PR #32972 完整报告

sgl-project/sglang

[unified-memory] Let Kimi-Linear use the paged MLA attention backends

合并时间：2026-07-31 16:32

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32972

执行摘要

- 一句话：统一内存池放开 paged MLA 注意力后端，并修复 cuda-graph 静默错读
- 推荐动作：值得精读。核心看点：(1) create_flashmla_kv_indices_triton 用 kernel 内 v2p gather + PAGE_MULT 取代 host 端 remap，省一次 launch 与临时 tensor；(2) 用能力探测（full_v2p_page_table 是否存在）而非 multiplier 推断 unified pool，规避单层 MLA 配置的静默错读；(3) flashinfer decode 采用原地写回而非 per-role allowlist 分裂，并明确依赖 flashinfer plan() 的复制合约；(4) cuda-graph 下"图外预计算 dense 写位置 + replay 前清零尾部"的写法，解决 pad 行 stale loc 问题。PR body 中关于 GSM8K 0.000 的复现与诊断过程（无 crash、无警告、仅在服务有负载历史后出现）本身就是很好的 cuda-graph 调试案例。

功能与动机

PR body 明确说明动机：With the dense per-layer MLA views from #32971 in place, the stock paged MLA kernels can read the unified pool directly — only their kv_indices / block tables need remapping to dense ids. No change to the physical layout, the allocator, or compaction. 此前 --enable-unified-memory 隐含 --enable-page-major-kv-layout，而 page-major 允许清单只有 triton，导致 unified pool 下的 MLA 模型无法使用性能更好的 paged MLA 后端；同时 review 阶段发现 flashinfer decode 在 cuda-graph 下读取未经翻译的虚拟 id 会静默产出错误输出（GSM8K 0.000，无 crash 无警告），需要一并修正。

实现拆解

1. 内核层：block table 的稠密页映射

python/sglang/kernels/ops/kvcache/kv_indices.py 的 create_flashmla_kv_indices_triton 新增 v2p_ptr 和 PAGE_MULT 两个参数，默认值（None / 1）保证静态池行为逐字节不变。kernel 在写 block table 时先由 token 算出虚拟页，再通过 v2p_ptr 做 masked gather 得到物理页，最后乘以 PAGE_MULT（= 全注意力 MLA 层数），直接定位 (num_pages * L, page_size, kv_cache_dim) 稠密视图中的页块。padded lane 因 mask 不参与 v2p 查表，保持 -1 填充。

2. 能力探测层：UnifiedMLAHooks / unified_mla_hooks

python/sglang/srt/layers/attention/flashinfer_mla_backend.py 新增冻结 dataclass UnifiedMLAHooks（v2p_page_table、translate_kv_loc_dense、kernel_page_multiplier、enabled）与探测函数 unified_mla_hooks(allocator)。判据是 full_v2p_page_table 是否存在而非 kernel_page_multiplier > 1，对应 review 中 P2 指出的单层 MLA 配置回归。multi_ended_allocator.py 配套新增 full_v2p_page_table property，透传 full 子池的 virtual_to_physical。

3. TRTLLM 路径接线与 cuda-graph 安全

python/sglang/srt/layers/attention/trtllm_mla_backend.py 在 __init__ 中取 hooks 并保存 _v2p_page_table / _kernel_page_multiplier / _translate_kv_loc_dense / _unified_mla。两处 block-table 构造（_create_block_kv_indices 与 _apply_cuda_graph_metadata）把新参数传给 Triton kernel。decode 侧：init_cuda_graph_state 分配捕获稳定的 cuda_graph_out_cache_loc_dense（int64）；init_forward_metadata_out_graph 在图外把 out_cache_loc 翻译成 dense 写位置，并把超出本次 batch 的尾部清零；forward_decode 用 _decode_dense_loc + loc_is_dense=True 写 KV。
python/sglang/srt/mem_cache/memory_pool.py 的 set_mla_kv_buffer 新增 loc_is_dense 参数，为真时跳过 _full_translate。

4. FlashInfer 的 token 级原地翻译

FlashInferMLAIndicesUpdaterDecode.call_begin_forward 在索引 kernel 之后执行 valid = kv_indices[:paged_kernel_lens_sum]; valid.copy_(self._translate_kv_loc_dense(valid))，只翻译索引 kernel 刚填充的 prefix。这是对 review 中 P1 的修复：cuda-graph replay 下 kv_indices 就是捕获稳定 buffer，而 fast_mla_decode_plan 忽略 kv_indices 参数，rebind 局部变量是死代码；原地写回依赖 flashinfer 侧 use_cuda_graph 下 plan() 把 _kv_indices_buf 复制进来的合约。prefill 端（不捕获图）则直接 eager gather 后 .to(torch.int32)。

