

PR #54736 完整报告

vllm-project/vllm

[Feature][SimpleCPU] Load fine-grained hybrid prefix hits

合并时间: 2026-09-11 09:36

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/54736>

执行摘要

- 一句话: SimpleCPU 支持 hybrid 细粒度外部前缀命中加载
- 推荐动作: 建议精读, 尤其是 SimpleCPUOffloadScheduler.update_state_after_alloc 的按组取整映射与 boundary handoff 消费逻辑, 它们体现了「同一批 token 在不同 block size 的 group 间如何对齐」这类典型 hybrid 难题。重点关注的三个设计决策: 一是用 has_positionally_stable_blocks 把「位置游标是否可用」抽象成 manager 级别的可覆写属性, 而非散落的 if mamba_mode == 'align' 判断; 二是几何尺寸统一到 coordinator 的 group_block_sizes, 删掉了重复的 spec.block_size * cp_world_size 手算式; 三是把「静默丢失的 handoff」用 BoundaryStoreStats 计数器显式化。若团队维护 hybrid 模型或自研 external prefix offload, 应把该 PR 作为块对齐规则的参考实现。

功能与动机

PR body 明确指出 hybrid 模型上 SimpleCPU offload 无法可靠服务细粒度外部前缀命中, 并给出两条根因: (1) Mamba 的 align 模式「nulls interior states and relocates speculative blocks in place, so its block table cannot be resolved by a monotonically advancing positional cursor. The connector must consume the exact boundary handoffs already published by the KV cache manager.」; (2) 「A prompt can end on a hash boundary that is not a full-attention effective-block boundary. The positional scan only preserves whole attention blocks, so the recurrent group and full-attention group otherwise do not hold the same boundary and the hybrid lookup reconciles to zero.」此外还发现旧的 `spec.block_size * cp_world_size` 几何估算会把复制型 group 按 DCP 放大, 导致 eager store 只写入一部分块。PR 以 #54735 为栈基。

实现拆解

1. 新增「块表位置稳定性」抽象: 在 `vllm/v1/core/single_type_kv_cache_manager.py` 为 `SingleTypeKVCacheManager` 增加 `has_positionally_stable_blocks` 属性 (默认 `True`), 并由 `MambaManager` 覆写为 `mamba_cache_mode != "align"`。原因见类内注释——align 模式会置空内部状态并原地搬迁 speculative block, 单调前进的位置游标无法解析其块表。影响是 eager store 的位置扫描范围收窄, `tests/v1/simple_kv_offload/test_kv_events.py` 中原来的 `test_mamba_dcp_metadata_guard` 被 `test_mamba_align_skips_positional_event_metadata` 取代。

2. 统一 group 几何口径: `vllm/v1/core/kv_cache_coordinator.py` 新增 `self.group_block_sizes = tuple(manager.block_size ...)`, 并新增 `allow_partial_hash_hits` 参数经 `get_kv_cache_coordinator` 透传到 `HybridKVCacheCoordinator` (`enable_partial_hash_hits = allow_partial_hash_hits and has_partial_mamba_group`), 同时把原局部变量 `group_block_sizes` 改名为 `cacheable_block_sizes` 以免混淆。`manager.py` 随之删除 `spec.block_size * cp_world_size` 手算式, 改用 `self.group_block_sizes[g]`, `_estimate_lazy_target_blocks` 也改用 `dcp_world_size_for_kv_cache_spec(spec, cp_world_size)`, 避免把 mamba / sliding window / chunked-local 等复制型 group 按 DCP 放大。
3. 加载路径支持细粒度外部命中: `SimpleCPUOffloadScheduler.get_num_new_matched_tokens` 在本地前缀非 scheduler-block 对齐时 `warning_once` 并放弃外部查找; `_pending_cpu_hits` 由 `(blocks, hit_length)` 扩为 `(blocks, hit_length, num_computed_tokens)`, 使 `update_state_after_alloc` 不必从 block metadata 反推本地前缀; 每组取块数由整除改为 `cdiv(num_external_tokens, g_block_size)`, `destination` 起点改为 `num_computed_tokens // g_block_size`, 并新增 `num_external_tokens % hash_block_size == 0` 断言。
4. 结束时的 full-attention 部分尾块: 新增 `_find_fa_partial_tail_source` 并在 `_queue_finished_eager_store` 中补存该 FA 块, 解决 prompt 结束在 hash 边界而 Mamba align 已按细边界 handoff、位置扫描只保留整块导致 hybrid lookup 归零的问题; 该路径是 SimpleCPU 本地查找, 不写入共享的 `boundary_state_offloads` 通道。
5. 消费显式 boundary handoff 并提供可观测性: `_prepare_eager_store_specs` 读取 `scheduler_output.kv_connector_block_state.boundary_state_offloads`, 按 `store_state.finished / preempted_req_ids / finished_req_ids` 过滤失效请求, 用 `_StoreAdmission` 枚举分类每个候选块, 并用新增的 `BoundaryStoreStats` 记录 `published/stored/dropped/skipped` 计数, 经 `SimpleCPUOffloadConnector.get_boundary_store_stats()` 暴露。
6. 测试与配套: `tests/v1/simple_kv_offload/test_scheduler.py` 新增 `_make_hybrid_attention_mamba_scheduler` 夹具与 `test_hybrid_store_uses_resolved_group_block_sizes`、`test_dcp_mixed_cache_loads_fine_grained_external_hit`、`test_finished_eager_store_caches_fa_partial_tail_for_hybrid_hit`、`test_finished_eager_store_does_not_duplicate_completed_partial_tail`、`test_external_lookup_rejects_misaligned_local_prefix`、`test_fine_grained_external_hits_require_eager_offload`、`test_eager_store_does_not_scan_mamba_blocks_positionally` 等用例; `lazy` 模式通过 `allow_partial_hash_hits=not lazy_offload` 保持 scheduler-block 对齐。

