PR #51100 完整报告

vllm-project/vllm

[Bugfix] Fix Mamba all-mode CPU offload boundary alignment

合并时间：2026-08-07 12:37

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/51100

执行摘要

- 一句话：修复 Mamba all 模式 offload 边界对齐错误
- 推荐动作：值得精读，尤其是给负责 KV offload 与状态缓存类组件的工程师。三点值得借鉴：一是 issue 驱动的根因分析路径（边界命中 = 点状态恢复 + 末 token 重算的组合），二是用 FP32 无损状态隔离精度噪声、再用负向对照证明测试有效性的方法论，三是 PR 作者对 CI 失败逐条归因（节点故障 vs 上游回归）的严谨态度。阅读重点放在 resolve_mamba_align_size() 的对齐语义与 test_mamba_cpu_offload_boundary 的断言设计上。

功能与动机

Issue #51094 报告：在 mamba_cache_mode="all" 与 OffloadingConnector + CPU KV offload + prefix caching 组合下，prompt 长度恰为 offload chunk 整数倍时，CPU 回读会产生与冷启动不一致的输出且不抛任何异常——第 5 个生成 token 从 1556 静默变为 1270。issue 明确指出根因：resolve_mamba_align_size() 只对 "align" 模式启用命中窗口对齐，而 "all" 模式同样存储 recurrent 点状态，精确边界查询可返回 N - 1 个 token 却恢复边界 N 处的状态，使最后一个 prompt token 被应用两次。issue 还给出了与最终修复一致的最小改动建议（把条件扩展为 in ("align", "all")）与回归测试思路。

实现拆解

变更入口是 vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py 中的 resolve_mamba_align_size()，配套测试在 tests/v1/kv_connector/unit/test_offloading_connector.py。

1. 根因确认：该函数按 KV cache group 扫描 MambaSpec，为需要对齐的模式计算 tokens_per_chunk（tokens_per_block * blocks_per_chunk）。原实现只对 mamba_cache_mode == "align" 生效；而 "all" 模式同样在 token/block 位置保存 recurrent 状态，导致精确 chunk 边界查询时命中窗口不收缩、直接恢复已消费最后一个 token 的点状态。Issue #51094 在 Nemotron-Nano-9B-v2 + 2816 token（2 × 1408 chunk）上复现了第 5 个生成 token 的静默变化。
2. 生产修复：把条件改为 mamba_cache_mode in ("align", "all")，使 "all" 模式的命中窗口同样向下取整到 chunk 边界；函数内对所有 Mamba 组对齐值一致的断言保持不变。该修改令边界场景从命中 N - 1 个 token 回退到命中 N - chunk 个 token，确保恢复的状态不含待重算 token。

3. 测试改造：test_mamba_align_cpu_offload 重命名为 test_mamba_cpu_offload_boundary，新增 @pytest.mark.parametrize("mamba_cache_mode", ["align", "all"])，每个模式都覆盖 block-boundary 与 block-mid 两类 prompt；同时为 LLM 增加 mamba_cache_dtype="float32"、mamba_ssm_cache_dtype="float32"，把"冷启动 vs CPU 回读"的精确相等断言从 FP16 状态压缩的精度噪声中隔离出来。测试流程仍是：冷启动生成 → 清空 GPU prefix 缓存（保留 connector 缓存）→ 再次生成 → 断言两次输出一致。
4. 验证与 CI：首次 CI 暴露的 all 用例失败被证明是 FP16 状态缓存精度问题而非边界逻辑问题；改为 FP32 后通过负向对照（回退修复则 all 用例立即失败）确认测试仍覆盖原 bug。最终所有 Mamba/KV-offload 相关测试通过，B200 节点超时与 Granite fullgraph 失败均被确认为基础设施或上游预存在问题。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 resolve_mamba_align_size）：生产修复所在：resolve_mamba_align_size() 将 mamba_cache_mode="all" 纳入命中窗口对齐，是本次静默输出错误修复的核心。
- tests/v1/kv_connector/unit/test_offloading_connector.py（模块 回归测试；类别 test；类型 test-coverage；符号 test_mamba_cpu_offload_boundary, test_mamba_align_cpu_offload）：回归测试从单一 align 模式参数化覆盖 align/all，并用 FP32 状态缓存隔离低精度 checkpoint 噪声，保证测试只针对边界选择。

关键符号：resolve_mamba_align_size, test_mamba_cpu_offload_boundary

关键源码片段

vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py

生产修复所在：resolve_mamba_align_size() 将 mamba_cache_mode="all" 纳入命中窗口对齐，是本次静默输出错误修复的核心。

