PR #34680 完整报告

sgl-project/sglang

[diffusion][Minimax H3]support subblock sparse attention on SM90

合并时间：2026-08-19 10:31

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34680

执行摘要

- 一句话：MiniMax-H3 SubBlock 稀疏注意力从 SM100 扩展到 SM90，Hopper 端到端提速最高 2×
- 推荐动作：值得精读。这是"同一路由方案、双 kernel 后端"的教科书式实现：重点看 (1) _get_subblock_sparse_attention_runner 的按设备一次性缓存分发，避免热路径开销；(2) _sm90_sparse_attention 中显式排序与 None full-block 的推理链条——为何排序是正确性的前提、为何 None 比空张量更优；(3) block_sparse_utils.py 的编译期守卫，区分"运行时值"与"编译期常量"是 CuTe DSL 调试的高频难点；(4) _tile_size_fwd_sm90 的 64×64 特例边界。Review 讨论本身也具有很高的学习价值，特别是对 kernel 内部消费顺序契约、compile key 隐式依赖的识别。

功能与动机

PR body 明确说明这是 #34148 的 follow-up：该 PR 为 MiniMax-H3 在 SM100 上引入了 SubBlock 稀疏注意力，本 PR 的目标是"extends the backend to SM90 GPUs using SGLang's CuTe-DSL block-sparse FlashAttention kernel without changing the existing SM100 path"。SM100 硬件成本高、可获得性受限，H100/H200 才是视频生成推理的主流部署机型，因此在 SM90 上复用仓库已有的 CuTe-DSL 稀疏 kernel 承载同一套 64×64 SubBlock 路由方案，可以在不改动 SM100 行为的前提下把稀疏加速的收益覆盖面显著扩大。

实现拆解

1. 统一分发入口（subblock_sparse_attn.py）：新增 _sm90_sparse_attention / _sm100_sparse_attention，分别封装 SM90 CuTe-DSL 稀疏 kernel 与 SM100 FlashInfer blk64 kernel；_get_subblock_sparse_attention_runner 用 functools.lru_cache(maxsize=None) 按 device 缓存架构判定（9.0 → SM90、10.0 → SM100、其余抛 RuntimeError）；_run_subblock_sparse_attention 成为 SubBlockSparseAttentionImpl._sparse_attention 的唯一出口，替换原先硬编码 load_bsa_attn_blk64_fwd 的 SM100 逻辑。
2. SM90 adapter 的排序语义（_sm90_sparse_attention）：显式对 q2k_block_index 做 sort(dim=-1) 升序排序。原因是 router 的公共契约允许 block id 任意顺序，而 SM90 稀疏消费者从最高槽位向下消费，且 mask_seqlen 只作用于第一个被消费的 block；升序排序保证最大 block id（可能的 ragged tail）永远落在被掩码的最高槽位。BlockSparseTensorsTorch 的 full_block_cnt/full_block_idx 传 None，因为 SubBlock 方

