PR #35840 完整报告

sgl-project/sglang

Add PD test for inkling with mxfp8 KV

合并时间：2026-08-25 00:12

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35840

执行摘要

- 一句话：新增 Inkling MXFP8 KV 的 PD 测试，修复 retraction 状态备份
- 推荐动作：值得精读。这是一个典型的 " 测试驱动发现缺陷并落地通用修复 " 的 PR：值得关注的点包括——如何通过有界设备池刻意制造 retraction 让测试有守卫价值；cpu_copy_carries_mamba 契约标志如何避免 KV pool 与调用方重复备份或双双漏备份；uses_ssm_state 统一判定如何防止 host_pool 后端误伤带 recurrent 状态的模型；以及单元测试用桩对象精确刻画两种备份场景的手法。

功能与动机

PR body 明确说明：Inkling 是多组件状态传输最宽的单一模型用例，一个请求携带主 KV 列表加四个状态组件，MXFP8 KV 下覆盖了五类索引负载构建器中的三类；此前没有任何 PD 配置启动过该模型，组件注册、逐组件索引负载以及背后的页粒度切片在交互路径上没有端到端守卫，而最近的两个缺陷恰恰都发生在这个交互路径上。作者还强调：A pool sized to avoid retraction would have passed and guarded nothing，即有界设备池是让 retraction 真正发生、从而让测试具备守卫价值的关键。MXFP8 KV 需要 SM100+，因此用例注册在 Blackwell runner 上。

实现拆解

1. 新增 PD 端到端测试（test/registered/disaggregation/test_disaggregation_inkling_mxfp8.py）：注册到 Blackwell 4-gpu-b200 runner（register_cuda_ci(est_time=800, stage="extra-b", runner_config="4-gpu-b200")），TP=2 双角色串行启动以避免同时加载 checkpoint 超出 runner 主机内存；prefill 角色启用 --enable-hierarchical-cache，decode 角色不启用（滑动窗口模型拒绝 decode 的 radix opt-in）；关键参数为 --kv-cache-dtype mxfp8、--max-total-tokens 65536（有界设备池迫使 retraction 真实发生）、--swa-full-tokens-ratio 0.1、--mamba-full-memory-ratio 0.1；精度闸门为 GSM8K 10-shot、阈值 0.80，与单机 Inkling 用例的 0.900 对齐。
2. 修复 retraction 时 recurrent 状态丢失（python/sglang/srt/managers/schedule_batch.py、mem_cache/memory_pool.py、mem_cache/common.py）：KVCache 新增 cpu_copy_carries_mamba 契约标志，HybridLinearKVPool 将其置为 True；Req 新增 _mamba_pool_needing_backup，判断 KV pool 是否已携带 mamba 状态，未携带时从 HybridReqToTokenPool.mamba_pool 单独备份；RetractionBackup 新增 mamba_cpu 字段，offload_kv_cache/load_kv_cache 中按标志决定是否额外搬运 recurrent 状态，避免

状态留在被复用槽位上造成错位。

3. 修复 host_pool 回退后端的选择（python/sglang/srt/mem_cache/kv_cache_builder.py）：新增 uses_ssm_state(model_config) 统一判定模型是否带 recurrent 状态（GDN、Mamba2、radix cache、Kimi linear、Lightning 等配置）；resolve_decode_retraction_backup 中 supports_host_pool 追加 not uses_ssm_state(...) 条件，因为 host_pool 回退只传输 full 与 sliding-window 两类 KV 分量，带 SSM 状态的模型强制退回 cpu_tensor 后端，避免丢状态。
4. 新增单元测试（test/registered/unit/mem_cache/test_retraction_mamba_backup.py）：用桩 _MambaPool/_Allocator 覆盖两种场景——KV pool 不携带 mamba 状态时状态必须单独备份并恢复、携带时不得二次复制；注册到 CPU CI（est_time=5）。
5. CI 迭代验证：共 4 次 rerun-test，前 3 次在 4-gpu-b200 上失败，最后 1 次通过；失败与通过过程见评论区精华。