关键文件:

- `vllm/v1/simple_kv_offload/manager.py` (模块 卸载调度; 类别 source; 类型 core-logic; 符号 `_StoreAdmission`, `BoundaryStoreStats`, `_classify_store_candidate`, `get_boundary_store_stats`) : SimpleCPU offload 调度侧核心: 删除手算 group 几何、改为按 coordinator 解析的 `group_block_sizes` 映射外部 token 范围, 新增 `_StoreAdmission/BoundaryStoreStats/_find_fa_partial_tail_source` 并消费 boundary

handoff, 是本次功能与两个 review 修复的落点。

- tests/v1/simple_kv_offload/test_scheduler.py (模块 卸载测试; 类别 test; 类型 test-coverage; 符号 `_make_hybrid_attention_mamba_scheduler`, `test_hybrid_store_uses_resolved_group_block_sizes`, `test_dcp_mixed_cache_loads_fine_grained_external_hit`, `test_finished_eager_store_caches_fa_partial_tail_for_hybrid_hit`) : 新增 672 行单元测试: `_make_hybrid_attention_mamba_scheduler` 夹具构造 attention + Mamba 双 group 调度器, 覆盖组几何、细粒度外部命中映射、FA 部分尾块存储与去重、lazy 门禁等, 是本次改动可验证性的主要依据。
- vllm/v1/core/single_type_kv_cache_manager.py (模块 缓存管理; 类别 source; 类型 core-logic; 符号 `has_positionally_stable_blocks`) : 新增 `has_positionally_stable_blocks` 属性, 把「位置游标能否跟随该块表」抽象为 manager 级契约, 并由 MambaManager 在 align 模式返回 False, 是连接 offload 扫描与 Mamba 状态语义的关键接口。
- vllm/v1/core/kv_cache_coordinator.py (模块 缓存协调; 类别 source; 类型 core-logic; 符号 `HybridKVCacheCoordinator.init`, `_cache_hit_alignment_tokens`, `get_kv_cache_coordinator`) : `coordinator` 新增 `group_block_sizes` 公共属性与 `allow_partial_hash_hits` 参数, 前者成为 SimpleCPU 几何计算的唯一来源, 后者让 lazy offload 关闭部分 hash 命中并保持 scheduler-block 对齐; 同时修正了 `cacheable_block_sizes` 的命名歧义。
- tests/v1/simple_kv_offload/test_kv_events.py (模块 事件测试; 类别 test; 类型 test-coverage; 符号 `test_mamba_dcp_metadata_guard`, `test_mamba_align_skips_positional_event_metadata`) : 用 `test_mamba_align_skips_positional_event_metadata` 取代原来的 `test_mamba_dcp_metadata_guard`, 断言 Mamba align 组不再从位置扫描产生事件、只由显式 handoff 触发, 验证了 `has_positionally_stable_blocks` 带来的行为变化。
- vllm/distributed/kv_transfer/kv_connector/v1/simple_cpu_offload_connector.py (模块 连接器; 类别 source; 类型 core-logic; 符号 `get_boundary_store_stats`) : 连接器新增 `get_boundary_store_stats()` 透传接口, 把调度侧的 BoundaryStoreStats 暴露给调用方, 是这次「静默丢边界」问题的唯一可观测出口。