案没有 always-dense block，mask-only 形式让 full-block 分支在编译期整段消除。

3. SM90 kernel 配套（interface.py / block_sparse_utils.py）：_tile_size_fwd_sm90 新增 sparse_block_size_kv 参数，并在 head_dim == 128 且 Q/K sparse block 均为 64 时返回 FwdConfig(64, 64, True, True)（num_wg_mma == 1，本仓库首次编译运行的配置）；head_dim=96 等其他情况保持原 128/192 tile 选择。block_sparse_utils.py 的 producer/consumer 增加 const_expr(blocksparse_tensors.full_block_cnt is None) 守卫，因为 mask_empty 是运行时值、CuTe 仍会编译两条动态分支，不特判会在 full-list 一侧对 None 下标而编译失败。
4. 平台 resolver（platforms/cuda.py）：_SubBlockSparseAttentionBackendResolver 从 required_capability = (10, 0) 改为 supported_capabilities = {(9, 0), (10, 0)}，并按架构分别校验依赖：SM90 检查 CuTe-DSL/Quack 导入，SM100 检查 FlashInfer blk64；10.3/12.x 仍保持 fail-closed。capability 先转 tuple 再查集合，避免对 None 取 major 的隐患。
5. 测试与文档：test/registered/cpu/test_subblock_sparse_attention.py（新增）验证分发缓存只解析一次、SM100 分发、10.3 拒绝；test_subblock_sparse_sm90.py（新增）验证 64×64 tile 选择仅在 head_dim=128 生效；test_subblock_sparse_attention.py 把 requires_sm100 泛化为 requires_subblock_kernel，SM90/SM100 数值测试共用，并新增 adapter 排序断言测试与 test_unsorted_ragged_tail_oversubscribes_sms（516 个 query-head tile、跨多波次、故意乱序）以及随机路由负对照；原仅在 SM100 CI 上跑的回归测试现在会落到 1-gpu-h100。README 与模块 docstring 同步为双路径描述。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/attention/backends/subblock_sparse_attn.py（模块 注意力后端；类别 source；类型 core-logic；符号 SubBlockSparseAttentionBackend, _load_sm90_block_sparse_attention, _sm90_sparse_attention, _sm100_sparse_attention）：PR 核心：新增 SM90/SM100 双路径分发与 SM90 adapter。_get_subblock_sparse_attention_runner 按计算能力 9.0/10.0 每设备缓存一次选择 kernel；_sm90_sparse_attention 完成 block id 显式排序与 mask-only BlockSparseTensorsTorch 构造，是 SM90 正确性的关键逻辑所在。
- python/sglang/kernels/ops/attention/flash_attn/cute/interface.py（模块 SM90 内核；类别 infra；类型 core-logic；符号 _tile_size_fwd_sm90, _flash_attn_fwd）：SM90 tile 选择的关键改动：_tile_size_fwd_sm90 新增 sparse_block_size_kv 参数，并在 head_dim == 128 的 64×64 sparse block 场景返回 FwdConfig(64, 64, True, True)，这是该配置首次被编译运行；同时修复 _flash_attn_fwd 中 sparse_kv 变量遮蔽 compile key 的风险。
- python/sglang/kernels/ops/attention/flash_attn/cute/block_sparse_utils.py（模块 SM90 内核；类别 infra；类型 core-logic；符号 produce_block_sparse_loads, consume_block_sparse_loads）：mask-only 稀疏模式能否编译运行的关键：producer/consumer 需要编译期守卫，避免 full-block 列表为 None 时 CuTe 对 None 下标导致编译失败；守卫键统一为 full_block_cnt 以保证与上游一致性。
- python/sglang/multimodal_gen/runtime/platforms/cuda.py（模块 设备解析；类别 source；类型 configuration；符号 _SubBlockSparseAttentionBackendResolver）：启动期 fail-closed 策略的调整点：resolver 从仅接受 10.0 改为接受 9.0/10.0，并按架构分别

校验 CuTe-DSL 或 FlashInfer 依赖，10.3/12.x 仍拒绝。这是用户能否在 H100/H200 上启动 subblock_sparse_attn 的入口。

- python/sglang/multimodal_gen/test/unit/test_subblock_sparse_attention.py（模块 单元测试；类别 test；类型 test-coverage；符号 _subblock_kernel_available, test_sm90_adapter_sorts_indices_and_uses_64x64_blocks, test_unsorted_ragged_tail_oversubscribes_sms, _FakeBlockSparseTensors）：测试主战场：requires_sm100 泛化为 requires_subblock_kernel 让原 SM100 专用数值测试（含 ragged tail 复现、随机路由负对照）落到 H100 CI；新增 adapter 排序断言测试与乱序超订多波次测试，把排序失效从静默错误变成可观测失败。
- test/registered/cpu/test_subblock_sparse_attention.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestSubBlockSparseAttentionDispatch, test_dispatch_is_resolved_once_per_device, test_dispatches_sm100, test_rejects_unsupported_compute_capability）：新增的 CPU 注册测试，验证 _get_subblock_sparse_attention_runner 的缓存语义（每设备只解析一次）、SM100 分发与 10.3 拒绝路径，把分发正确性变成无 GPU 依赖的确定性回归。
- test/registered/kernels/ops/attention/test_subblock_sparse_sm90.py（模块 内核测试；类别 test；类型 test-coverage；符号 TestSubBlockSparseSM90, test_64x64_routing_mask_uses_matching_compute_tile, test_64x64_special_case_is_limited_to_head_dim_128）：新增的 SM90 kernel 回归测试，固定 64×64 tile 选择只在 head_dim=128 生效、head_dim=96 仍保持 128×128 的既有结论，防止未来 tile 选择逻辑回归。
- python/sglang/multimodal_gen/runtime/layers/attention/backends/subblock_sparse/README.md（模块 文档；类别 docs；类型 documentation）：文档同步：从“10.0 only”改为“9.0 or 10.0”，明确 H100/H200 用 CuTe-DSL、B200/GB200 用 FlashInfer，并保留 10.3/12.x fail-closed 的说明。

关键符号：_run_subblock_sparse_attention, _get_subblock_sparse_attention_runner, _sm90_sparse_attention, _sm100_sparse_attention, _load_sm90_block_sparse_attention, _tile_size_fwd_sm90, produce_block_sparse_loads, consume_block_sparse_loads, _SubBlockSparseAttentionBackendResolver.resolve

关键源码片段

python/sglang/multimodal_gen/runtime/layers/attention/backends/subblock_sparse_attn.py

