PR #52466 完整报告

vllm-project/vllm

[KV Connector] Add decode offloading to Mooncake Store consumers

合并时间：2026-08-20 08:40

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52466

执行摘要

- 一句话：Mooncake Store 消费者新增 decode KV 卸载能力
- 推荐动作：值得精读。重点关注 4 个设计决策：（1）用 store_job_id 账本替代按 req_id 计数，抵御 preemption 与请求 ID 复用带来的状态污染；（2）KV event 的 parent 关系从 request hash 链推导，而非依赖 Store 返回顺序；（3）增量 token 快照加按功能开关裁剪元数据，避免长请求全量拷贝；（4）异步 store 生命周期下“调度器只持有 token 后缀、GPU block 由 worker 报告完成才释放”的 pin 语义。

功能与动机

PR body 明确这是 Mooncake Store Connector 功能路线图 #45036 中的 “Decode-Phase KV Cache Put” 一项：改动前 kv_consumer 只能从 Store 加载 prompt KV，decode 阶段生成的 KV 只留在本地，无法被其他请求复用。原文描述："Before this change, a Mooncake Store consumer could load prompt KV cache from the Store, but KV cache generated during decode remained local." 新增 save_decode_cache 后，消费者可打开 Store 写路径，在 decode 开始后 PUT 新生成的 KV 块；同时 review 中确认，当 P-D KV 通过 MooncakeConnector 直传时 prompt KV 可能不在 Store 中，D 端必须在首次 decode 保存时补齐缺失的 block 对齐 prompt 前缀，否则后续 prefill 的连续前缀匹配无法成立。

实现拆解

1. 连接器开关与写路径（connector.py、worker.py）：MooncakeStoreConnector.__init__ 从 kv_connector_extra_config 读取 save_decode_cache，并将其纳入 _capacity_only 判定——开启后 kv_consumer 不再被当作纯容量节点，会正常进入 KV cache 形状校验并持有写能力；worker 侧相应放开 can_put 与 Store 发送线程的启动条件。对应测试 test_save_decode_cache_keeps_transfer_path_enabled、test_consumer_starts_send_thread_only_when_put_is_enabled。
2. 调度器保存条件与元数据裁剪（scheduler.py）：新增 save_decode_cache 与 enable_kv_events 两个配置解析。build_connector_meta 中原 force_skip_save 被拆为 is_consumer 与 can_process_cached：prefill 路径仍以 skip_save=is_consumer 跳过（保持 consumer 不上传 prefill KV 的既有语义），decode 路径仅在 can_process_cached 为真时构造保存元数据；enable_kv_events 关闭时 RequestTracker 不再维护 token_ids，decode 未启用时提前走 fast path 跳过 tracker 记账。