5. 配置与测试配套

python/sglang/srt/server_args.py 的 _handle_page_major_kv_layout 在 enable_unified_memory and use_mla_backend() 时把允许清单从 {"triton"} 扩为 {"triton", "trtllm_mla", "flashinfer", "cutedsl_mla", "tokenspeed_mla"}，其余分支保持原断言。新增三个测试：test_unified_mla_dense_block_table.py（kernel 与 Python 参照一致性、identity 情形、单层回归、padded lane、token 级与页级翻译一致性）、test_page_major_backend_allowlist.py（allowlist 边界：MLA+unified 才放行、MHA 拒绝、非 unified 拒绝、未接线后端拒绝）、test_kimi_linear_unified_memory.py（nightly 2×H200 TP2 真实 Kimi-Linear-48B，GSM8K + prefix-cache 分支，后端钉 triton 以便 H100 runner 启动）。

关键文件：

- python/sglang/srt/layers/attention/flashinfer_mla_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 UnifiedMLAHooks, unified_mla_hooks, FlashInferMLAIndicesUpdaterDecode.call_begin_forward, FlashInferMLAIndicesUpdaterPrefill.call_begin_forward）：新增 UnifiedMLAHooks /

unified_mla_hooks 能力探测，并在 flashinfer decode/prefill updater 中实现 token 级 dense 翻译；decode 的原地写回是修复 cuda-graph 静默 0.000 的关键。

- python/sglang/srt/layers/attention/trtllm_mla_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 TRTLLMMLABackend._create_block_kv_indices, TRTLLMMLABackend.init_cuda_graph_state, TRTLLMMLABackend.init_forward_metadata_out_graph, TRTLLMMLABackend.forward_decode）：paged MLA 主路径：block table 传入 v2p_ptr/PAGE_MULT，cuda-graph 解码的稠密写位置改为图外预计算，并处理 replay-prep 尾部清零。
- python/sglang/kernels/ops/kvcache/kv_indices.py（模块 内核层；类别 infra；类型 infrastructure；符号 create_flashmla_kv_indices_triton）：内核层新增 v2p_ptr / PAGE_MULT 参数，是全部 paged MLA 后端 dense 映射的物理基础，默认值保证静态池行为不变。
- test/registered/unit/mem_cache/test_unified_mla_dense_block_table.py（模块 单元测试；类别 test；类型 test-coverage；符号 _fill_block_table, _reference, TestDenseBlockTable, _make_batch）：新增 375 行单元测试，覆盖 kernel 与 Python 参照的一致性、identity 保真、单层 MLA 回归、padded lane 与 token/ 页级翻译一致性，是该 PR 正确性的核心防线。
- test/registered/unit/server_args/test_page_major_backend_allowlist.py（模块 单元测试；类别 test；类型 test-coverage；符号 _accepts, TestPageMajorBackendAllowlist, test_triton_always_allowed, test_dense_mla_backends_allowed_under_unified_mla）：钉住 allowlist 边界：只有 MLA + unified memory 才放行 paged MLA 后端，MHA、非 unified、未接线后端一律拒绝，防止异常静默扩大。
- test/registered/models_e2e/test_kimi_linear_unified_memory.py（模块 端到端；类别 test；类型 test-coverage；符号 TestKimiLinearUnifiedMemory）：nightly 端到端测试，真实 Kimi-Linear-48B TP2 验证 unified memory 与静态池准确率相当，并覆盖 prefix-cache 分支在 compaction 后的虚拟 loc 重放。
- python/sglang/srt/server_args.py（模块 服务参数；类别 source；类型 configuration；符号 ServerArgs._handle_page_major_kv_layout）：_handle_page_major_kv_layout 的 allowlist 扩展是本 PR 的配置入口，决定哪些后端在 unified MLA 下可用，并保留对 MHA / 静态池的严格拒绝。
- python/sglang/srt/mem_cache/memory_pool.py（模块 缓存池；类别 source；类型 core-logic；符号 UnifiedKVTokenToKVPool.set_mla_kv_buffer）：set_mla_kv_buffer 新增 loc_is_dense 参数，使捕获图内写 KV 时可跳过 _full_translate，是 cuda-graph dense 写路径的落点。
- python/sglang/srt/mem_cache/multi_ended_allocator.py（模块 分配器；类别 source；类型 core-logic；符号 full_v2p_page_table）：新增 full_v2p_page_table property，是 unified_mla_hooks 能力探测的依赖项，透传 full 子池的页面级 virtual_to_physical 表。

