

PR #39332 完整报告

sgl-project/sglang

[PD][LoRA] Gate decode admission on adapter slots

合并时间: 2026-09-15 13:11

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39332>

执行摘要

- 一句话: PD 解码前检查 LoRA adapter 槽位, 防越限崩溃
- 推荐动作: 值得精读, 尤其关注三点: 1) `running_loras` 的统计口径 (包含 `finished` 请求) 对 "槽位即 GPU 资源" 的建模; 2) 复用调度器闸门而非另写一套判定的模块化选择; 3) `e2e` 用确定性单服务器参考做逐字符比对的测试设计。对维护 PD 调度或 LoRA 多 adapter 场景的同学有直接参考价值。

功能与动机

PR body 明确写出: "PD decode can admit more LoRA adapters than its slots allow, crashing in `fetch_new_loras`." 即 decode 侧预分配请求时只检查 KV 内存与设备缓冲, 未检查 LoRA 槽位上限, 导致 adapter 数量超限后后端在加载新 adapter 时崩溃; 同时需要保证被阻塞的请求不会 abort, 而是排队等待。

实现拆解

1. 公开调度器的 LoRA 闸门: `python/sglang/srt/managers/scheduler.py` 中把私有方法 `_can_schedule_lora_req` 重命名为 `can_schedule_lora_req`, 同步更新 `prefill` 路径 `_get_new_batch_prefill_raw` 的调用点。该方法统一了 drainer 状态检查与 `validate_lora_batch` / `try_overlap_load_lora` 的可用性判定, 公开后可供 decode 侧跨模块复用, 避免两套判定逻辑漂移。
2. decode 侧构建在途 adapter 集合: `python/sglang/srt/disaggregation/decode.py` 的 `DecodePreallocQueue.pop_preallocated` 在预分配循环前, 当 `scheduler.enable_lora` 为真时汇总所有占槽请求: PP 模式下遍历 `scheduler.running_mbs`, 否则取 `running_batch`; 再并入 `transfer_queue.queue` 与 `scheduler.waiting_queue`。注释明确包含已 `finished` 请求, 因为 GPU 上 adapter 权重可能仍在排空, 提前释放会造成后续 `fetch_new_loras` 崩溃。
3. 逐请求闸门与槽位记账: 循环内依次检查 token pool、metadata buffer、hisparse budget 之后, 在 `_pre_alloc` 与 radix 锁定之前插入 `can_schedule_lora_req` 检查; 不满足则 `continue`, 请求保留在 `self.queue` 等待下一轮。预分配成功后立刻把 `lora_id` 加入 `running_loras`, 保证同轮循环中后续请求能看到刚被占用的槽位。
4. 测试配套: 新增 e2e 测试 `test/registered/e2e/disaggregation/test_disaggregation_lora.py`, 用 3 个真实 adapter + base 模型在单服务器上生成确定性参考输出, 再以 PD 双角色并发重放并逐字符比对; 新增单元测试 `test_prealloc_lora_slots_cover_inflight_microbatch`

`hes_and_queues` 覆盖 `running / finished / transferring / waiting` 四种占槽场景；既有 `decode` 队列清理测试统一补 `scheduler.enable_lora = False` mock，避免新分支导致 `AttributeError`。

关键文件：

- `python/sglang/srt/disaggregation/decode.py`（模块 解码调度；类别 `source`；类型 `core-logic`；符号 `pop_preallocated`）：核心修复点：
`DecodePreallocQueue.pop_preallocated` 在 KV 预分配与 `radix` 锁定之前新增 LoRA 槽位闸门，并跨 `microbatch` 与 `transfer/waiting` 队列统计在途 `adapter`，是防止 `fetch_new_loras` 崩溃的关键改动。
- `python/sglang/srt/managers/scheduler.py`（模块 调度器；类别 `source`；类型 `core-logic`；符号 `can_schedule_lora_req`）：将原有 LoRA 调度判定方法 `_can_schedule_lora_req` 公开为 `can_schedule_lora_req`，使 `decode` 侧能复用同一套 `drainer + adapter` 可加载性判定，是本次无重复逻辑的关键前置改动。
- `test/registered/e2e/disaggregation/test_disaggregation_lora.py`（模块 集成测试；类别 `test`；类型 `test-coverage`；符号 `generate, TestDisaggregationLoRA, test_more_adapters_than_slots_match_single_server`）：新增 GPU `e2e`：8 个并发请求（`base + 3 adapter、2 slot`）与单服务器确定性参考逐字符比对，并校验双角色进程健康，是该 `bugfix` 最重要的端到端回归保障。
- `test/registered/unit/managers/test_priority_scheduling_disaggregation.py`（模块 单元测试；类别 `test`；类型 `test-coverage`；符号 `test_prealloc_lora_slots_cover_inflight_microbatches_and_queues`）：新增单元测试覆盖 `running / finished / transferring / waiting` 四种在途请求占槽场景，验证 `blocked` 请求留在队列、可调度请求正常进入预分配，是核心逻辑的最小回归保障。
- `test/registered/unit/disaggregation/test_decode_queue_cleanup.py`（模块 单元测试；类别 `test`；类型 `test-coverage`）：为既有 `decode` 队列清理测试的 mock `scheduler` 补充 `enable_lora = False`，避免新增分支读取未定义属性导致回归测试失败，属于配套兼容修改。