关键文件：

- python/sglang/srt/managers/schedule_batch.py（模块 调度器；类别 source；类型 core-logic；符号 _mamba_pool_needing_backup, offload_kv_cache, load_kv_cache）：核心修复点：Req 的 offload_kv_cache/load_kv_cache 新增 mamba 状态单独备份 / 恢复逻辑，_mamba_pool_needing_backup 按 cpu_copy_carries_mamba 决定是否需要额外搬运 recurrent 状态，是本次缺陷修复的主路径。
- python/sglang/srt/mem_cache/kv_cache_builder.py（模块 缓存构建；类别 source；类型 core-logic；符号 uses_ssm_state, resolve_decode_retraction_backup）：新增 uses_ssm_state 统一模型 SSM 状态判定，并据此在 resolve_decode_retraction_backup 中把带 SSM 状态的模型从 host_pool 后端排除，避免回退时丢失 recurrent 状态。
- test/registered/disaggregation/test_disaggregation_inkling_mxfp8.py（模块 PD 分离；类别 test；类型 test-coverage；符号 TestDisaggregationInklingMXFP8, setUpClass, start_prefill, start_decode）：PR 的核心交付物：首次为 Inkling + MXFP8 KV 建立 PD 端到端测试，覆盖多组件状态传输的最宽路径，并刻意触发 retraction 以验证状态备份。
- test/registered/unit/mem_cache/test_retraction_mamba_backup.py（模块 回退备份；类别 test；类型 test-coverage；符号 _MambaPool, init, get_cpu_copy, load_cpu_copy）：针对 retraction 中 mamba 状态备份行为的快速单元测试，用桩对象覆盖两种场景（KV pool 是否携带 mamba 状态），保障新备份逻辑的正确性。
- python/sglang/srt/mem_cache/memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 KVCache.cpu_copy_carries_mamba）：为 KVCache 增加 cpu_copy_carries_mamba 标志并让 HybridLinearKVPool 置 True，这是备份逻辑判断的契约基础。
- python/sglang/srt/mem_cache/common.py（模块 通用层；类别 source；类型 core-logic；符号 RetractionBackup.mamba_cpu）：RetractionBackup 增加 mamba_cpu 字段，承载单独备份的 recurrent 状态，是备用数据结构的关键扩展。

关键符号：uses_ssm_state, _mamba_pool_needing_backup, Req.offload_kv_cache, Req.load_kv_cache, TestDisaggregationInklingMXFP8.setUpClass, TestDisaggregationInklingMXFP8.start_prefill, TestDisaggregationInklingMXFP8.start_decode,

TestRetractionMambaBackup.test_state_travels_when_kv_pool_leaves_it_behind,TestRetractionMambaBackup.test_state_is_not_copied_twice_when_the_kv_pool_carries_it