3. 数据契约增量快照（data.py）：ReqMeta 新增 token_ids_start 字段并删除遗留死变量 is_last_chunk。from_request_tracker 以 num_saved_tokens 作为绝对起始偏移，只把 [token_ids_start, num_tokens_to_save) 的 token 切片复制进元数据，避免长请求全量拷贝；由于调度器侧 tracker 会继续随 decode 增长，该切片保证异步 worker 拿到稳定快照。
4. worker 异步生命周期与失败恢复（worker.py）：KVCacheStoreSendingThread 新增 _retry_token_ids 字典及配套 _get_retry_token_ids/_update_retry_token_ids，两者均在 done_task_lock 下先校验 store_job_id 是否存活，防止 stale job 在 req_id 复用后污染重试状态；_handle_request 的 KV event 路径改为从原始 request hash 链推导 parent_block_hash，覆盖 Store 去重、稀疏 miss 与 TP-strided PUT；失败时保留 token 后缀供事件重试，成功保存后清理。
5. 测试与文档配套：test_mooncake_store_scheduler.py 与 test_mooncake_store_worker.py 新增约 344 行回归用例，覆盖默认关闭、首个 decode 保存回填 prompt、仅保存完整 block 边界、异步完成、KV event parent 链与重试恢复；docs/features/mooncake_store_connector_usage.md 补充 save_decode_cache 的双层语义。RDMA 集群 e2e 脚本在 fork 仓库提供，仓库内 CI 以单元测试与 pre-commit 为主。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/mooncake/store/worker.py（模块 卸载线程；类别 source；类型 core-logic；符号 KVCacheStoreSendingThread.init, _get_retry_token_ids, _update_retry_token_ids, _handle_request）：核心实现文件：新增 _retry_token_ids 重试状态、_get_retry_token_ids/_update_retry_token_ids 判活逻辑，以及 _handle_request 中基于 request hash 链的 KV event parent 推导，decode PUT 的异步完成与失败恢复均在此落地。
- 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 MooncakeStoreScheduler.init, build_connector_meta）：decode 保存的调度入口：consumer 角色开启 save_decode_cache 后才处理 cached decode 请求的保存元数据；token_ids 快照按 enable_kv_events 裁剪，并落实 decode 未启用时的 fast path。
- vllm/distributed/kv_transfer/kv_connector/v1/mooncake/store/data.py（模块 数据契约；类别 source；类型 data-contract；符号 ReqMeta, ReqMeta.from_request_tracker）：数据契约变更：ReqMeta 新增 token_ids_start，from_request_tracker 只传增量 token 后缀，移除遗留 is_last_chunk 字段，是长请求优化与 KV event 重试的基础。
- vllm/distributed/kv_transfer/kv_connector/v1/mooncake/store/connector.py（模块 连接器；类别 source；类型 configuration；符号 MooncakeStoreConnector.init）：连接器入口：读取 save_decode_cache 并调整 _capacity_only 判定，使消费者打开 Store 写路径并正常通过 KV cache 形状校验。
- tests/v1/kv_connector/unit/test_mooncake_store_scheduler.py（模块 调度器；类别 test；类型 test-coverage；符号 _make_decode_scheduler_output, _make_new_scheduler_output, _setup_decode_request, test_scheduler_only_tracks_token_ids_for_kv_events）：调度器侧回归测试：覆盖默认跳过、首个 decode 保存回填 prompt、仅保存完整块、KV event token 快照语义。

- tests/v1/kv_connector/unit/test_mooncake_store_worker.py（模块 卸载线程；类别 test；类型 test-coverage；符号 test_save_decode_cache_keeps_transfer_path_enabled, _make_event_store_req, test_store_sending_thread_kv_events_use_request_chain_parents, test_store_sending_thread_kv_events_retry_without_covered_tokens）：worker 侧回归测试：覆盖 KV event 重试 / 恢复、parent 链推导、消费者发送线程启动条件。
- docs/features/mooncake_store_connector_usage.md（模块 文档；类别 docs；类型 documentation）：用户文档：明确 save_decode_cache 的双层语义（开启 decode KV 卸载；对 kv_consumer 变为纯 decode 卸载、继续跳过 prefill offload）。

关键符号：MooncakeStoreConnector.init, MooncakeStoreScheduler.init, MooncakeStoreScheduler.build_connector_meta, ReqMeta.from_request_tracker, KVCacheStoreSendingThread.init, KVCacheStoreSendingThread._get_retry_token_ids, KVCacheStoreSendingThread._update_retry_token_ids, KVCacheStoreSendingThread._handle_request, KVCacheStoreSendingThread.delete_finished_stored_request