关键符号：`can_schedule_lora_req, pop_preallocated, test_more_adapters_than_slots_match_single_server, test_prealloc_lora_slots_cover_inflight_microbatches_and_queues`

关键源码片段

`python/sglang/srt/disaggregation/decode.py`

核心修复点：`DecodePreallocQueue.pop_preallocated` 在 KV 预分配与 `radix` 锁定之前新增 LoRA 槽位闸门，并跨 `microbatch` 与 `transfer/waiting` 队列统计在途 `adapter`，是防止 `fetch_new_loras` 崩溃的关键改动。

```
# 启用 LoRA 时，先汇总所有“在途”请求占用的 adapter 槽位，
# 后续循环据此判断新请求能否被接纳。
if self.scheduler.enable_lora:
    # 流水线并行下每个 microbatch 都占用 GPU 资源；
    # 非 PP 模式只需检查当前 running_batch。
    running_batches = (
```

```

        self.scheduler.running_mbs
        if is_pp_mode
        else (self.scheduler.running_batch,)
    )
    # 包含已 finished 请求: GPU 上可能仍有未排空的
    # adapter 权重在工作, 过早释放槽位会导致 fetch_new_loras 崩溃。
    running_loras = {
        req.lora_id for batch in running_batches for req in batch.reqs
    }
    # transfer_queue 与 waiting_queue 中的请求同样占用槽位预算
    running_loras.update(r.req.lora_id for r in self.transfer_queue.queue)
    running_loras.update(req.lora_id for req in self.scheduler.waiting_queue)

for i, decode_req in enumerate(self.queue):
    # ... 前面的 token pool / metadata buffer / hisparse budget 检查 ...

    # 槽位不足时跳过请求, 留在队列等待后续轮次,
    # 而不是进入 KV 预分配流程, 避免 fetch_new_loras 崩溃
    if self.scheduler.enable_lora and not self.scheduler.can_schedule_lora_req(
        decode_req.req, running_loras
    ):
        continue

    # ... 预分配与 send_metadata ...

    # 预分配成功后立即记账, 保证同轮循环中
    # 后续请求能看到刚被占用的 adapter 槽位
    if self.scheduler.enable_lora:
        running_loras.add(decode_req.req.lora_id)

```

python/sglang/srt/managers/scheduler.py

将原有 LoRA 调度判定方法 `_can_schedule_lora_req` 公开为 `can_schedule_lora_req`, 使 decode 侧能复用同一套 drainer+adapter 可加载性判定, 是本次无重复逻辑的关键前置改动。

```

def can_schedule_lora_req(
    self, req: Req, running_loras: set[Optional[str]]
) -> bool:
    """
    检查一个 LoRA 请求当前是否可调度。

    两个条件:
    1. drainer 允许调度 (基于 draining 状态);
    2. adapter 可以加载 (已在运行中, 或能加入新的 batch)。
    """
    if self.lora_drainer and not self.lora_drainer.can_schedule(req):
        return False

    # 已在 running_loras 中的 adapter 直接放行, 不重复占用新槽位
    if req.lora_id in running_loras:

```

```

        return True

    if self.enable_lora_overlap_loading:
        # 重叠加载模式下逐个尝试加载 adapter，避免批量加载导致显存峰值
        return self.lora_overlap_loader.try_overlap_load_lora(
            req.lora_id, running_loras
        )
    else:
        # 非重叠模式：询问 LoRA 管理器，当前集合能否作为一个 batch 加载
        new_lora_set = {req.lora_id} | running_loras
        return self.tp_worker.model_runner.lora_manager.validate_lora_batch(
            new_lora_set
        )

