PR #33046 完整报告

sgl-project/sglang

[unified-memory] Support fa3, the default MLA backend on pre-Blackwell hosts

合并时间：2026-08-01 02:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33046

执行摘要

- 一句话：支持 fa3 后端，修复统一内存默认配置启动失败
- 推荐动作：值得精读，尤其关注三个设计决策： 1) 把 dense 翻译折进 normal_decode_set_metadata 融合 gather，capture 稳定的 page_table 写入即已翻译，避免调用方遗漏与跨 replay 指针稳定性问题； 2) eager 路径选在 init_forward_metadata 尾部 // page_size 归约之前做单点漏斗，利用 dense(page_start) // ps == phys_page * L 恒等式让一个位置服务两种 page size； 3) hooks 模块独立抽取避免后端间反向依赖，以及刻意解除 nightly 测试后端固定的理念。需要跟进的是 full-prefill graph 组合尚未覆盖的缺口。

功能与动机

PR body 明确指出："the page-major full-attention allowlist still omitted fa3 — which is exactly what an unspecified --attention-backend resolves to for an MLA model on Hopper. So on H100/H200 the feature failed at startup under its own default configuration, and you had to know to pass --attention-backend triton. This is the pre-Blackwell counterpart of the same gap #32972 had for flashinfer, on the more common hardware." 即 #32971/#32972 让 unified-memory 支持 Kimi-Linear 后，用户在 H100/H200 上按默认配置启动仍会失败，属于功能上线即不可用的关键缺口。

实现拆解

1. Hooks 独立成模块：新增 python/sglang/srt/layers/attention/unified_mem_hooks.py（+70 行），将 UnifiedMLAHooks（msgspec.Struct，frozen）与探测函数 unified_mla_hooks(allocator) 从 flashinfer_mla_backend.py 迁出，后者删除 45 行本地定义改为 import；trtllm_mla_backend.py 的 import 同步迁移。这样 fa3 无需反向依赖 flashinfer MLA 模块即可获得钩子。探测键刻意选用 full_v2p_page_table 而非 kernel_page_multiplier > 1：只拥有一个 full-attention 层的 rank 其 multiplier 为 1 但 req_to_token 仍是虚拟 id，漏掉翻译会在 compaction 后静默寻址错误页。
2. captured decode：翻译折进融合内核：python/sglang/kernels/ops/attention/metadata.py 的 _fused_metadata_kernel_general 与 _fused_metadata_kernel_ps1_no_swa 新增 v2p_ptr / PAGE_MULT 参数，normal_decode_set_metadata 对外新增 v2p_page_table / kernel_page_multiplier 并透传。内核内翻译从 page_table_val 派生而非 page_index（SWA 分支仍需虚拟值），mask 保证 padding lane 不越界。折叠而非事后翻译，是为了让

capture 稳定的 page_table 写入时即已是 dense id：没有调用方可遗忘的独立 pass，也没有跨 cuda-graph replay 必须保持指针稳定的临时量；默认 None / 1 身份映射使静态池路径与改动前字节一致。

3. eager 路径单点漏斗：flashattention_backend.py 构造函数计算 self._unified_dense = self._unified_hooks.enabled and self.use_mla（MLA-only，MHA/SWA 子池的 strided 布局不受影响）；在 init_forward_metadata 尾部、// page_size 归约之前对 page_table 做一次 translate_kv_loc_dense 翻译（先 flatten 再 reshape）。数学依据：dense(t) = phys_page * (ps * L) + t % ps，故 dense(page_start) // ps == phys_page * L，恰为内核需要的 dense page id，一个位置服务两种 page size，且继承 tombstone 夹取（未写槽落入保留的 page-0 sink）。
4. 配置放行：server_args.py::_handle_page_major_kv_layout 的 allowed_full 集合加入 "fa3" 并更新注释；test_page_major_backend_allowlist.py 把 fa3 从 UNWIRED_BACKENDS（flashmla / cutlass_mla / trtllm_mha / aiter）移至 DENSE_MLA_BACKENDS，同时锁定 MHA 路径与无 unified-memory 场景仍拒绝。
5. 测试配套与验证：test_unified_mla_dense_block_table.py 新增 TestFa3MetadataDenseBlockTable 5 个 GPU 用例（hooks 缺省身份一致、ps1 快速路径、general 路径、单 full-attention 层、与 flashmla block table 逐页对齐）；e2e test_kimi_linear_unified_memory.py 删除 --attention-backend triton 固定，让 nightly 跑用户真实默认配置（H100 即 fa3）。GSM8K 验证：2×H200 TP2、默认解析（fa3）400 例 0.917 vs 静态池 0.915；--page-size 64 200 例 0.905（走 general 内核）。