关键源码片段

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

核心实现文件：新增 _retry_token_ids 重试状态、_get_retry_token_ids/_update_retry_token_ids 判活逻辑，以及 _handle_request 中基于 request hash 链的 KV event parent 推导，decode PUT 的异步完成与失败恢复均在此落地。

```
# vllm/distributed/kv_transfer/kv_connector/v1/mooncake/store/worker.py
# KV event 重试需要 token 后缀来重建 KV；这里只保留失败后的增量状态，
# 并且所有读写都先校验 store job 是否仍然存活。

class KVCacheStoreSendingThread:
    def _get_retry_token_ids(
        self, req_meta: ReqMeta
    ) -> tuple[int, list[int]] | None:
        """Return retry state only if this store job is still live."""
        with self.done_task_lock:
            # store_job_id 在整个引擎生命周期内不重复，而 req_id 会在
            # preemption 恢复后被复用；因此只有"当前存活"的 job 才能
            # 读取重试状态，防止旧代 job 读取复用 req_id 上的脏数据。
            if req_meta.store_job_id not in self.stored_requests.get(
                req_meta.req_id, ()
            ):
                return None
            return self._retry_token_ids.get(req_meta.req_id)

    def _update_retry_token_ids(
        self,
        req_meta: ReqMeta,
        save_completed: bool,
        token_ids_start: int,
```

```
        event_token_ids: list[int] | None,
    ) -> None:
        """Update retry state without letting a stale job touch a reused ID."""
        with self.done_task_lock:
            if req_meta.store_job_id not in self.stored_requests.get(
                req_meta.req_id, ()
            ):
                return
            if save_completed:
                # 本次保存成功（或已决定放弃），清理重试状态。
                self._retry_token_ids.pop(req_meta.req_id, None)
            elif event_token_ids is not None:
                # 只保留失败的 token 后缀与绝对起始偏移；正常路径
                # 不需要为每个请求维护全量 token 快照。
                self._retry_token_ids[req_meta.req_id] = (
                    token_ids_start,
                    event_token_ids,
                )
```

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

decode 保存的调度入口：consumer 角色开启 save_decode_cache 后才处理 cached decode 请求的保存元数据；token_ids 快照按 enable_kv_events 裁剪，并落实 decode 未启用时的 fast path。

```
# vllm/distributed/kv_transfer/kv_connector/v1/mooncake/store/scheduler.py
# decode 保存的调度入口：consumer 只有显式开启 save_decode_cache 后，
# 才会进入 decode 段的保存元数据构造。

def build_connector_meta(
    self, scheduler_output: SchedulerOutput
) -> KVConnectorMetadata:
    """Build connector metadata for this scheduler step."""
    # kv_consumer 默认只读；开启 save_decode_cache 后才处理 decode 段保存。
    is_consumer = self.kv_role == "kv_consumer"
    can_process_cached = not is_consumer or self.save_decode_cache

    # 此处省略 finished / preempted 请求的清理，与改动前行为一致。

    meta = MooncakeStoreConnectorMetadata(
        self._unfinished_request_ids,
        preempted_ids,
    )

    # 新请求（prefill）路径：consumer 始终 skip_save，decode 卸载
    # 不改变“prefill 不上传”的既有语义。
    for request in scheduler_output.scheduled_new_reqs:
        request_tracker = RequestTracker(
            req_id=request.req_id,
            token_len=num_tokens_to_compute,
```

```
                allocated_block_ids=unfolded_block_ids,
                num_saved_tokens=0,
                token_ids=(
                    prefill_tokens[:num_tokens_to_compute]
                    if self.enable_kv_events
                    else None
                ),
                prefill_end_tokens=len(prefill_tokens),
            )
            self._request_trackers[request.req_id] = request_tracker
            req_meta = ReqMeta.from_request_tracker(
                request_tracker,
                self._block_size,
                load_spec=load_spec,
                # consumer 也可以写 decode KV，但 prefill 仍被跳过；
                # load 信息继续由同一份元数据携带。
                skip_save=is_consumer,
                block_hashes=request_real.block_hashes,
            )
            if req_meta is not None:
                meta.add_request(req_meta)

        # decode 路径：只有非 consumer 或显式开启 save_decode_cache 时才构造
        # 保存请求。fresh consumer 的 num_saved_tokens 从 0 起步，首个 decode
        # 保存因此会回填 block 对齐的 prompt 前缀（见 review 中的语义澄清）。
        cached_reqs = scheduler_output.scheduled_cached_reqs
        if can_process_cached:
            for i, req_id in enumerate(cached_reqs.req_ids):
                new_block_ids = cached_reqs.new_block_ids[i]
                # 此处省略 resumed 分支与 per-request tracker 构造，
                # 与 prefill 路径共用相同的 token_ids 条件化逻辑。
                ...
```