关键符号：create_flashmla_kv_indices_triton, unified_mla_hooks, FlashInferMLAIndicesUpdaterDecode.call_begin_forward, FlashInferMLAIndicesUpdaterPrefill.call_begin_forward,

TRTLLMMLABackend._create_block_kv_indices,TRTLLMMLABackend.init_cuda_graph_state, TRTLLMMLABackend.init_forward_metadata_out_graph, TRTLLMMLABackend.forward_decode,ServerArgs._handle_page_major_kv_layout, MultiEndedAllocator.full_v2p_page_table,UnifiedKVTokenToKVPool.set_mla_kv_buffer

关键源码片段

python/sglang/srt/layers/attention/flashinfer_mla_backend.py

新增 UnifiedMLAHooks / unified_mla_hooks 能力探测，并在 flashinfer decode/prefill updater 中实现 token 级 dense 翻译；decode 的原地写回是修复 cuda-graph 静默 0.000 的关键。

```
# python/sglang/srt/layers/attention/flashinfer_mla_backend.py
#
# 统一内存 MLA 池下，paged MLA 后端需要把 VIRTUAL token / page id 翻译成
# DENSE 内核 id。这一组钩子统一封装分配器能力，探测判据是 v2p 表是否
# 存在，而不是 kernel_page_multiplier > 1 —— 只有单 full-attention
# MLA 层的配置（如 PP rank 只拥有一个 MLA 层）multiplier 为 1，但
# req_to_token 仍是虚拟 id，漏掉 v2p 映射会在 compaction 后读写错页。

@dataclass(frozen=True)
class UnifiedMLAHooks:
    """Unified pool 下 paged MLA 后端所需的分配器钩子。"""
    v2p_page_table: Optional[torch.Tensor] # 页面级 virtual -> physical 映射表
    translate_kv_loc_dense: Optional[Callable[..., torch.Tensor]] # virtual token -> dense id
    kernel_page_multiplier: int # dense 页步长（= 全注意力 MLA 层数）
    enabled: bool


def unified_mla_hooks(allocator) -> UnifiedMLAHooks:
    v2p = getattr(allocator, "full_v2p_page_table", None)
    if v2p is None:
        # 静态分区池：req_to_token 已是物理 id，全部钩子关闭，行为与改动前一致。
        return UnifiedMLAHooks(None, None, 1, False)
    return UnifiedMLAHooks(
        v2p_page_table=v2p,
        translate_kv_loc_dense=getattr(allocator, "translate_kv_loc_dense", None),
        kernel_page_multiplier=getattr(allocator, "kernel_page_multiplier", 1),
        enabled=True,
    )


class FlashInferMLAIndicesUpdaterDecode:
    def call_begin_forward(self, wrapper, req_pool_indices, paged_kernel_lens,
                           paged_kernel_lens_sum, q_indptr, kv_indptr,
                           init_metadata_replay=False, spec_info=None, **kwargs):
        # ... create_flashinfer_kv_indices_triton 先写入 VIRTUAL kv_indices ...
        if self._translate_kv_loc_dense is not None:
```

```
            # 必须原地写回：cuda-graph replay 时 kv_indices 就是捕获稳定的
            # buffer（fast_decode_kwargs["kv_indices"]），而 fast_mla_decode_plan
            # 完全忽略 kv_indices 参数 —— 只 rebind 局部变量会让捕获的内核
            # 继续读 VIRTUAL id，曾直接复现为 GSM8K 0.000。只翻译索引内核
            # 刚填充的 prefix，陈旧 tail 从不参与 v2p 查表。
            valid = kv_indices[:paged_kernel_lens_sum]
            valid.copy_(self._translate_kv_loc_dense(valid))
        # ... wrapper.plan(...) —— flashinfer 在 use_cuda_graph 下把构造时
        # 传入的 buffer 存为 _kv_indices_buf，plan() 复制进该 buffer，
        # 因此内核读到的正是刚写回的 dense id（无需改 flashinfer 本体）。
```

