

PR #53598 完整报告

vllm-project/vllm

[ROCm][DSpark][DCP] Serve prefix cache hits under DCP for Kimi-K3

合并时间: 2026-09-01 02:18

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53598>

执行摘要

- 一句话: 修复 DCP 下混合 KV 组缓存几何不一致, 恢复 Kimi-K3 前缀命中
- 推荐动作: 值得精读。dcp_world_size_for_kv_cache_spec 的 spec 分类与 UniformTypeKVCacheSpecs 解包处理, 是理解 vLLM 混合 KV cache 组与并行策略交互的很好的切入点; 协调器把 _cache_hit_alignment_tokens 传播到 manager 保留掩码的设计, 体现了 "查找粒度与保留粒度必须一致" 的约束; review 中 GirasoleY 对正交变更拆分、最小复现与边界数学的要求, 则是大型基础设施 PR 评审的规范示范。关注 DCP/MLA/投机解码组合的工程师, 应同时阅读 #51705 与后续 replay-boundary PR, 才能评估端到端命中率收益。

功能与动机

PR body 明确指出问题本质: Kimi-K3 混合了 DCP-sharded 的 MLA/ 全注意力组与 replicated 的 Mamba 组, 而进程级 DCP 大小被应用到所有组, 导致 scheduler、block hashing 与 cache managers 对 block 几何理解不一致 ("Applying the process-wide DCP size to every group makes the scheduler, block hashing, and cache managers disagree about block geometry")。PR 同时明确限定范围: 不含 DSpark/DCP 运行时注意力 (#51705)、EAGLE/Mamba 重放边界保留 (与 #51295/#50897 重叠)、以及 offloader 几何与 failed KV-load 恢复 (#53917), 目的是让 per-group DCP 缓存几何这一层修复独立合入。

实现拆解

该 PR 围绕 "每个 KV cache group 只在自己的有效 DCP 几何下工作" 这一原则, 分 5 步实现:

1. 几何分类入口 (vllm/v1/core/kv_cache_utils.py): 新增 `dcp_world_size_for_kv_cache_spec(spec, dcp_world_size)`。进程 DCP 大小不大于 1 时直接返回 1; 对 `UniformTypeKVCacheSpecs` 包装的复合组取第一个内层 spec 作为代表; 只有 `FullAttentionSpec` (含 MLA 路径) 返回进程 DCP 大小, Mamba、滑窗、chunked-local 等复制状态组一律返回 1。这样把 "KV 在 rank 间分片则 block 放大" 的几何语义收敛到单一函数, 后续 spec 类型扩展只需改这一处。
2. manager 构造贯通 (vllm/v1/core/kv_cache_coordinator.py): `KVCacheCoordinator.__init__` 构造每个 `SingleTypeKVCacheManager` 时, 把原先统一的 `dcp_world_size=dcp_world_size` 换成 `dcp_world_size_for_kv_cache_spec(kv_cache_gro`

up.kv_cache_spec, dcp_world_size)。同时 UnitaryKVCacheCoordinator.__init__ 不再自行 block_size *= dcp，改为从 single_type_managers[0] 读取已解析后的 block_size 与 dcp_world_size，避免单组路径与 per-group 路径双轨漂移。

3. 查找阶段使用 manager 自有几何 (kv_cache_coordinator.py) :
HybridKVCacheCoordinator.find_longest_cache_hit 与 find_longest_cache_hit_per_group 在调用 find_longest_cache_hit 时，dcp_world_size 与 pcp_world_size 改为从 single_type_managers[first_group_id] 读取，替代原先按 FullAttentionSpec 推断 DCP 大小、否则为 1 的写法，消除了 spec 类型判断与 manager 实际几何不一致的隐患。
4. 保留对齐传播 (single_type_kv_cache_manager.py + kv_cache_coordinator.py) :
SingleTypeKVCacheManager 新增 cache_hit_alignment_tokens 属性，默认等于 scheduler_block_size (粗对齐)；协调器在完成 enable_partial_hash_hits 校验后，把 _cache_hit_alignment_tokens (细粒度开启时为 hash_block_size) 统一写入所有 manager。cache_blocks 中的 reachable_block_mask 改用它，保证非 EAGLE 的 DCP 细粒度命中场景下，稀疏保留的 Mamba 检查点不会被较大的 scheduler LCM 提前丢弃。
5. 测试配套：三个测试文件新增针对性用例——tests/v1/core/test_prefix_caching.py 的 test_prefix_cache_hit_uses_per_group_dcp_geometry 构造 target MLA / draft MLA / Mamba 三组、DCP=8 配置，验证 manager 的 dcp_world_size 与 block_size 分配及二次请求命中块数 [2, 2, 16]；tests/v1/core/prefix_cache/test_partial_prefix_cache_hits.py 的 test_dcp_fine_hit_retention_uses_hash_alignment_without_eagle 覆盖 DCP=4 + retention_interval=0 下 Mamba 状态按 hash 对齐保留并可复用 6 个 token；tests/v1/core/test_kv_cache_utils.py 的 test_dcp_world_size_for_kv_cache_spec_shard_s_full_attention_only 单测分类函数本身。