关键符号: `SimpleCPUOffloadScheduler.get_num_new_matched_tokens`,
`SimpleCPUOffloadScheduler.update_state_after_alloc`,
`SimpleCPUOffloadScheduler._prepare_eager_store_specs`,
`SimpleCPUOffloadScheduler._queue_finished_eager_store`,
`SimpleCPUOffloadScheduler._find_fa_partial_tail_source`,
`SimpleCPUOffloadScheduler._classify_store_candidate`,
`SimpleCPUOffloadScheduler.get_boundary_store_stats`,
`SingleTypeKVCacheManager.has_positionally_stable_blocks`,
`MambaManager.has_positionally_stable_blocks`,
`HybridKVCacheCoordinator._cache_hit_alignment_tokens`,
`SimpleCPUOffloadConnector.get_boundary_store_stats`

关键源码片段

vllm/v1/simple_kv_offload/manager.py

SimpleCPU offload 调度侧核心：删除手算 group 几何、改为按 coordinator 解析的 `group_block_sizes` 映射外部 token 范围，新增 `_StoreAdmission/BoundaryStoreStats/_find_fa_partial_tail_source` 并消费 boundary handoff，是本次功能与两个 review 修复的落点。

外部命中映射 (`update_state_after_alloc`，已按 PR 语义重新整理)：

```
def update_state_after_alloc(self, request, blocks, num_external_tokens: int) -> None:
    pending = self._pending_cpu_hits.pop(request.request_id, None)
    if pending is None:
        return
    # pending 由二元组扩为三元组：num_computed_tokens 是命中解析时的本地前缀，
    # 加载路径必须用它给外部块定位，不能再从 block metadata 反推。
    cpu_hit_blocks_full, _, num_computed_tokens = pending

    # 传输依旧是整块拷贝，所以本地前缀必须落在 scheduler block 边界上。
    assert num_computed_tokens % self.block_size == 0, (
        "SimpleCPU external loads must start at a scheduler-block boundary"
    )
    # 细粒度命中只保证 hash block 对齐，因此每组块数要向上取整；
    # hash 对齐本身仍必须成立，它保证外部后缀的起点对每个 group 都可寻址。
    assert num_external_tokens % self.hash_block_size == 0, (
        f"num_external_tokens={num_external_tokens} is not aligned to "
        f"hash_block_size={self.hash_block_size}"
    )

    cpu_hit_blocks: list[list[KVCacheBlock]] = []
    for g in range(num_groups):
        # 用 coordinator 解析出的物理块大小取块，取代旧的
        # spec.block_size * cp_world_size 手算口径（后者会把复制型 group 也按 DCP 放大）。
        g_block_size = self.group_block_sizes[g]
        n_take_g = cdiv(num_external_tokens, g_block_size)
        cpu_hit_blocks.append(cpu_hit_blocks_full[g][:n_take_g])

    gpu_block_ids: list[int] = []
    for g in range(num_groups):
        # ... (省略与本段无关的分支)
        if n_ext_g == 0:
            continue
        g_block_size = self.group_block_sizes[g]
        # destination 起点直接由本地前缀 token 数除以本组块大小得到，
        # 不再用 total_computed_tokens 反推：反推只在各组块大小相同时才成立。
        gpu_ext_start = num_computed_tokens // g_block_size
        group_gpu_ids = block_ids_by_group[g]
        for i, cpu_blk in enumerate(cpu_blocks_g):
            # ... (外部块逐个挂到 destination 位置上)
            pass
```