python/sglang/srt/layers/attention/trtllm_mla_backend.py

paged MLA 主路径：block table 传入 v2p_ptr/PAGE_MULT，cuda-graph 解码的稠密写位置改为图外预计算，并处理 replay-prep 尾部清零。

```
# python/sglang/srt/layers/attention/trtllm_mla_backend.py
#
# cuda-graph 解码路径下，KV 写位置不能在捕获图内做 virtual -> dense 翻译
# （会捕获分配）。方案：图外用捕获稳定 buffer 预计算 dense 写位置，
# 图内 set_mla_kv_buffer 只消费该 buffer，因此不捕获任何翻译。

def init_forward_metadata_out_graph(self, forward_batch, in_capture=False):
    # ... 既有 metadata / block table 更新 ...
    if self._unified_mla and forward_mode.is_decode_or_idle():
        out_cache_loc = forward_batch.out_cache_loc
        n = out_cache_loc.shape[0]
        dst = self.cuda_graph_out_cache_loc_dense[:n]
        # 图外翻译：VIRTUAL loc -> DENSE loc，写入捕获稳定 buffer 前缀。
        self._translate_kv_loc_dense(out_cache_loc, out=dst)
        # replay-prep 拿到的是 raw（未 padding）的 out_cache_loc，但捕获的
        # 内核消费整条 tier —— 清零尾部，否则 pad 行沿用旧 replay 留下的
        # stale dense loc，把垃圾 KV 散射进存活页面。
        self.cuda_graph_out_cache_loc_dense[n:].zero_()
        self._decode_dense_loc = dst


def forward_decode(self, layer, forward_batch, ...):
    if self._decode_dense_loc is not None:
        # 捕获图内写 KV：loc 已是 dense，跳过池层 _full_translate。
        self.token_to_kv_pool.set_mla_kv_buffer(
            layer, self._decode_dense_loc, k, k_rope, loc_is_dense=True
        )
    else:
        # eager 或静态池：走 pool 自身翻译，行为与改动前一致。
        self.token_to_kv_pool.set_mla_kv_buffer(
            layer, forward_batch.out_cache_loc, k, k_rope
        )
```

python/sglang/kernels/ops/kvcache/kv_indices.py

内核层新增 v2p_ptr / PAGE_MULT 参数，是全部 paged MLA 后端 dense 映射的物理基础，默认值保证静态池行为不变。

```
# python/sglang/kernels/ops/kvcache/kv_indices.py
# create_flashmla_kv_indices_triton 新增两个参数，默认值保持静态池行为
# 完全不变（v2p_ptr=None、PAGE_MULT=1）。

def create_flashmla_kv_indices_triton(
    req_to_token_ptr, req_pool_indices_ptr, seq_lens_ptr,
    kv_indptr, kv_indices_ptr, req_to_token_ptr_stride,
    kv_indices_ptr_stride, PAGED_SIZE: tl.constexpr = 64,
    v2p_ptr=None, PAGE_MULT: tl.constexpr = 1,
):
    # ... 每行每个 block 内逐 token 加载，data 为该 token 的 VIRTUAL id ...
    page = data // PAGED_SIZE
    if v2p_ptr is not None:
        # masked load：padded lane（mask_out 为 0）永不越界索引 v2p 表。
        page = tl.load(v2p_ptr + page, mask=mask_out, other=0)
    # dense_page = physical_page * layer_num，直接定位
    # (num_pages * L, page_size, kv_cache_dim) 稠密视图中的页块。
    tl.store(
        kv_indices_ptr + pid * kv_indices_ptr_stride + paged_offset_out,
        page * PAGE_MULT,
        mask=mask_out,
    )
```

评论区精华

两条 Codex 自动 review 评论都指向真实缺陷并被落地修复：

- P1（server_args.py，第 7722 行附近）：When unified-memory MLA uses FlashInfer for decode with the default CUDA-graph path, replay writes raw virtual IDs into the capture-stable fast_decode_kwargs["kv_indices"], then the new translation rebinds only the local kv_indices to a fresh tensor; fast_mla_decode_plan does not use that tensor...。建议把 flashinfer 从 decode allowlist 排除。作者未采纳排除方案，而是改为原地写回翻译，原因是 --attention-backend flashinfer 同时解析 prefill 与 decode 两个角色，per-role 排除会拒绝唯一实际使用的调用组合；修复依赖 flashinfer 在 use_cuda_graph 下 plan() 复制进构造时 buffer 的合约，无需改 flashinfer 本体。
- P2（trtllm_mla_backend.py）：kernel_page_multiplier is 1 even though the allocator still returns virtual token IDs... leaving both the block table and CUDA-graph write locations in virtual ID space。该问题在第 3 个 commit（dbe8227）修复：unified_mla_hooks 改用 full_v2p_page_table 能力探测，并新增 TestUnifiedMLAHookDetection.test_single_full_attention_layer_pool_is_still_unified 与 TestDenseBlockTable.test_single_full_attention_layer_still_maps_v2p 钉住该回归。
  - flashinfer decode 是否应从 unified allowlist 排除 (correctness): 作者未采用 per-role 排除，而是改为在 create_flashinfer_kv_indices_triton 之后对 kv_indices[:paged_kernel_lens_sum] 做原地 copy_ 翻译；理由是