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

数据契约变更：ReqMeta 新增 token_ids_start，from_request_tracker 只传增量 token 后缀，移除遗留 is_last_chunk 字段，是长请求优化与 KV event 重试的基础。

```
# vllm/distributed/kv_transfer/kv_connector/v1/mooncake/store/data.py
# 从调度器 tracker 生成 worker 元数据：按 block 边界裁剪本次可保存范围，
# 并让元数据持有独立的增量 token 切片（长请求优化，见 review 讨论）。

@classmethod
def from_request_tracker(
    cls,
    tracker: RequestTracker,
    block_size: int,
    load_spec: LoadSpec | None = None,
    skip_save: bool = False,
    block_hashes: list[BlockHash] | None = None,
```

```
) -> "ReqMeta | None":
    """Create ReqMeta from a RequestTracker."""
    if block_hashes is None:
        block_hashes = []

    input_token_len = tracker.token_len
    # token_ids_start 等于上一次已保存的 token 数：本次保存只关心其后缀，
    # 长请求下避免为每个 store job 拷贝整段 token 序列。
    token_ids_start = tracker.num_saved_tokens
    chunk_boundary = cdiv(token_ids_start + 1, block_size) * block_size
    num_tokens_to_save = input_token_len // block_size * block_size
    skip_save = skip_save or num_tokens_to_save < chunk_boundary

    # 推进 tracker 的已保存进度（此处省略 block id 的相关记账）。
    tracker.num_saved_tokens = num_tokens_to_save

    token_ids = None
    if tracker.token_ids and not skip_save:
        # scheduler 侧的 token_ids 会随 decode 继续增长，元数据必须持有
        # 稳定快照切片，避免异步 worker 在排队期间读到漂移的数据。
        token_ids = tracker.token_ids[token_ids_start:num_tokens_to_save]

    # load_spec 日志与校验在此省略，行为与改动前一致。
    return cls(
        req_id=tracker.req_id,
        can_save=not skip_save,
        load_spec=load_spec,
        block_hashes=block_hashes,
        token_ids=token_ids,
        token_ids_start=token_ids_start,
        num_prompt_tokens=tracker.prefill_end_tokens,
    )
```

评论区精华

review 的核心交锋集中在四类问题：一是语义澄清——ivanium 指出 fresh consumer 首次 decode 保存会处理 prompt 块，与 PR 描述矛盾，chengy-sysu 确认回填 block 对齐 prompt 前缀是预期行为并更新文档；二是正确性——ivanium 发现 decode 路径中 prompt key 被 batch_is_exist() 过滤后，第一个 decode block 会被错误标记为 root block，作者改为从 request hash 链推导 parent；三是性能——ivanium 建议长请求只携带增量 token 后缀、decode 未启用时提前 fast path，作者均落实；四是清理——ivanium 指出 is_last_chunk 是死变量，作者移除该字段。ivanium 还在评论中要求 rebase 到 #52372 并借鉴其 store_job_id 保护机制，本 PR 第 3 个提交即为其在 KV event retry 路径上的延伸。

- 首个 decode 保存回填 prompt 前缀的语义澄清 (correctness): chengy-sysu 确认回填 block 对齐 prompt 前缀是预期行为：P-D KV 通过 MooncakeConnector 直传时 prompt KV 可能不在 Store，D 需在首次保存时补齐；Store 存在性检查会去重，已更新文档。