显式 boundary handoff 的消费与准入分类 (`_prepare_eager_store_specs`, 按 diff 与 review 结论整理) :

```
# 显式 boundary handoff: 调度器只负责下发, 不按请求存活性过滤,
# 因此这里必须自己判断请求是否还在, 否则会把已释放、甚至已被重新分配给
# 其它请求的 block 以合法 hash 发布出去, 所以在不确定时保守丢弃。
block_state = scheduler_output.kv_connector_block_state
boundary_offloads = (
    block_state.boundary_state_offloads if block_state is not None else {}
)
stats = self.boundary_store_stats
preempted_req_ids = scheduler_output.preempted_req_ids or ()
for req_id, entries in boundary_offloads.items():
    store_state = self._reqs_to_store.get(req_id)
    if (
        store_state is None
        or store_state.finished
        or req_id in preempted_req_ids
        or req_id in scheduler_output.finished_req_ids
    ):
        stats.published += len(entries)
        stats.dropped_request_gone += len(entries)
        continue
    for _group_idx, gpu_block_id, _num_tokens in entries:
        stats.published += 1
        # 每个候选块给出明确准入结论, decline 原因全部落到计数器上,
        # 避免「命中率莫名变低但没有日志」这种静默故障。
        admission = self._classify_store_candidate(gpu_block_id)
        if admission is _StoreAdmission.READY:
            cpu_block = cpu_block_pool.get_new_blocks(1)[0]
            merged_gpu_block_ids.append(gpu_block_id)
            merged_cpu_block_ids.append(cpu_block.block_id)
            # block_meta 必须与两个 id 列表按下标一一对应; handoff 的 hash
            # 在完成时回放, 所以这里补一个空 dict 占位, 否则
            # _process_store_completion 的长度断言会失败并导致后续 metadata 错位。
            if merged_block_meta is not None:
                merged_block_meta.append({})
            in_flight.add(gpu_block_id)
            stats.stored += 1
        elif admission is _StoreAdmission.ALREADY_CACHED:
            stats.skipped_already_cached += 1
        elif admission is _StoreAdmission.IN_FLIGHT:
            stats.skipped_in_flight += 1
        elif admission is _StoreAdmission.NULL_BLOCK:
            stats.dropped_null_block += 1
        elif admission is _StoreAdmission.NOT_HASHED:
            stats.dropped_not_hashed += 1
```

`tests/v1/simple_kv_offload/test_scheduler.py`

新增 672 行单元测试：_make_hybrid_attention_mamba_scheduler 夹具构造 attention + Mamba 双 group 调度器，覆盖组几何、细粒度外部命中映射、FA 部分尾块存储与去重、lazy 门禁等，是本次改动可验证性的主要依据。

新增夹具与几何断言（节选，已重新整理）：

```
def _make_hybrid_attention_mamba_scheduler(
    *,
    num_cpu_blocks: int = 8,
    num_gpu_blocks: int = 16,
    attention_block_size: int = BLOCK_SIZE,
    block_size: int = 4 * BLOCK_SIZE,
    scheduler_block_size: int | None = None,
    hash_block_size: int | None = None,
    dcp_world_size: int = 4,
    lazy: bool = False,
    mamba_cache_mode: str = "align",
    enable_kv_cache_events: bool = False,
) -> SchedulerFixture:
    """Build a scheduler for one attention group plus one Mamba group."""
    # attention 组与 mamba 组共享 scheduler_block_size（两者的 LCM），
    # MambaSpec 的 block_size 通常更大且支持细粒度 hash 命中。
    attention_spec = FullAttentionSpec(
        block_size=attention_block_size,
        num_kv_heads=NUM_KV_HEADS,
        head_size=HEAD_SIZE,
        dtype=DTYPE,
    )
    mamba_spec = MambaSpec(
        block_size=block_size,
        shapes=((1, 1),),
        dtypes=(torch.float32,),
        mamba_cache_mode=mamba_cache_mode,
    )
    vllm_config.cache_config.prefix_cache_retention_interval = 0
    vllm_config.cache_config.mamba_cache_mode = mamba_cache_mode
    sched = SimpleCPUOffloadScheduler(
        vllm_config=vllm_config,
        kv_cache_config=kv_cache_config,
        cpu_capacity_bytes=cpu_capacity_bytes,
        scheduler_block_size=scheduler_block_size,
        hash_block_size=hash_block_size,
        lazy_offload=lazy,
    )
    return SchedulerFixture(scheduler=sched, gpu_block_pool=gpu_block_pool, ...)

def test_hybrid_store_uses_resolved_group_block_sizes() -> None:
    """Replicated groups must not be scaled by the DCP world size."""
```

```

attention_size, mamba_size = sched.group_block_sizes
assert attention_size == attention_block_size * dcp_world_size
# 复制型 group 保持自身块大小: DCP 不会把它切开。
assert mamba_size == mamba_block_size
# 两个 group 都位置稳定, 因此已确认的块必须全部被 offload;
# 若按 cp_world_size 误放大, mamba 组只会写入一半的块。
assert sched._reqs_to_store[req.request_id].num_stored_blocks == [
    confirmed // attention_size,
    confirmed // mamba_size,
]