关键文件：

- python/sglang/srt/layers/attention/unified_mem_hooks.py（模块 统一内存；类别 source；类型 dependency-wiring；符号 UnifiedMLAHooks, unified_mla_hooks）：新增的共享钩子模块，承载 UnifiedMLAHooks 与 unified_mla_hooks 探测逻辑，是 fa3、flashinfer_mla、trtllm_mla 三个后端族共同的基础设施，也是本 PR 依赖解耦的关键。
- python/sglang/srt/layers/attention/flashattention_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 FlashAttentionBackend, init_forward_metadata, _apply_cuda_graph_metadata）：fa3 后端（FlashAttentionBackend）接入 unified-memory 的主战场：eager 路径在 init_forward_metadata 尾部做单点漏斗翻译，captured decode 路径向 normal_decode_set_metadata 传 v2p 参数，是本 PR 两条路径的分叉点。
- python/sglang/kernels/ops/attention/metadata.py（模块 融合内核；类别 infra；类型 infrastructure；符号 normal_decode_set_metadata, _fused_metadata_kernel_general, _fused_metadata_kernel_ps1_no_swa）：融合元数据内核的改造点：_fused_metadata_kernel_general 与 _fused_metadata_kernel_ps1_no_swa 新增 v2p_ptr / PAGE_MULT，normal_decode_set_metadata 透传，captured decode 的 page_table 写入时即完成 dense 翻译。
- python/sglang/srt/server_args.py（模块 启动参数；类别 source；类型 configuration；符号 _handle_page_major_kv_layout）：启动失败的根因所在：_handle_page_major_kv_layout 的 allowed_full 集合加入 fa3，是本次修复的开关点。

- 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 TestFa3MetadataDenseBlockTable, _run, _assert_live_prefix, test_identity_when_hooks_absent）：核心测试配套：新增 TestFa3MetadataDenseBlockTable 5 个 GPU 用例，覆盖 fa3 融合内核的身份一致、ps1 快速路径、general 路径、单层与跨后端逐页对齐。
- test/registered/models_e2e/test_kimi_linear_unified_memory.py（模块 端到端测试；类别 test；类型 test-coverage；符号 TestKimiLinearUnifiedMemory）：nightly e2e 测试删除 --attention-backend triton 固定，改为跑宿主真实默认后端（fa3），是『默认配置缺陷必须暴露』测试理念的落地。
- python/sglang/srt/layers/attention/flashinfer_mla_backend.py（模块 MLA 后端；类别 source；类型 refactor；符号 UnifiedMLAHooks, unified_mla_hooks）：UnifiedMLAHooks 与 unified_mla_hooks 的源出处，本 PR 将其迁出为独立模块，删除 45 行本地定义。
- test/registered/unit/server_args/test_page_major_backend_allowlist.py（模块 参数测试；类别 test；类型 test-coverage；符号 TestPageMajorBackendAllowlist, DENSE_MLA_BACKENDS, UNWIRED_BACKENDS）：锁定 allowlist 语义：fa3 从拒绝集移入 dense-MLA 集，同时保证 MHA 路径与无 unified-memory 场景仍拒绝。
- python/sglang/srt/layers/attention/trtllm_mla_backend.py（模块 MLA 后端；类别 source；类型 dependency-wiring；符号 unified_mla_hooks）：import 从 flashinfer_mla_backend 迁移到 unified_mem_hooks，配合 hooks 模块抽取，消除后端间反向依赖。

关键符号：unified_mla_hooks, normal_decode_set_metadata, _fused_metadata_kernel_general, _fused_metadata_kernel_ps1_no_swa, init_forward_metadata, _handle_page_major_kv_layout