```

test/registered/e2e/disaggregation/test_disaggregation_lora.py

新增 GPU e2e：8 个并发请求（base + 3 adapter、2 slot）与单服务器确定性参考逐字符比对，并校验双角色进程健康，是该 bugfix 最重要的端到端回归保障。

```

def test_more_adapters_than_slots_match_single_server(self):
    """超额 adapter 请求必须等待，不能中止或改变输出。"""
    names = [None, *ADAPTERS]
    # 第一步：用单服务器（非 PD）确定性推理生成参考输出。
    # 同一 prompt + adapter 组合在 PD 模式下必须逐字符一致。
    process = popen_launch_server(
        self.model,
        self.base_url,
        timeout=DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH,
        other_args=LORA_ARGS,
    )
    try:
        expected = {
            (prompt, name): generate(self.base_url, prompt, name)
            for prompt in PROMPTS
            for name in names
        }
    finally:
        terminate_and_kill_process_tree(process)

    # 校验 adapter 确实生效：同一 prompt 下 adapter 输出必须与 base 不同
    for name in ADAPTERS:
        self.assertTrue(
            any(
                expected[prompt, name] != expected[prompt, None]
                for prompt in PROMPTS
            ),
            name,
        )

    # 第二步：PD 模式下并发打出 8 个请求（base + 3 adapter）× 2 轮，

```

```
# 每个请求的输出必须与单服务器参考完全一致。
self.launch_all()
jobs = list(zip(PROMPTS, names)) * 2
with ThreadPoolExecutor(max_workers=len(jobs)) as pool:
    texts = list(pool.map(lambda job: generate(self.lb_url, *job), jobs))
for job, text in zip(jobs, texts):
    self.assertEqual(text, expected[job], job)

# 压力过后两个角色进程都必须仍然健康
assert_process_healthy(self, "prefill", self.process_prefill, self.prefill_url)
assert_process_healthy(self, "decode", self.process_decode, self.decode_url)
```

评论区精华

评审 ShangmingCai 一次性 approve, 原话 "Everything looks good.", 无任何 review 评论, 说明实现无争议。PR body 中作者明确 "Decode-side starvation handling remains a follow-up": 被闸门拦下的 adapter 请求没有公平性保障, 可能长期排队, 这是本次改动遗留的最大未决问题。另外 AMD ROCm 10 CI 运行失败, 作者通过 [/rerun-failed-ci](#) 重跑, 最终 Base / Extra CI 均绿后合并, 失败项未阻塞。

- 评审一次性通过, 无争议 (other): 无修改要求, PR 按原样合并。
- decode 侧 starvation 处理留待后续 (design): 本次仅保证不崩溃、不 abort, 公平调度作为已知后续工作, 未在本 PR 解决。
- AMD ROCm 10 CI 失败与重跑 (other): 失败未阻塞合并, 属环境性或既有问题, 与本次 LoRA 改动无直接关联。

风险与影响

- 风险: 回归风险: decode.py 现在直接读取 scheduler.enable_lora, 任何手工构造 Scheduler mock 的测试或第三方嵌入代码若不设置该属性会抛 AttributeError——本 PR 已同步修补 3 处既有测试, 但仓库外调用方需注意。starvation: 被闸门拦下的请求没有超时或优先级机制, 长期满载场景下可能出现 decode 侧饥饿, 作者已承认留作后续。性能: 每轮 pop_preallocated 构建 running_loras 集合的复杂度为 $O(\text{在途请求数})$, 且逐请求调用一次 can_schedule_lora_req, 对调度热路径有轻微额外开销, 量级很小。一致性: prefill 与 decode 两个角色必须启用相同 LoRA 配置与 adapter 集合, 否则 decode 端槽位判定与实际加载结果可能不一致。CI 成本: 新增 e2e 注册为 base-b 阶段 2-gpu-large、约 400 s 的 GPU 用例。
- 影响: 用户影响: PD + LoRA 部署不再因 adapter 超限崩溃, 超额请求以排队方式降级, 且 e2e 保障了输出与单服务器确定性推理逐字符一致。系统影响: decode 侧调度循环新增一道 LoRA 闸门, 槽位语义从 "运行中独占" 扩展为 "运行中 + 排队中 + 传输中 + 排空中 adapter 都占槽", 对 PP 多 microbatch 场景尤其关键。团队影响: 后续 PD LoRA 公平调度 (starvation 处理) 将以此为基座, 新增 GPU e2e 为回归兜底, 同时多 worker 配置一致性成为部署前提。
- 风险标记: decode 调度热路径变更, LoRA 请求可能饥饿等待, 多 worker LoRA 配置一致性要求, 新增 GPU e2e 测试成本

关联脉络

- PR #39318 Scope prefetch cache state to the request attempt: 同为 PD decode 调度路径 (scheduler / decode) 的稳定性修复，与本 PR 在同一批 PD 健壮性工作线上，共享 running_batch 与队列语义。
- PR #39357 [PD] Preserve the prefill rank during rebootstrap: PD 调度修复系列之一，与本 PR 同属 PD 拆分解码稳定性演进，后续饥饿处理可能与其队列逻辑交互。
- PR #39368 [CI][PD] Skip the flaky decode HiCache file-backend disaggregation test: 同一 PD 测试维护面，说明 PD 路径近期处于高频修复阶段，本 PR 新增的 LoRA e2e 也进入该 CI 区域。