关键文件：

- vllm/v1/core/kv_cache_utils.py (模块 缓存工具；类别 source；类型 core-logic；符号 dcp_world_size_for_kv_cache_spec)：新增核心辅助函数 dcp_world_size_for_kv_cache_spec，按 spec 类型决定每个 KV cache group 的有效 DCP 大小，是整个修复的几何分类入口。
- vllm/v1/core/kv_cache_coordinator.py (模块 缓存协调器；类别 source；类型 core-logic；符号 KVCacheCoordinator.init, HybridKVCacheCoordinator.find_longest_cache_hit, find_longest_cache_hit_per_group, UnitaryKVCacheCoordinator.init)：per-group 几何在 manager 构造、前缀查找与保留对齐三层贯通的枢纽文件，改动最大且直接影响混合缓存命中路径。
- vllm/v1/core/single_type_kv_cache_manager.py (模块 缓存管理器；类别 source；类型 core-logic；符号 SingleTypeKVCacheManager.cache_blocks)：新增 cache_hit_alignment_tokens 并把 cache_blocks 的保留掩码从固定 scheduler_block_size 切换到该属性，是保留对齐修复的关键载体。
- tests/v1/core/test_prefix_caching.py (模块 前缀缓存；类别 test；类型 test-coverage；符号 test_prefix_cache_hit_uses_per_group_dcp_geometry)：新增 test_prefix_cache_hit_uses_per_group_dcp_geometry，验证 target MLA / draft MLA / Mamba 三组在 DCP=8 下的几何分配与命中块数，是本 PR 核心行为的回归测试。

- tests/v1/core/prefix_cache/test_partial_prefix_cache_hits.py (模块 细粒度命中; 类别 test; 类型 test-coverage; 符号 test_dcp_fine_hit_retention_uses_hash_alignment_with_out_eagle) : 新增 test_dcp_fine_hit_retention_uses_hash_alignment_without_eagle, 覆盖 DCP=4 + retention_interval=0 下 Mamba 状态按 hash 对齐保留、可被后续请求复用, 验证保留对齐传播。
- tests/v1/core/test_kv_cache_utils.py (模块 缓存工具; 类别 test; 类型 test-coverage; 符号 test_dcp_world_size_for_kv_cache_spec_shards_full_attention_only) : 新增 test_dcp_world_size_for_kv_cache_spec_shards_full_attention_only, 直接单测几何分类函数在各 spec 类型上的输出。

关键符号: dcp_world_size_for_kv_cache_spec, KVCacheCoordinator.init, HybridKVCacheCoordinator.find_longest_cache_hit, find_longest_cache_hit_per_group, UnitaryKVCacheCoordinator.init, SingleTypeKVCacheManager.cache_blocks

关键源码片段

vllm/v1/core/kv_cache_utils.py

新增核心辅助函数 `dcp_world_size_for_kv_cache_spec`, 按 spec 类型决定每个 KV cache group 的有效 DCP 大小, 是整个修复的几何分类入口。

```
def dcp_world_size_for_kv_cache_spec(spec: KVCacheSpec, dcp_world_size: int) -> int:
    """返回真正拥有该组 block 几何的 DCP 大小。
```

```
    全注意力 KV (含 MLA) 在 DCP rank 间分片, 因此前缀哈希与 cache manager
    的 block_size 使用进程级 DCP 大小; Mamba、滑窗、chunked-local 等保持
    per-rank 复制状态, 即使进程以 DCP > 1 运行也必须固定为 1。
```

```
    """
```

```
    if dcp_world_size <= 1:
        # 单 rank 没有分片语义, 直接短路返回。
        return 1
```

```
    inner = spec
```

```
    # UniformTypeKVCacheSpecs 包装多个同构 layer spec, 取第一个内层 spec
    # 即可代表该组的几何类型。
```

```
    if isinstance(spec, UniformTypeKVCacheSpecs):
        inner = next(iter(spec.kv_cache_specs.values()))
```

```
    if isinstance(inner, FullAttentionSpec):
        # 目标 MLA 与 DSpark draft MLA 都走 FullAttentionSpec / MLAAttentionSpec
        # 路径, 因此分片组保持进程 DCP 大小; 若将来出现复制的 draft 组,
        # 需要新的 spec 类型, 而不是靠本函数特判。
        return dcp_world_size
```

```
    return 1
```