```

vllm/v1/core/single_type_kv_cache_manager.py

新增 `has_positionally_stable_blocks` 属性, 把「位置游标能否跟随该块表」抽象为 manager 级契约, 并由 `MambaManager` 在 align 模式返回 False, 是连接 offload 扫描与 Mamba 状态语义的关键接口。

```

class SingleTypeKVCacheManager(ABC):
    supports_fine_grained_hash_lookup: ClassVar[bool] = False

    @property
    def has_positionally_stable_blocks(self) -> bool:
        """Whether positional offload scans can follow this block table.

        True means already-scanned indices will not be reused for a different
        token range. Managers may still null blocks that left their retention
        window, as long as the cursor's positional history remains valid.
        """
        # 默认假定块表位置稳定: 已扫描过的下标可以被单调游标记住,
        # 只有会原地搬迁或复用下标的 spec 才需要覆写。
        return True


class MambaManager(SingleTypeKVCacheManager):
    supports_fine_grained_hash_lookup: ClassVar[bool] = True

    @property
    def has_positionally_stable_blocks(self) -> bool:
        # align 模式的 Mamba 会把内部状态置空, 并把 speculative block 原地搬迁,
        # 位置游标无法解析其块表; all 等其它模式仍保持位置上的确定性。
        return self.mamba_cache_mode != "align"

```

评论区精华

- CodeRabbit 在 `vllm/v1/simple_kv_offload/manager.py` 第 707 行给出 Critical 级别结论: 「merged_block_meta must stay index-parallel with merged_gpu_block_ids ... When enable_kv_cache_events is True ... _process_store_completion then fails its len(block_meta) == len(gpu_block_ids) assertion」, 并建议边界循环里补一个空 dict。作者回复「Fixed. The boundary loop now appends an empty metadata dict alongside

the block ids ... Covered by test_boundary_handoff_keeps_block_meta_index_parallel, which fails without it.]

- CodeRabbit 在 `_queue_finished_eager_store` 附近 (第 1169 行) 指出 Major 问题: 若 decode token 把最后一个 prompt 物理块填满, `_select_eager_blocks_to_store()` 已收入该块, `_find_fa_partial_tail_source()` 会再返回同一块, 「This appends one GPU block twice, allocates two CPU blocks, and reduces effective CPU-cache capacity」。作者确认并加上 `partial_tail not in gpu_ids` 与回归用例 `test_finished_eager_store_does_not_duplicate_completed_partial_tail`。
- 测试层面的两个未定论 nitpick: 其一, `get_num_new_matched_tokens` 的日志断言建议改用 `caplog_vllm fixture` 而非 `caplog` (围绕 `test_scheduler.py` 2367-2373 行); 其二, `_make_hybrid_attention_mamba_scheduler` 中 `num_cpu_blocks` 未生效, `cpu_capacity_bytes` 由 GPU tensor 尺寸推导, 且返回的 `fixture` 未带 `num_groups`。这两条在提供的材料中没有明确的解决记录, 属于遗留改进点。
- 维护者 `dllehr-amd` 直接 APPROVED (「Thanks @YukioZzz」), 未提出额外技术异议; `billishyahao` 协助找人 review 并追问 CI 失败。
- 过程性噪音较多: `mergify` 反复报 merge conflict, `/amd-ci run` 因权限被拒多次, 最终由 `shen-shanshan` 触发 AMD CI #12806 / #12818 / #12829, 作者最后又跑 `/ci run` 与 `/ci retry`。
- `boundary handoff` 的 `block_meta` 与 `block id` 列表必须下标对齐 (correctness): 作者确认修复: 边界循环补空 metadata 占位, 并新增 `test_boundary_handoff_keeps_block_meta_index_parallel` 回归用例, 且把修复折进引入该循环的那个 commit 而不是叠加新 commit。
- FA 部分尾块可能被重复存储、重复占用 CPU block (correctness): 作者加上 `partial_tail not in gpu_ids` 守卫, 并补 `test_finished_eager_store_does_not_duplicate_completed_partial_tail`, 说明该用例在没有守卫时会失败。
- 测试日志断言应使用 `caplog_vllm fixture` (testing): 材料中未见明确采纳记录, 属于遗留建议。
- 新测试夹具的 `num_cpu_blocks` 参数未生效 (testing): 属于潜在陷阱而非当前失败点, 材料中未见修复记录, 未来写容量压力测试时需先修夹具。
- CI 与合并冲突的反复拉锯 (other): PR 在通过 AMD CI 与上游 CI 后由 `dllehr-amd` 批准合并。