关键源码片段

python/sglang/srt/managers/schedule_batch.py

核心修复点：Req 的 offload_kv_cache/load_kv_cache 新增 mamba 状态单独备份 / 恢复逻辑，_mamba_pool_needing_backup 按 cpu_copy_carries_mamba 决定是否需要额外搬运 recurrent 状态，是本次缺陷修复的主路径。

```
def _mamba_pool_needing_backup(self, req_to_token_pool, allocator):
    """判断 KV pool 的 get_cpu_copy/load_cpu_copy 是否已携带 recurrent 状态。

    cpu_copy_carries_mamba 是 KV pool 的契约标志：True 表示 pool 自己会搬运
    ShortConv 状态，调用方无需再碰 mamba_pool；False 时只有 HybridReqToTokenPool
    才可能有独立的 mamba_pool 需要单独备份。
    """
    if allocator.get_kvcache().cpu_copy_carries_mamba:
        return None
    if not isinstance(req_to_token_pool, HybridReqToTokenPool):
        return None
    return req_to_token_pool.mamba_pool


def offload_kv_cache(self, req_to_token_pool, token_to_kv_pool_allocator):
    token_indices = req_to_token_pool.req_to_token[
        self.req_pool_idx, : self.seqlen - 1
    ]
    # 先判断状态是否需要单独搬，再同时备份 KV 与 recurrent 状态
    mamba_pool = self._mamba_pool_needing_backup(
        req_to_token_pool, token_to_kv_pool_allocator
    )
    self.retraction_backup = RetractionBackup(
        cpu_tensors=token_to_kv_pool_allocator.get_cpu_copy(
            token_indices, mamba_indices=self.mamba_pool_idx
        ),
        mamba_cpu=(
            mamba_pool.get_cpu_copy(self.mamba_pool_idx.unsqueeze(0))
            if mamba_pool is not None and self.mamba_pool_idx is not None
            else None
        ),
    )


def load_kv_cache(self, req_to_token_pool, token_to_kv_pool_allocator):
    token_indices = req_to_token_pool.req_to_token[
        self.req_pool_idx, : self.seqlen - 1
    ]
```

```
    # 恢复顺序不能反：先把 recurrent 状态放回 mamba_pool 再加载 KV，
    # 否则复用槽位上残留的旧状态会被下一次 forward 读到
    mamba_cpu = self.retraction_backup.mamba_cpu
    if mamba_cpu is not None and self.mamba_pool_idx is not None:
        req_to_token_pool.mamba_pool.load_cpu_copy(
            mamba_cpu, self.mamba_pool_idx.unsqueeze(0)
        )
    # 加载 KV；对不携带 mamba 状态的 pool，mamba_indices 参数会被忽略
    token_to_kv_pool_allocator.load_cpu_copy(
        self.retraction_backup.cpu_tensors,
        token_indices,
        mamba_indices=self.mamba_pool_idx,
    )
```

python/sglang/srt/mem_cache/kv_cache_builder.py

新增 uses_ssm_state 统一模型 SSM 状态判定，并据此在 resolve_decode_retraction_backup 中把带 SSM 状态的模型从 host_pool 后端排除，避免回退时丢失 recurrent 状态。

```
BACKUP_ONLY_HICACHE_RATIO = 0.2


def uses_ssm_state(model_config) -> bool:
    """模型是否在 attention KV 之外还维护 recurrent/conv 状态。

    host_pool 回退后端只搬运 full 与 sliding-window 两类 KV 分量，
    无法携带 recurrent 状态，因此带 SSM 状态的模型必须走 cpu_tensor 后端，
    由调用方（Req 的 offload/load）单独备份 mamba 状态。
    """
    spec = linear_attn_model_spec(model_config)
    return (
        hybrid_gdn_config(model_config) is not None
        or mamba2_config(model_config) is not None
        or (spec.uses_mamba_radix_cache if spec is not None else False)
        or kimi_linear_config(model_config) is not None
        or hybrid_lightning_config(model_config) is not None
    )


def resolve_decode_retraction_backup(*, tp_worker: BaseTpWorker) -> str:
    disagg = get_disagg()
    memory = get_memory()
    fields = {}

    backend = disagg.disaggregation_decode_retraction_backup
    if backend is None:
        kv_cache = tp_worker.get_memory_pool()[1].get_kvcache()
        full_tokens_per_layer = (
            tp_worker.get_tokens_per_layer_info()[0]
```

```
        if tp_worker.is_hybrid_swa
        else None
    )
    # 关键过滤：host_pool 只覆盖 full + sliding-window，因此
    # 带 recurrent 状态的模型即使池类型满足条件也退回 cpu_tensor
    supports_host_pool = not uses_ssm_state(
        tp_worker.model_runner.model_config
    ) and (
        isinstance(kv_cache, MHATokenToKVPool)
        or (isinstance(kv_cache, SWAKVPool) and full_tokens_per_layer > 0)
    )
    # 其余后端选择逻辑（cpu_tensor / host_pool 分支）省略
    ...
    fields["disaggregation_decode_retraction_backup"] = backend
```