生产修复核心：将 mamba_cache_mode="all" 纳入命中窗口对齐。

```
def resolve_mamba_align_size(
    spec: "OffloadingSpec", kv_cache_config: KVCacheConfig
) -> int | None:
    """Scan all KV cache groups in *spec* and return the single mamba alignment
    size, or None if no group requires mamba alignment.

    For MambaSpec groups in "align" or "all" cache mode the hit window must be
    rounded down to a multiple of the offloaded chunk size. Asserts that all
    such groups agree on the same value.
    """
    # Mamba 缓存条目是"点状态"（point state），命中窗口必须向下取整到
    # offload chunk 边界；否则恢复出的状态可能已消费最后一个 prompt token，
    # 重算时该 token 会被应用两次，输出静默改变且不抛异常。
    mamba_align_size: int | None = None
    for idx, tokens_per_block in enumerate(spec.tokens_per_block):
```

```
        kv_spec = kv_cache_config.kv_cache_groups[idx].kv_cache_spec
        # “all” 模式同样按 token/block 位置保存 recurrent 状态，
        # 因此与 “align” 一样需要边界对齐，这是本次修复的核心。
        if isinstance(kv_spec, MambaSpec) and kv_spec.mamba_cache_mode in (
            "align",
            "all",
        ):
            # 一个 offload chunk 含 tokens_per_block * blocks_per_chunk 个 token，
            # 对齐大小即该 chunk 的 token 数。
            tokens_per_chunk = tokens_per_block * spec.blocks_per_chunk
            # 所有 Mamba 组的对齐值必须一致，防止多组配置互相冲突。
            assert mamba_align_size is None or mamba_align_size == tokens_per_chunk
            mamba_align_size = tokens_per_chunk
    return mamba_align_size
```

tests/v1/kv_connector/unit/test_offloading_connector.py

回归测试从单一 align 模式参数化覆盖 align/all，并用 FP32 状态缓存隔离低精度 checkpoint 噪声，保证测试只针对边界选择。

回归测试：双模式参数化 + FP32 状态缓存隔离精度噪声。

```
@pytest.mark.parametrize("mamba_cache_mode", ["align", "all"])
def test_mamba_cpu_offload_boundary(
    model: str, block_size: int, tp_size: int, mamba_cache_mode: str
):
    # 构造 OffloadingConnector：CPU 侧预留 4 GiB 空间，按 block_size 切块。
    kv_transfer_config = KVTransferConfig(
        kv_connector="OffloadingConnector",
        kv_role="kv_both",
        kv_connector_extra_config={
            "cpu_bytes_to_use": 4 << 30,
            "block_size": block_size,
        },
    )
    llm = LLM(
        model=model,
        max_model_len=block_size * 10,
        gpu_memory_utilization=0.85,
        tensor_parallel_size=tp_size,
        kv_transfer_config=kv_transfer_config,
        language_model_only=True,
        enable_prefix_caching=True,
        mamba_cache_mode=mamba_cache_mode,
        # 使用 FP32 无损状态缓存：让“冷启动 vs CPU 回读”的精确相等断言
        # 只反映边界选择结果，排除 FP16 状态压缩的精度噪声，生产默认不变。
        mamba_cache_dtype="float32",
        mamba_ssm_cache_dtype="float32",
        disable_hybrid_kv_cache_manager=False,
    )
```

```
    _PROMPT_SIZE: int = block_size * 2 # 恰好两个 offload chunk，命中精确边界
    _PROMPT_TEXT = "Hi. Give me a set of trivia questions and their answers "

    # 用占位 token 把 prompt 补齐到目标长度，保证边界精确可控。
    tokenizer = llm.get_tokenizer()
    raw_ids: list[int] = tokenizer.encode(_PROMPT_TEXT)
    while len(raw_ids) < _PROMPT_SIZE:
        raw_ids = tokenizer.encode("....") + raw_ids
    initial_ids: list[int] = raw_ids[:_PROMPT_SIZE]

    sampling_params = SamplingParams(max_tokens=128, temperature=0, ignore_eos=True)
    failures: list[str] = []

    def _verify(llm, prompt, label: str):
        # 先冷启动生成，清空 GPU prefix 缓存但保留 connector 缓存，
        # 再生成一次强制走 CPU 回读，两次输出必须逐字节一致。
        cold_outputs = llm.generate([prompt], sampling_params, use_tqdm=False)
        _wait_for_prefix_cache_reset(llm)
        cpu_outputs = llm.generate([prompt], sampling_params, use_tqdm=False)
        cold_text = cold_outputs[0].outputs[0].text
        cpu_text = cpu_outputs[0].outputs[0].text
        if cold_text != cpu_text:
            failures.append(
                f"{label}: mismatch\n  cold: {cold_text!r}\n  cpu:  {cpu_text!r}"
            )

    # block-boundary-prompt：prompt 长度恰为 chunk 整数倍，此时 CPU offload
    # 绝不能命中边界处的缓存状态——它已包含必须重算的最后一个 token；
    # 其他 attention 类型因命中块内含全部 token KV，不受此限制。
    prompt = TokensPrompt(prompt_token_ids=initial_ids)
    _verify(llm, prompt, "block-boundary-prompt")

    # block-mid-prompt：不在边界上，复用缓存状态是安全的。
    prompt = TokensPrompt(prompt_token_ids=[0] + initial_ids)
    _verify(llm, prompt, "block-mid-prompt")

    assert not failures, "

".join(failures)
```