关键源码片段

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

新增的共享钩子模块，承载 UnifiedMLAHooks 与 unified_mla_hooks 探测逻辑，是 fa3、flashinfer_mla、trtllm_mla 三个后端族共同的基础设施，也是本 PR 依赖解耦的关键。

```
# 统一内存池下 paged MLA 后端的 dense-view 钩子。
# 独立成模块的原因：fa3、flashinfer_mla、trtllm_mla（含 cutedsl_mla / tokenspeed_mla
# 子类）三个后端族都要用，谁也不该为了拿它去 import 另一个后端的模块。

class UnifiedMLAHooks(msgspec.Struct, frozen=True):
    """一个 KV allocator 的 dense-view 钩子。

    静态分区池下全部为 None / 1 / False——那时 req_to_token 里存的
    就是物理 id，不需要任何翻译。
    """

    # page 级虚拟 -> 物理表，由 block-table 内核 gather。
    v2p_page_table: Optional[torch.Tensor]
```

```
    # 虚拟 token id -> DENSE 内核面 id（tombstone 会被夹到 sink 页）。
    translate_kv_loc_dense: Optional[Callable[..., torch.Tensor]]
    # dense page 步长缩放（= full-attention MLA 层数）。
    kernel_page_multiplier: int
    enabled: bool


_STATIC_POOL = UnifiedMLAHooks(
    v2p_page_table=None,
    translate_kv_loc_dense=None,
    kernel_page_multiplier=1,
    enabled=False,
)


def unified_mla_hooks(allocator) -> UnifiedMLAHooks:
    """探测 allocator 是否挂了统一内存池的 dense-view 钩子。

    检测键是 v2p 表而不是 kernel_page_multiplier > 1：某个 rank 若只拥有
    一个 full-attention 层，multiplier 是 1 但 req_to_token 仍是虚拟 id——
    此时 dense id 与物理 id 重合，v2p gather 本身就是完整的翻译；
    漏掉它会在紧凑（compaction）后静默寻址错误的页。
    """
    v2p = getattr(allocator, "full_v2p_page_table", None)
    if v2p is None:
        return _STATIC_POOL
    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,
    )
```

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

fa3 后端（FlashAttentionBackend）接入 unified-memory 的主战场：*eager* 路径在 init_forward_metadata 尾部做单点漏斗翻译，*captured decode* 路径向 normal_decode_set_metadata 传 v2p 参数，是本 PR 两条路径的分叉点。

```
# ---------- FlashAttentionBackend.__init__ ----------
# 统一池：req_to_token 存虚拟 id，而 MLA 每层视图是 DENSE 的，
# 所以每个 page_table 都需要重映射。仅限 MLA——MHA/SWA 子池
# 保持 strided envelope 布局，fa3 根本无法读取。
self._unified_hooks = unified_mla_hooks(model_runner.token_to_kv_pool_allocator)
self._unified_dense = self._unified_hooks.enabled and self.use_mla

# ---------- init_forward_metadata 收尾 ----------
# 统一池：对上面所有 eager 分支（约 9 个）统一补一次翻译。
# 这些分支各自产出了新张量，直接 rebind 是安全的；captured 路径
# 则把翻译折进 normal_decode_set_metadata，必须原地写。
```

```
#
# 放在 `// page_size` 归约之前、token 空间内很关键：
# dense(t) = phys_page * (ps * L) + t % ps
# 所以 dense(page_start) // ps == phys_page * L，正好是内核要的
# dense page id——一个位置服务两种 page size。同时继承
# translate_kv_loc_dense 的 tombstone 夹取，未写过的 req_to_token
# 槽会落到保留的 page-0 sink 而非负 page id。
if self._unified_dense and metadata.page_table is not None:
    # 先展平再 reshape：page_size == 1 的翻译路径走 index_select，
    # 拒绝 2-D 索引；而 page_size > 1 走高级索引能接受 2-D——
    # 这正是当初那个 bug 只在默认配置下崩溃的原因。
    pt = metadata.page_table
    metadata.page_table = (
        self._unified_hooks.translate_kv_loc_dense(pt.reshape(-1))
        .to(torch.int32)
        .view(pt.shape)
    )
```

python/sglang/kernels/ops/attention/metadata.py

融合元数据内核的改造点：_fused_metadata_kernel_general 与 _fused_metadata_kernel_ps1_no_swa 新增 v2p_ptr / PAGE_MULT，normal_decode_set_metadata 透传，captured decode 的 page_table 写入时即完成 dense 翻译。

```
# ---------- _fused_metadata_kernel_general 内的 remap ----------
# 统一内存：虚拟 page -> 物理 page -> 该层的 dense block。
# 必须从 page_table_val 派生而不是 page_index——下面的 SWA 分支
# 还需要虚拟空间的值。mask 保证 padding lane 不会越界索引 v2p 表。
if v2p_ptr is not None:
    page_table_val = tl.load(v2p_ptr + page_table_val, mask=mask, other=0)
    page_table_val = page_table_val * PAGE_MULT
# 写回 page_table（pt_offsets 由 batch / column chunk 计算）
tl.store(page_table + pt_offsets, page_table_val, mask=mask, cache_modifier=".cg")

# ---------- _fused_metadata_kernel_ps1_no_swa 内的 remap ----------
# page_size == 1 快速路径：虚拟 token id 本身就是虚拟 page id。
# Kimi-Linear 默认走这里（fa3 不强制 page size），同样要先翻译再写回。
if v2p_ptr is not None:
    page_index = tl.load(v2p_ptr + page_index, mask=mask, other=0)
    page_index = page_index * PAGE_MULT

# ---------- normal_decode_set_metadata 新增参数 ----------
# 统一内存下第 4b 步直接折进内核：
# page_table = v2p_page_table[page] * kernel_page_multiplier
# 折叠而非事后翻译，是为了让 capture 稳定的 page_table 写入时
# 就已翻译好——没有调用方会忘记的独立 pass，也没有跨
# cuda-graph replay 必须保持指针稳定的临时量。静态池默认 None / 1。
def normal_decode_set_metadata(
    ...,
```

```
    page_size: int,
    swa_page_table: Optional[torch.Tensor] = None,
    token_to_kv_pool: Optional["SWAKVPool"] = None,
    v2p_page_table: Optional[torch.Tensor] = None,
    kernel_page_multiplier: int = 1,
):
    ...
```