评论区精华

该 PR 的 review_comments_count 为 0，没有传统意义上的 review 讨论，核心交锋发生在 CI 重跑与 PR body 论证中。作者在 PR body 里明确解释了有界设备池的设计权衡：A pool sized to avoid retraction would have passed and guarded nothing——如果设备池足够大就不会触发 retraction，测试会假绿而失去守卫意义。4 次 /rerun-test（issue 评论）中前 3 次失败，暴露的正是 #35888 描述的 MXFP8 KV 拒绝 CPU offload 的 NotImplementedError；#35888 修复后最后一次 rerun 通过，gsm8k 从 0.12 恢复到 0.855。

- CI 反复失败：MXFP8 KV retraction 崩溃 (testing): #35888 实现 MXFP8 KV 的 CPU offload 并同步备份 block-scale 分量后，第 4 次 rerun（run 32688047831）通过，gsm8k 恢复 0.855。作者在 PR body 中强调：有界设备池是让 retraction 真正发生、从而让测试有守卫价值的关键。

风险与影响

- 风险：
  1. 核心 retraction 路径变更：schedule_batch.py 的 offload_kv_cache/load_kv_cache 是所有带 SSM 状态模型（Mamba、HybridLinear、Kimi、Lightning、GDN 等）共用的 retraction 路径，备份 / 恢复顺序或 cpu_copy_carries_mamba 标志判断错误会导致状态被放到错误槽位；此类错误不抛异常，只表现为生成质量下降，回归难以发现。
  2. 状态识别覆盖面：uses_ssm_state 依赖各模型配置 helper（hybrid_gdn_config、mamba2_config、linear_attn_model_spec 等）的完整覆盖，未来新增 SSM 模型时容易漏配，导致 host_pool 后端被误选而丢状态。
  3. 硬件与配置依赖：测试仅覆盖 MXFP8 KV（bf16 依赖已有 disaggregation 测试），且强依赖 Blackwell（SM100+）专属 runner，硬件可用性影响 CI 稳定性；est_time=800 较长。
  4. 测试守卫有效性：测试刻意用 --max-total-tokens 65536 制造 retraction，未来内存布局或参数语义变化可能使 retraction 不再触发，测试会静默失去守卫作用。 - 影响：用户侧：修复了 PD 模式下带 SSM 状态模型在设备池压力下 retraction 导致调度器崩溃、后续请求全部失败的问题，gsm8k 从 0.12 恢复到 0.855。系统侧：为 retraction 备份

机制引入了通用契约（cpu_copy_carries_mamba 标志 +RetractionBackup.mamba_cpu 字段），后续新增 KV pool 类型必须显式声明是否携带recurrent 状态，否则默认走调用方单独备份的路径。团队侧：首次在 MXFP8 KV + PD链路上建立端到端精度闸门，SWA/MAMBA/BLOCK_SCALE/BLOCK_SCALE_SWA 多组件状态传输从此有回归防线，也为后续其他混合状态模型的 PD 测试提供了模板。 - 风险标记：核心 retraction 路径变更，状态识别需随新模型扩展，Blackwell 专属硬件依赖 ,仅覆盖 MXFP8 配置

关联脉络

- PR #36203 Cleanup duplicate mamba backup helper: 本 PR 在 schedule_batch.py 中引入了 mamba 状态备份逻辑，后续 #36203 清理了该文件中重复的 mamba backup 辅助函数，说明 retraction 备份逻辑仍在持续演进。
- PR #35888 Support CPU offload for mxfp8 KV cache: 本 PR 的 PD 测试是 #35888 缺陷的直接触发者（#35888 body 明确写到 Surfaced by the pair test in #35840）；#35888 修复后本测试才转绿，构成测试与修复的闭环。