删除 Tinker staged-push 路径，updater 回归普通 base 同步，该问题随代码删除而消失。
- HTTP 200 + success=false 的推送结果被丢弃 (correctness): 对应的 `push_slot` 代码在最终版本被删除；该结果校验要求需要由后续 runtime（#2845）的模型加载路径承接，本

PR 未直接验证。

- abort 失败后引擎仍被标记 ready，磁盘回填可能失败 (design): 最终版本移除了 `_updatable_engines/update window` 相关改动，恢复语义转移给 #2845 的 runtime；本 PR 未定义引擎替换策略，属于残留未决项。
- 对照 `sglang / miles` 的既有实现路线 (question): 本 PR 采纳了「不可变版本目录 + 磁盘回填、引擎按需加载」的路线，并在此后给出 APPROVED。

风险与影响

• 风险:

1. 发布链路失败即终止训练: `export_slot` 不再把检查点 IO 失败转成结果字典 (PR body : 「`export_slot no longer converts checkpoint IO failures into result dictionaries`」) , `write_checkpoint_dir`、`safetensors` 序列化、文件写入、目录发布任一环节出错都会作为异常穿过 `TrainerCell` 终止服务，文件系统抖动会被放大为训练中断。
2. 多 LoRA 配置语义收紧: `ModelConfig.resolve` 在 `is_multi_lora_enabled(args)` 时强制 `update_weights=False` 并断言 `effective_model_path == args.hf_checkpoint`；原先自定义 `model_path` 只打 warning 并默认 `False`，现在会直接 `assert` 失败，属于对既有 multi-LoRA 配置的兼容性破坏。
3. 公共符号 `slot_lora_name` 从 `miles/utils/multi_lora.py` 的 `__all__` 移除，本 PR 未在同仓给出替代定义；若还有 `rollout`、`session` 或脚本侧未迁移的 `import`，会直接 `ImportError`。
4. `materialize=is_writer` 的隐式集合通信契约: `non-writer rank` 走 `materialize=False` 必须仍执行同样的 `gather`，任一侧提前返回或抛错都会让 NCCL 挂死；当前实现正确，但后续修改 `materialize_adapter` 时容易踩坑。
5. `max_loaded_loras` 默认 `2 * max_loras_per_batch` 是带 TODO 的启发式，`slot` 数超过该值时 SGLang 会触发淘汰 / 重载，表现为 on-demand load 抖动，且该值直接影响显存占用。
6. 测试覆盖缺口: 本 PR 仅更新两个 fast 测试，`WeightPublisher.publish_adapter` 与 `export_slot` 的回归（进程在发布前后退出、重复保存）按 PR body 位于 #2849 的 `checkpoint` 测试文件，单独合并本 PR 时磁盘发布路径缺少直接单测。
7. 对已有脚本的迁移: `miles/utils/lora.py` 新增 `save_adapter_to_disk` 与 `WeightPublisher` 内部写盘逻辑存在两套写 `adapter` 目录的实现，若长期并存容易出现配置字段（如 `r/lora_alpha` 是否覆盖）不一致。
 - 影响: 用户侧: Tinker 用户可以在训练进行中让引擎加载某个固定版本 `adapter`，不再受 `push window` 与 `update session` 影响；但多 LoRA 用户必须满足「引擎加载与训练一致的冻结 base」这一新约束，显式 `update_weights: true` 或自定义 `model_path` 的配置会启动失败。系统侧: 训练与推理通过文件系统解耦，`engine` 从被动 `push` 变为按需拉取，减少了训练侧同步阻塞，代价是 `rank 0` 上多一次完整 `adapter` 的 D2H 拷贝与 `safetensors` 写盘（由增量目录 + 原子改名保证读侧一致性）。团队侧: 发布者、迭代器、runtime 三层职责被切开，后续 5/7、6/7 只消费 `export_slot` 与快照路径；同时失败语义从「返回结果字典」变成「抛异常终止 cell」，运维需要据此调整告警与重启策略。
 - 风险标记: 权重发布路径重写，公共符号 `slot_lora_name` 删除，多 LoRA 强制 `update_weights=False`，IO 失败直接终

止 TrainerCell, 本 PR 缺少发布器直接测试, max_loaded_loras 启发式默认值

关联脉络

- PR #2842 checkpoint IO 助手 (本 PR 的 stacked base, 标题未在本上下文提供): PR body 明确 Stacked on #2842, 并提供 write_checkpoint_dir: 保留成功写屏障与完整目录发布, 本 PR 的 WeightPublisher 直接复用它实现不可覆盖的原子发布。
- PR #2844 [multi-lora v2 5/7] Tinker gateway protocol face: 同一 stack 的前一篇, 提供进程内协议层与每模型调度器; 本 PR 移除的 slot_lora_name 等引擎侧命名职责正是由这一层与 runtime 承接。
- PR #2845 [multi-lora v2 6/7] Tinker gateway runtime and entry point: PR body 说明 runtime 会把快照名与磁盘路径随采样请求带给 SGLang 的 on-demand loading; 本 PR 的 export_slot 与冻结 base 约束是它的前置换入条件, nanjiangwill 关于 abort/ 结果校验的关注点也转移到此处。
- PR #2849 [multi-lora v2 7/7] Tinker gateway example and tests: 同一 stack 的收尾 PR: PR body 指出本 PR 的发布回归测试 (进程在链接替换前后退出、随后重试) 就落在 #2849 的 checkpoint 测试文件里, 两者共享同一套磁盘发布语义。
- PR #3127 [LoRA] Publish immutable adapter versions alongside generation: review 中 yushengsu-thu 明确建议对照的 miles 侧实现, 同样以「不可变 adapter 版本」作为与生成并行的发布方式, 是本 PR 设计取向的直接来源。