评论区精华

唯一的 review 评论来自 chatgpt-codex-connector[bot]（自动机器人），P1 级：full-prefill graph（cuda_graph_config.prefill.backend == Backend.FULL）与 unified-memory + fa3 组合下，flashattention_backend.py 的 _init_full_cg_prefill_metadata 仍直接复制 page_indices // page_size（603-607 行），虚拟 req_to_token id 直达 FA3；新 remap 只覆盖 eager metadata 与 captured decode，虚拟物理分叉后会静默读错 KV 页。建议要么 remap 该指针稳定的 prefill 表、要么拒绝该组合。PR 内无回复、无修复，评论保持未解决状态。此外 PR body 自述了验证前捕获的一个同类型 bug：第一版 eager 漏斗把 2-D page_table 直接传给 translate_kv_loc_dense，而 page_size == 1 路径走 index_select 拒绝 2-D 索引——该 bug 只在默认配置下崩溃，作者以此论证解除 e2e 测试后端固定的必要性。

- full-prefill graph 路径缺少 dense remap (correctness): PR 内无回复与修复，问题遗留；建议后续 remap 该指针稳定的 prefill 表，或对 unified-memory + fa3 组合拒绝 full prefill capture。

风险与影响

- 风险： 1) full-prefill graph 缺口（正确性）：codex 机器人 P1 指出 cuda_graph_config.prefill.backend == Backend.FULL 组合未做 dense remap，虚拟物理分叉后会静默读错 KV 页，PR 内未修复。 2) 默认路径 shape 脆弱：translate_kv_loc_dense 的 index_select 拒绝 2-D 索引，而 page_size > 1 的高级索引能接受 2-D——同一类 bug 只在默认配置（ps=1）下暴露，说明默认路径缺少显式 shape 断言。 3) 内核签名扩展影响面：normal_decode_set_metadata 新增两个可选参数，除 fa3 外其它调用方靠默认值 None / 1 保持行为不变，pre-existing test_normal_decode_set_metadata.py（16 用例 + 8 子测试）回归通过。 4) 性能：enabled 时 Triton 内核多一次 v2p gather，静态池身份映射无开销，PR 未提供性能数据，属低风险。 5) 测试策略风险：nightly e2e 解除后端固定后依赖宿主默认解析（fa3），若未来默认后端变化可能导致夜间测试行为漂移——这是刻意取舍。
  - 影响：用户侧：H100/H200 上 Kimi-Linear + --enable-unified-memory 开箱即用，不再需要手工指定 --attention-backend triton；GSM8K 精度 0.917 vs 静态池 0.915，无精度回退。系统侧：fa3 成为 unified-memory MLA 池的第三个后端族，page-major allowlist 语义从『Triton-only + paged MLA 例外』扩展为『支持 dense 翻译的后端全集』；unified_mem_hooks 成为跨后端共享基础设施，后续新后端可循同一模式接入。团队侧：unified-memory 功能线完成默认配置闭环（triton → flashinfer/trtllm_mla → fa3），测试策略上确立了『不固定后端、暴露默认配置缺陷』的回归理念。

- 风险标记：full-prefill graph 组合未覆盖，默认配置路径 shape 脆弱，nightly e2e 解除后端固定，融合内核签名扩展

关联脉络

- PR #32971 [unified-memory] Support MLA-hybrid-Mamba (Kimi-Linear) on the Triton backend: 本 PR 的前置：让 unified-memory 在 Triton 后端跑通 Kimi-Linear 的 MLA 混合 Mamba 结构，fa3 支持是在此基础上的默认配置补齐。
- PR #32972 [unified-memory] Let Kimi-Linear use the paged MLA attention backends: 直接前驱：为 flashinfer / trtllm_mla 接入 dense 翻译；本 PR 是其 pre-Blackwell（fa3）对位补齐，且其 review 中发现的两个缺陷只在默认解析配置下可达，构成解除 nightly 测试后端固定的论据。