评论区精华

核心交锋围绕“测试失败的归因”展开：

- orozery 在首次 CI（Buildkite #82467）中报告 test_mamba_cpu_offload_boundary[all-state-spaces/mamba-1.4b-hf-16-1] 在 block-boundary-prompt 出现 cold/cpu 文本不一致。
- jairitAge 复现后指出该失败与边界修复无关：“With the default FP16 state cache, cold prefill and prefix resume can diverge even with OffloadingConnector disabled and the prefix kept entirely on GPU.”——FP16 状态 checkpoint 本身就是混淆变量；随后把

测试改为 FP32 状态缓存，并通过“回退 all 模式对齐改动后 FP32 测试立即失败”的负向对照证明测试仍然有效。

- varun-sundar-rabindranath 在 review 中提问 float32 改动是否必需（“In the PR, I see tests passing without the float32 change. is this required?”），jairitAge 说明 FP32 仅用于测试隔离、不改变生产默认后获得认可。
- CI 层面，jairitAge 逐一论证 B200 节点 dgxb200-16 多轮超时属节点故障（同一作业在 dgxB200-14 上 17 分钟通过 2432 个测试）、Granite fullgraph 失败在 #51074 的 CI 中同样出现属上游回归，均与本 PR 无关。
- all 模式用例首次 CI 失败：FP16 状态缓存噪声 vs 边界逻辑缺陷 (testing): 测试改用 mamba_cache_dtype="float32" 与 mamba_ssm_cache_dtype="float32" 后 align/all 均精确通过；以“回退 all 模式对齐改动后测试立即失败”作负向对照，确认测试仍覆盖原 bug。
- float32 测试配置是否必需 (question): 保留 float32 测试配置，reviewer 接受并 approve。
- B200 节点超时与 Granite fullgraph 失败是否预存在 (other): 判定为基础设施节点故障与上游 CI 回归，PR 相关测试全部通过，最终所有检查绿色后合并。

风险与影响

- 风险：
  - 缓存命中率影响："all" 模式命中窗口现在必须对齐到 chunk 边界，精确边界场景会从 N - 1 回退到 N - chunk（如 issue 中的 2815 → 1408），CPU 回读 token 数显著增加，可能带来额外传输与重算开销——这是换取正确性的必要代价，值得在长 prompt + offload 场景观察性能变化。
  - 行为变更面：仅影响 hybrid Mamba 模型 + OffloadingConnector + enable_prefix_caching=True + mamba_cache_mode="all" 的组合；"align"、"none" 及无 offload 路径不受影响，全注意力模型不受影响。
  - 测试隔离度：生产默认仍是 FP16 状态缓存，FP32 仅用于测试；FP16 精度噪声问题独立存在（本 PR 已证明与边界逻辑无关），未来若在 FP16 下做精确相等断言仍会踩坑。
  - 配置一致性断言：resolve_mamba_align_size() 要求所有 Mamba 组对齐值一致，若未来模型配置出现多 Mamba 组且 chunk 大小不同，assert 会直接失败，属于提前暴露而非静默错误。
  - 测试平台限制：回归测试标记为 CUDA-only，all 模式在 ROCm/XPU 等平台的 offload 行为缺乏同等覆盖。
  - 影响：该修复消除的是一个静默正确性问题：此前 CPU 回读会在无任何异常的情况下改变生成结果，对使用 hybrid Mamba 模型 + KV offload + 长 prompt 前缀复用的用户是隐性数据风险。影响范围集中在 OffloadingConnector 路径，且仅在精确 chunk 边界触发；对团队而言，测试从单一 align 模式扩展到 align/all 双模式，函数重命名后 CI 测试名由 test_mamba_align_cpu_offload 变为 test_mamba_cpu_offload_boundary，语义更准确地反映覆盖范围。
  - 风险标记：缓存命中语义变更，潜在命中率下降，测试仅限 CUDA，多 Mamba 组断言敏感

关联脉络

- PR #51113 [Bugfix] Keep mamba align prefill chunks block-aligned past last_cache_position: 同为 Mamba chunk 边界对齐类修复：一个处理 prefill 分块在 last_cache_position 后的对齐，一个处理 offload 命中窗口在 chunk 边界的对齐，共同保证 Mamba 状态不在边界上被重复消费。