vllm/v1/core/kv_cache_coordinator.py

per-group 几何在 manager 构造、前缀查找与保留对齐三层贯通的枢纽文件, 改动最大且直接影响混合缓存命中路径。

- # KVCacheCoordinator.__init__: 每个 KV cache group 构造独立的
- # SingleTypeKVCacheManager, DCP 大小按 spec 类型逐组解析, 而不是

```

# 所有组共用进程级 dcp_world_size。
self.single_type_managers = tuple(
    get_manager_for_kv_cache_spec(
        kv_cache_spec=kv_cache_group.kv_cache_spec,
        max_in_flight_tokens=max_in_flight_tokens,
        max_model_len=max_model_len,
        block_pool=self.block_pool,
        enable_caching=enable_caching,
        kv_cache_group_id=i,
        # 关键变化：每个组只在自己的有效 DCP 几何下工作，全注意力 /MLA
        # 组被分片放大 block；Mamba 复制组保持 dcp_world_size=1。
        dcp_world_size=dcp_world_size_for_kv_cache_spec(
            kv_cache_group.kv_cache_spec, dcp_world_size
        ),
        pcp_world_size=pcp_world_size,
        scheduler_block_size=self.scheduler_block_size,
        needs_kv_cache_zeroing=self.kv_cache_config.needs_kv_cache_zeroing,
    )
    for i, kv_cache_group in enumerate(self.kv_cache_config.kv_cache_groups)
)

# HybridKVCacheCoordinator.find_longest_cache_hit 内部：查找阶段不再按
# spec 类型推断 DCP 大小，而是直接读取该组 manager 已解析并验证过的
# 几何参数，保证 target MLA、draft MLA 与 Mamba 各自按正确粒度对齐。
hit_blocks, _new_hit_length = manager_cls.find_longest_cache_hit(
    block_hashes=block_hashes,
    max_length=_max_length,
    kv_cache_group_ids=group_ids,
    block_pool=self.block_pool,
    kv_cache_spec=spec,
    drop_eagle_block=drop_eagle_block,
    alignment_tokens=self._cache_hit_alignment_tokens,
    dcp_world_size=self.single_type_managers[first_group_id].dcp_world_size,
    pcp_world_size=self.single_type_managers[first_group_id].pcp_world_size,
)

```

vllm/v1/core/single_type_kv_cache_manager.py

新增 `cache_hit_alignment_tokens` 并把 `cache_blocks` 的保留掩码从固定 `scheduler_block_size` 切换到该属性，是保留对齐修复的关键载体。

```

class SingleTypeKVCacheManager:
    def __init__(self, ...):
        ...
        self.scheduler_block_size = scheduler_block_size
        # 混合细粒度查找通过协调器验证后，可能把命中对齐降到 hash 块
        # 粒度；协调器会统一写入该属性。默认保持 scheduler 对齐（粗），
        # 保证非混合或细粒度被禁用的场景行为不变。
        self.cache_hit_alignment_tokens = scheduler_block_size
        ...

```

```
def cache_blocks(self, request, num_tokens, retention_interval=None):
    ...
    block_mask = self.reachable_block_mask(
        start_block=num_cached_blocks,
        end_block=num_full_blocks,
        # 保留掩码使用缓存命中对齐而非固定 scheduler 块大小:
        # DCP 下细粒度命中按 hash 块对齐时, 较大的 scheduler LCM
        # 会过早丢弃可复用的 Mamba 检查点, 导致后续请求重新计算。
        alignment_tokens=self.cache_hit_alignment_tokens,
        kv_cache_spec=self.kv_cache_spec,
        use_eagle=self.use_eagle,
        retention_interval=retention_interval,
        reachable_boundaries=reachable_boundaries,
    )
    self.block_pool.cache_full_blocks(...)
```

评论区精华

review 的核心交锋集中在范围控制与边界正确性上:

- 拆分正交变更 (design) : GirasoleY 在 CHANGES_REQUESTED 中指出 PR 同时处理了 per-group DCP geometry、Mamba replay boundary、SimpleCPU offload、CoW、cache load failure 等多个问题, 要求保持聚焦。作者回应: offload 与 failed-load 处理移到 #53917; CoW refresh 保留, 因为 allocate_new_blocks() 会消费 _partial_hit_reqs 并调用 _apply_cow(), 没有 refresh 时 stale 元数据会指向请求已不拥有的块。
- failed-load 最小复现 (question) : GirasoleY 要求对 scheduler.py 中 failed blocks 处理给出实际运行失败或最小复现; 作者指出单测 test_hybrid_load_failure_uses_runtime_manager_block_sizes 即为最小复现, 且旧代码中 # TODO (davidb): add support for hybrid memory allocator 已标注该路径未支持混合组。
- CoW 断言正确性 (correctness) : GirasoleY 认为 _partial_hit_reqs stale 在当前控制流不应发生, 静默替换 source 或跳过 CoW 只会掩盖问题并请 @ivanium 复核; 作者同意, 说明更新几何后该分支本会触发 _apply_cow 安全断言, 删除是正确做法。
- 细粒度命中误启用 (correctness) : GirasoleY 指出上一版 has_dcp_partial_full_attention_group 会在纯 full-attention DCP 场景启用 partial prefix hit, 与 " 仅 --mamba-cache-mode align 才启用 " 的初衷不符; 作者修复, 最终版本未包含该扩展。
- replay boundary 归属 (design) : GirasoleY 指出 Mamba replay boundary 变更与 #51295 重叠; 作者决定本 PR 保持 per-group DCP geometry 范围, 边界问题独立推进, 并给出实测对比: 无 replay-boundary retention 时命中率仅 15.60% (理论 80.37%), 有则升至 71.90%。
 - 拆分正交变更以保持 PR 聚焦 (design): offload 与 load 失败处理拆出到 #53917, CoW refresh 作为同一 partial-hit 正确性路径的一部分保留。
 - failed-load 处理需要最小复现 (question): 该变更移出到 #53917 独立处理。
 - CoW stale 状态是否可能发生 (correctness): 采纳移除 CoW refresh 逻辑, 避免掩盖底层不一致。

- 纯全注意力 DCP 场景误启用细粒度命中 (correctness): 作者修复, 最终版本未包含该扩展逻辑, 细粒度命中仍由 Mamba align 组触发。
- Mamba replay boundary 与 #51295 重叠 (design): 本 PR 不做 replay-boundary 保留, 实测命中率差距留待后续独立 PR 解决。

风险与影响

• 风险:

1. 核心路径回归风险: KVCacheCoordinator.__init__、find_longest_cache_hit、find_longest_cache_hit_per_group 是 v1 前缀缓存核心路径, 改动会影响所有混合 KV group 配置 (含非 DCP 场景, 此时 dcp_world_size_for_kv_cache_spec 返回 1 走原路径)。dcp_world_size_for_kv_cache_spec 的分类若漏掉未来新增的 spec 类型 (如复制型 draft 组), 会静默放大 block 几何并产生错误缓存命中而非报错, 这是该类逻辑最危险的行为模式。
2. 保留对齐传播的耦合: cache_hit_alignment_tokens 由协调器在 enable_partial_hash_hits 校验后统一写入所有 manager; 若某条路径 (如 CPU offload 协调器) 关闭细粒度查找却没有同步回写默认值, 稀疏保留的 Mamba 状态可能被更粗对齐丢弃。当前实现将默认值保持在 scheduler 对齐, 降低了风险, 但该耦合点值得持续关注。
3. 依赖合入顺序: 本 PR 单独合入时端到端效果受限——DSpark/DCP 运行时注意力依赖 #51705, replay-boundary 保留依赖后续独立 PR; PR body 实测显示无 replay-boundary retention 时实际命中率仅 15.60% (理论 80.37%), 用户只合入本 PR 可能误判修复无效。
4. 测试覆盖边界: 新增测试覆盖 DCP4/DCP8 的几何与保留对齐, 但未覆盖 mamba_cache_mode 非 align、非 DCP 混合组等回退路径, 这些路径依赖旧默认值的行为一致性。- 影响: 用户影响: DCP 大于 1 且混合全注意力 / MLA + Mamba 的部署 (如 Kimi-K3 DSpark + 前缀缓存) 是直接受益者, 前缀命中从几何破坏状态恢复; 单类型组或 DCP=1 场景走原默认路径, 行为不变。系统影响: per-group 几何贯通 manager 构造、查找与保留三层, 消除了 block 哈希与几何不一致导致的缓存 miss 与潜在错误命中, 为后续 offloader 几何与 replay-boundary 修复提供了统一基础。团队影响: 该 PR 与 #51705、#53821 等构成 ROCm DCP 投机解码功能主线, review 中已厘清范围拆分与合入顺序, 后续 PR 可在此基础上继续推进。- 风险标记: 核心缓存路径变更, 依赖 #51705 运行时支持, Mamba 重放边界问题未解决, DCP 混合几何静默错误风险

关联脉络

- PR #51705 [ROCm][MLA][DCP] Support causal multi-token verification: PR body 明确声明 DSpark/DCP 运行时注意力支持由 #51705 处理, 本 PR 的端到端验证栈依赖它; 两者共同构成 Kimi-K3 DSpark DCP 前缀缓存可用的前提。
- PR #53821 [Bugfix][ROCm] Preserve AITER unified-attention metadata during graph replay: 同一 ROCm 投机解码 / 注意力功能线的兄弟修复, 同样作用于 DCP 下的 ROCm 注意力栈 (AITER MLA 元数据保留), 与本 PR 的缓存几何修复配套。