--attention-backend flashinfer 同时解析 prefill 和 decode，decode-only 拒绝会拒绝唯一实际使用的调用组合。修复依赖 flashinfer 在 use_cuda_graph 下 plan() 复制进构造 buffer 的合约，无需改 flashinfer。

- 用 kernel_page_multiplier > 1 检测 unified pool 的缺陷 (correctness): 第 3 个 commit（dbe8227）把探测改为基于 allocator 能力（full_v2p_page_table 是否存在），并新增两个回归测试：test_single_full_attention_layer_still_maps_v2p 与 TestUnifiedMLAHookDetection.test_single_full_attention_layer_pool_is_still_unified。

风险与影响

- 风险：
  1. cuda-graph 正确性依赖（flashinfer_mla_backend.py）：原地翻译的正确性依赖 flashinfer 侧 BatchMLAPagedAttentionWrapper 在 use_cuda_graph 下把构造时传入的 buffer 保存为 _kv_indices_buf 且 plan() 复制进该 buffer 的合约。若 flashinfer 将来改为 rebind，会静默回到虚拟 id 错读。
  2. stale tail 清零范围（trtllm_mla_backend.py）：cuda_graph_out_cache_loc_dense[n:].zero_() 只在 init_forward_metadata_out_graph 的 decode/idle 分支执行；若有新 forward mode 绕过该入口，pad 行会沿用旧 replay 的 stale dense loc，散射垃圾 KV。
  3. TP 覆盖不足：PR 自述 TP1 only，未覆盖 TP-sharded 行为，且无 perf A/B、无 retraction 或长上下文压力测试。
  4. e2e 测试钉死 triton：test_kimi_linear_unified_memory.py 因 H100 runner 上解析默认会落到 fa3（无法读 dense 视图），被迫钉 --attention-backend triton，因此 paged MLA 后端（最容易出 dense-id 翻译 bug 的地方）没有端到端 CI 覆盖，只有单元测试与手工 B300 结果。
  5. 两个 pre-existing 问题未解决（非本 PR 引入）：flashinfer prefill 与不同 decode 后端配对会崩（mha_chunk_kv_cache.forward 的 q.shape[0] 与 qo_indptr[-1] 不匹配）；fp8 × flashinfer prefill 会把整个 KV pool 物化为 bf16 导致 270 GiB OOM。 - 影响：对用户：--enable-unified-memory 下的 MLA 模型（当前主要是 Kimi-Linear 这类 KDA + MLA 混合模型）不再被迫使用 Triton 注意力后端，可在 Blackwell（B300 / SM120）上选用 trtllm_mla、cutedsl_mla、tokenspeed_mla（fp8）等 paged MLA 内核，decode 阶段获得与静态池相当的正确性（GSM8K 0.895–0.925 噪声带内），并解锁后续性能优化空间。对系统：静态池路径在 v2p_ptr=None / PAGE_MULT=1 下保持逐字节一致（有 identity 测试钉住），回归面被限制在 unified-memory 分支。对团队：确立了 unified pool 下 paged 后端接入的统一模式（hook 探测 + 内核内映射 + 图外预计算），后续 flashmla / cutlass_mla 可按同样方式接入；Codex review 中发现的"捕获稳定 buffer 与局部 rebind 的差异"是值得沉淀的 cuda-graph 调试经验。 - 风险标记：cuda-graph 正确性依赖，核心路径变更，TP1 验证受限，paged 后端缺少 e2e 覆盖，能力探测替代脆推断

关联脉络

- PR #32971 [unified-memory] Kimi-Linear unified-memory Triton 支持（标题未在材料中给出，仅从 body 可知为 stack 基础）：PR body 明确声明 Stacked on #32971，且本 PR 的 base branch 为 feature/unified-memory-kimi-linear-triton；#32971 提供的稠密逐层 MLA 视图（build_dense_mla_views）是本 PR 让 paged MLA 内核直接读 unified pool 的前提。