PR #43281 完整报告

vllm-project/vllm

[KV Connector] Handle Mooncake finish after preemption

合并时间：2026-05-25 16:58

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43281

执行摘要

- 一句话：修复 MooncakeStore preemption 后 request_finished 断言崩溃
- 推荐动作：该 PR 值得精读，展示了如何在分布式 KV 传输组件中优雅处理 preemption 边界情况，设计决策（重置 tracker 而非移除）和 review 讨论都具有参考价值。

功能与动机

在高 KV-cache 压力下 (kv_usage ≥ 99%)，vLLM 调用 MooncakeStoreConnector.request_finished 时，对应的 request_tracker 已被移除，导致 assert tracker is not None 崩溃并引发整个 API 服务器宕机。PR body 中给出了具体的崩溃堆栈和错误信息。

实现拆解

1. 在 data.py 中为 RequestTracker 添加 reset() 方法：将 token_len、allocated_block_ids、num_saved_tokens、token_ids 和 prefill_end_tokens 全部重置为初始值，避免下次重新调度时残留旧状态。
2. 在 scheduler.py 的 build_connector_meta 中处理 preemption：对于被抢占的 req_id，不再直接 pop 移除 request_tracker，而是调用 reset() 重置其状态，同时新增清理 load_specs 的逻辑。
3. 在 scheduler.py 的 request_finished 中替换断言为安全判断：将原有的 assert tracker is not None 改为 if tracker is None 的检查，当 tracker 不存在时直接返回 False, None，避免崩溃。
4. 补充测试用例：新增 _make_preemption_scheduler_output 辅助函数和两个测试函数，分别验证 preemption 后 tracker 重置和清除 stale load state 的行为。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/mooncake/store/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 build_connector_meta, request_finished）：核心逻辑变更：替换 request_finished 中的 assert 为安全条件判断，在 build_connector_meta 中重置 tracker 而非移除。
- vllm/distributed/kv_transfer/kv_connector/v1/mooncake/store/data.py（模块 数据模型；类别 source；类型 core-logic；符号 reset）：新增 RequestTracker.reset() 方法，用于 preemption 时将 tracker 状态清零。

- tests/v1/kv_connector/unit/test_mooncake_store_scheduler.py（模块 测试；类别 test；类型 test-coverage；符号 _make_preemption_scheduler_output, test_preemption_resets_tracker_before_request_finished, test_preemption_clears_stale_load_state）：新增两个测试函数，覆盖 preemption 后 tracker 重置和 load state 清除的场景。

关键符号：RequestTracker.reset, MooncakeStoreScheduler.build_connector_meta, MooncakeStoreScheduler.request_finished

关键源码片段

vllm/distributed/kv_transfer/kv_connector/v1/mooncake/store/scheduler.py

核心逻辑变更：替换 request_finished 中的 assert 为安全条件判断，在 build_connector_meta 中重置 tracker 而非移除。

```
# vllm/distributed/kv_transfer/kv_connector/v1/mooncake/store/scheduler.py

# 在 build_connector_meta 中处理 preemption:
preempted_ids = scheduler_output.preempted_req_ids or set()
self._preempted_req_ids.update(preempted_ids)
for req_id in preempted_ids:
    self.load_specs.pop(req_id, None) # 同时清理 load_specs
    if request_tracker := self._request_trackers.get(req_id):
        request_tracker.reset() # 重置 tracker 而非移除，保持对象存在以供后续可能使用
    self._unfinished_requests.pop(req_id, None)

# request_finished 中避免断言崩溃:
def request_finished(self, request, block_ids):
    if self.kv_role == "kv_consumer":
        return False, None
    tracker = self._request_trackers.get(request.request_id)
    # tracker 可能为 None：请求被抢占、中止等场景
    if tracker is None or tracker.num_saved_tokens <= 0:
        return False, None
    total_blocks = sum(len(g) for g in block_ids)
    delay_free_blocks = total_blocks > 0
    return delay_free_blocks, None
```