- KV event parent 链推导 (correctness): chengy-sysu 修复：每个 KV event 的 parent_block_hash 直接从原始 request hash 链推导，覆盖 Store 去重、稀疏 miss 与 TP-strided PUT，并新增参数化测试。
- 长请求 token 快照的增量优化 (performance): z-zanez 分两部分实现：KV events 关闭时调度器完全不维护 token_ids；开启时 ReqMeta 只带 [token_ids_start, num_tokens_to_save) 切片与绝对偏移。
- store_job_id 保护 KV event 重试 (correctness): 实现 _get_retry_token_ids/_update_retry_token_ids 均先校验 store_job_id 是否仍存活，防止 stale job 触碰复用 req_id 的状态。
- decode 未启用时的 fast path (performance): z-zanez 落实：build_connector_meta 在 decode 未启用分支提前返回，测试验证 tracker 不再推进。
- is_last_chunk 死变量清理 (design): 作者从 ReqMeta 与 from_request_tracker 中删除该字段及相关计算，未来实现精确非 block 对齐 offload 时再加回。
- save_decode_cache 文档语义 (documentation): 文档更新为双层语义说明，并补充 fresh consumer 首次保存回填 prompt 前缀的行为。

风险与影响

- 风险：
  1. 调度器控制流重构影响面：build_connector_meta 的 force_skip_save 语义被拆分，kv_both/kv_producer/kv_consumer 角色均经过该路径，prefill 路径 token_ids 迁移到 enable_kv_events 条件后，配置不一致可能导致 KV event 数据缺失。
  2. KV event parent 依赖 request hash 链：若 req_meta.block_hashes 不完整或 TP-strided PUT 下各 rank 的 hash 链不一致，会生成错误的前缀关系。
  3. 异步状态并发：_retry_token_ids、_saved_offset、stored_requests 跨发送线程与调度器共享，依赖 done_task_lock；新增成员虽有清理路径，但异常分支回归风险仍存在。
  4. 内存与带宽：开启 enable_kv_events 时 tracker 每步 extend token_ids，长请求在调度器侧仍持有全量 token；Store 写带宽与容量占用随 decode 卸载增加，首次保存可能一次性回填整段 prompt 产生突发 PUT。
  5. 测试覆盖：仓库内为单测与 pre-commit，RDMA e2e 脚本在 fork，跨机场景无法由 CI 自动回归。 - 影响：用户侧：PD 分离与非 PD 多实例共享 Mooncake Store 的场景中，decode KV 可跨实例复用，长对话续接的缓存命中率提升。系统侧：Store 写带宽与 token 数据库容量增加，首次 decode 保存可能产生突发 PUT 流量。团队侧：完成 route map #45036 的关键里程碑，为后续 hetero TP support（ivanium approve 时明确期待）铺路；store_job_id 账本与增量 token 快照成为 KV connector 组件可复用的防御模式。默认关闭的配置设计使既有用户无感。 - 风险标记：新增配置项默认关闭，默认路径无回归，scheduler 控制流重构影响所有 connector 角色，KV event parent 依赖 request hash 链完整性，重试状态跨异步线程共享，依赖 done_task_lock，RDMA 集群 e2e 未纳入仓库 CI

关联脉络

- PR #52372 （标题未在提供材料中给出；ivanium 在本 PR 评论中提及）：ivanium 要求本 PR rebase 到 #52372，并建议对 _retry_token_ids 采用与 #52372 相同的 store_job_id 保护以正确处理 preemption；本 PR 第 3 个提交即落实该保护。
- PR #50809 [Bugfix][V1] Sync mamba_block_size via EngineCoreReadyResponse: 同属 v1 引擎与 KV 连接器之间的元数据契约调整（EngineCoreReadyResponse 与 ReqMeta），说明 v1 KV connector 元数据同步是近期持续演进的主题。
- PR #52281 [ROCm] Give EngineCore cleanup grace after request abort: 同属 v1 引擎请求生命周期与异步清理逻辑，与本 PR 的异步 store job 生命周期、block 引用释放关注点一致。