风险与影响

- 风险:
 1. 位置稳定性契约易被漏写: `has_positionally_stable_blocks` 默认 True, 只有 `MambaManager` 覆写。若后续新增会原地搬迁或复用下标的 `spec` 而忘记覆写, `eager store` 会静默把错误 token 范围的 KV 以合法 hash 发布, 属于难以察觉的正确性回归。
 2. 跨模块数据契约: `eager store` 现在依赖 `scheduler_output.kv_connector_block_state.boundary_state_offloads` 的格式与下发时机。上游 (KV cache manager / scheduler) 若调整 `handoff` 语义, 功能会退化为低命中率而非报错, 只能靠 `BoundaryStoreStats` 计数发现。

3. 触发条件收窄带来的命中率波动: `get_num_new_matched_tokens` 在 `num_computed_tokens % self.block_size != 0` 时直接放弃外部查找 (`logger.warning_once`) , PR 也在 Limitation 中承认「If a local prefix is not scheduler-block aligned, SimpleCPU keeps the local hit and skips the external lookup rather than starting a transfer inside a destination block.」
4. 容量与配额: finish 时新增的 FA 部分尾块会额外占用 CPU block, 且 `len(gpu_ids) < self.cpu_block_pool.get_num_free_blocks()` 在 append 前只做一次判断, 批量场景下需要留意边界。
5. 兼容性: `HybridKVCacheCoordinator.__init__` 与 `get_kv_cache_coordinator` 新增 `allow_partial_hash_hits: bool = True` 默认参数, 对未传参的调用方保持旧行为; `group_block_sizes` 由局部变量升级为实例属性, 重命名 `cacheable_block_sizes` 只影响本文件, 风险可控。
6. 验证方式: 仓库内只有单元测试, 端到端正确性依赖作者在 Kimi-K3 TP8/DCP8 与 Qwen3.5-4B TP8/DCP2 上的人工跑分 (GSM8K、AgentX/AIPerf) , 缺少可持续回归的 e2e 用例。 - 影响: 用户侧: hybrid 模型 (Kimi-K3、Qwen3.5 这类 GDN 支撑的 MambaSpec 组) 在 eager SimpleCPU offload 下能真正命中外部 KV——PR body 给出 Kimi-K3 replay external hit $4048896/4693652 = 86.26\%$ 、Qwen3.5 93.44% , 且 cold/replay 输出一致。系统侧: `KVCacheCoordinator.group_block_sizes` 与 `allow_partial_hash_hits` 成为跨模块接口, `has_positionally_stable_blocks` 成为 KV cache manager 与 connector 之间的新契约; 同时 `kv_connector_block_state` 升级为 eager store 的一等输入。运维 / 团队侧: `BoundaryStoreStats` 提供了 `published/stored/dropped_reason` 等计数器, 把「边界未 offload 导致命中率下降」从无日志的静默行为变成可观测指标。 - 风险标记: 核心路径变更, 跨模块契约变更, Mamba align 语义耦合, 缺少仓库内端到端用例, 新增诊断计数器需人工解读

关联脉络

- PR #54735 SimpleCPU offload 前置改造 (本 PR 的栈基): PR body 明确写「Stacked on #54735」, 本 PR 的 `head_ref` 也命名为 `yichaozhu/pr53917-fine-grained-hybrid-hits`, 说明细粒度 hybrid 命中是一条被拆分推进的功能线。
- PR #55531 [KV Connector] Support symmetric DCP disagg for hybrid mamba models: 同属 hybrid mamba + DCP + KV connector 方向, 改动文件与 `vllm/distributed/kv_transfer/.../nixl/*`、`vllm/config/parallel.py` 有交集语义, 本 PR 的 DCP 组几何口径调整与之呼应。
- PR #53695 [ROCm][Feature] Support KV connectors with `ROCM_AITER_UNIFIED_ATTN`: 同一条「KV connector 与 attention backend / 缓存布局对接」的演进线, 本 PR 的 `has_positionally_stable_blocks` 与 `supports_fine_grained_hash_lookup` 属于同类 manager 能力契约。
- PR #49962 Mamba + DCP 事件元数据捕获路径修复 (被本 PR 测试引用): 被删除的 `test_mamba_dcp_metadata_guard` docstring 中写明「When #49962 fixes the capture path, the guard stops triggering」, 本 PR 用 `test_mamba_align_skips_positional_event_metadata` 取代该守卫断言, 两条线在

Mamba+DCP 事件元数据上直接相关。