vllm/distributed/kv_transfer/kv_connector/v1/mooncake/store/data.py

新增 RequestTracker.reset() 方法，用于 preemption 时将 tracker 状态清零。

```
# vllm/distributed/kv_transfer/kv_connector/v1/mooncake/store/data.py

@dataclass
class RequestTracker:
    """Tracks per-request state across scheduler ticks."""
    req_id: str
    token_len: int
    allocated_block_ids: tuple[list[int], ...]
```

```
    num_saved_tokens: int = 0
    token_ids: list[int] | None = None
    prefill_end_tokens: int = 0

    def reset(self) -> None:
        """重置所有状态，用于 preemption 后重新开始。"""
        self.token_len = 0
        self.allocated_block_ids = ()
        self.num_saved_tokens = 0
        self.token_ids = None
        self.prefill_end_tokens = 0

    def update(self, new_block_ids):
        # ... 原有 update 逻辑
```

tests/v1/kv_connector/unit/test_mooncake_store_scheduler.py

新增两个测试函数，覆盖 preemption 后 tracker 重置和 load state 清除的场景。

```
# tests/v1/kv_connector/unit/test_mooncake_store_scheduler.py

# 辅助函数：构造一个包含 preemption 信息的 scheduler_output
def _make_preemption_scheduler_output():
    return SimpleNamespace(
        finished_req_ids=set(),
        preempted_req_ids={"req-0"}, # 标记 req-0 被抢占
        scheduled_new_reqs=[],
        scheduled_cached_reqs=SimpleNamespace(
            req_ids=[], new_block_ids=[], num_computed_tokens=[]
        ),
        num_scheduled_tokens={},
        scheduled_spec_decode_tokens={},
    )

# 测试 1：抢占后 tracker 被重置，request_finished 返回 (False, None)
def test_preemption_resets_tracker_before_request_finished():
    scheduler = _make_bare_scheduler()
    _add_unfinished_request(
        scheduler,
        token_ids=list(range(44)),
        block_hashes=[b"h0", b"h1"],
        prefill_end_tokens=48,
    )
    # 执行 build_connector_meta 触发 preemption 处理
    scheduler.build_connector_meta(_make_preemption_scheduler_output())
    tracker = scheduler._request_trackers["req-0"]
    # 验证 tracker 已被重置
    assert tracker.token_len == 0
    assert tracker.allocated_block_ids == ()
    assert tracker.num_saved_tokens == 0
```

```
    assert tracker.token_ids is None
    assert tracker.prefill_end_tokens == 0
    # 验证 request_finished 不再崩溃
    request = SimpleNamespace(request_id="req-0")
    assert scheduler.request_finished(request, ([0, 1],)) == (False, None)

# 测试 2：抢占时同时清除 stale load_specs
def test_preemption_clears_stale_load_state():
    scheduler = _make_bare_scheduler()
    _make_pending_load_unfinished_request(...)
    scheduler.load_specs["req-0"] = LoadSpec(...)
    meta = scheduler.build_connector_meta(_make_preemption_scheduler_output())
    assert meta.requests == [] # 不应该有请求元数据
    assert "req-0" not in scheduler.load_specs # load_specs 被清除
    assert "req-0" not in scheduler._unfinished_requests
```

评论区精华

reviewer ivanium 建议将 request_finished 中 tracker 为 None 时 debug logging 改为注释，因为还有多种场景（如请求被 vLLM 调度器中止）也可能导致 tracker 为 None。另外建议将 reset_after_preemption 方法简化为 reset。这些建议均被采纳。

- tracker 缺失时的处理方式 (design): 最终代码采用了条件判断并返回默认值，注释说明了 tracker 可能缺失的场景。
- 方法命名 (style): 方法重命名为 reset。

风险与影响

- 风险：低风险。变更仅影响 MooncakeStore Scheduler 的 preemption 和 finish 路径，将原本的 assert 替换为条件判断，并重置 tracker 状态而非移除。回归风险主要在于其他依赖 request_tracker 的逻辑，但由于 reset 保留了 tracker 对象，且测试覆盖了关键场景，风险可控。
- 影响：直接影响使用 MooncakeStoreConnector 的高 KV-cache 压力场景，避免了 EngineCore 崩溃。对系统稳定性有正向影响。
- 风险标记：暂无

关联脉络

- PR #43494 [KV Connector] Keep MooncakeStore full hits block-aligned: 同属 MooncakeStore connector 的修复，修改了相同模块的 scheduler.py 和 test 文件。
- PR #43392 [Mooncake] Add metrics for MooncakeStoreConnector operations: 同属 MooncakeStore connector 的增强，可能影响相同模块的稳定性。