PR 核心：新增 SM90/SM100 双路径分发与 SM90 adapter。_get_subblock_sparse_attention_runner 按计算能力 9.0/10.0 每设备缓存一次选择 kernel；_sm90_sparse_attention 完成 block id 显式排序与 mask-only BlockSparseTensorsTorch 构造，是 SM90 正确性的关键逻辑所在。

```
# 按 CUDA 设备做一次架构判定并缓存结果，避免每次前向都走 capability 查询。
@functools.lru_cache(maxsize=None)
def _get_subblock_sparse_attention_runner(device: torch.device):
    """Resolve the architecture-specific kernel once per CUDA device."""
```

```
    capability = torch.cuda.get_device_capability(device)
    if capability == (9, 0):
        return _sm90_sparse_attention # Hopper 走 SGLang CuTe-DSL kernel
    if capability == (10, 0):
        return _sm100_sparse_attention # Blackwell 走 FlashInfer blk64 kernel
    raise RuntimeError(
        "SubBlock sparse attention supports compute capability 9.0 or 10.0; "
        f"this tensor is on a {capability[0]}.{capability[1]} device."
    )


def _run_subblock_sparse_attention(
    q: torch.Tensor, k: torch.Tensor, v: torch.Tensor,
    q2k_block_index: torch.Tensor, topk: int, softmax_scale: float,
) -> torch.Tensor:
    """把同一份 64x64 路由方案分发给 Hopper 或 Blackwell 的实现。"""
    runner = _get_subblock_sparse_attention_runner(q.device)
    return runner(q, k, v, q2k_block_index, topk, softmax_scale)


def _sm90_sparse_attention(
    q: torch.Tensor, k: torch.Tensor, v: torch.Tensor,
    q2k_block_index: torch.Tensor, topk: int, softmax_scale: float,
) -> torch.Tensor:
    """用 SGLang 已有的 SM90 CuTe 稀疏 kernel 执行 SubBlock 路由方案。"""
    BlockSparseTensorsTorch, flash_attn_func = _load_sm90_block_sparse_attention()

    # router 契约允许 block id 任意顺序，而 SM90 稀疏消费者从最高槽位向下消费，
    # 且 mask_seqlen 只作用于第一个被消费的 block。显式升序排序保证可能的 ragged
    # tail（最大 block id）永远落在被掩码的最高槽位，不依赖 fused top-k kernel
    # 当前恰好升序输出的内部行为。
    ordered_index = q2k_block_index.sort(dim=-1).values
    block_counts = torch.full(
        ordered_index.shape[:-1], topk, dtype=torch.int32, device=ordered_index.device,
    )
    sparse_tensors = BlockSparseTensorsTorch(
        mask_block_cnt=block_counts,
        mask_block_idx=ordered_index,
        # SubBlock 方案没有 always-dense block，full 侧两个张量传 None。
        # block_sparse_broadcast_pattern 会把 None 原样保留在 broadcast 模式里，
        # 而该模式参与 compile key，因此 mask-only 与 mask+full 调用不会复用
        # 同一个已编译 kernel——这就是传 None 安全的依据。
        full_block_cnt=None,
        full_block_idx=None,
        block_size=(SUBBLOCK_SPARSE_BLOCK_SIZE, SUBBLOCK_SPARSE_BLOCK_SIZE),
    )
    out, _ = flash_attn_func(
        q, k, v,
        softmax_scale=softmax_scale,
```

```
        causal=False,
        num_splits=1,
        block_sparse_tensors=sparse_tensors,
    )
    return out
```

python/sglang/kernels/ops/attention/flash_attn/cute/interface.py

SM90 tile 选择的关键改动：_tile_size_fwd_sm90 新增 sparse_block_size_kv 参数，并在 head_dim == 128 的 64×64 sparse block 场景返回 FwdConfig(64, 64, True, True)，这是该配置首次被编译运行；同时修复 _flash_attn_fwd 中 sparse_kv 变量遮蔽 compile key 的风险。

```
def _tile_size_fwd_sm90(
    head_dim, head_dim_v, is_causal, is_local,
    sparse_block_size_q=None, sparse_block_size_kv=None,
):
    """SM90 forward 的 tile 选择；出现 sparse block 时计算 tile 必须贴合 mask 两个轴。"""
    # SubBlock 场景特例：每个 64 行 query block 拥有自己独立路由的 64 行 KV
    # block 列表，因此不能像常规 sparse mask 那样粗化成 128x128 tile——
    # 那样会把某一行 query 的 mask 错误套到相邻行上。注意 FwdConfig(64, 64, True, True)
    # 对应 num_wg_mma == 1，是本仓库首次被编译运行的配置，head_dim == 128 是
    # 额外硬约束，避免让从未使用 tile_m=64 的 192/256 head_dim 静默继承该配置。
    if (
        head_dim == 128
        and sparse_block_size_q == 64
        and sparse_block_size_kv == 64
    ):
        return FwdConfig(64, 64, True, True)
    # 其余 sparse mask 只要求 tile_m 整除 sparse_block_size_q；head_dim <= 96
    # 时优先用 192，不兼容再回退 128（沿用 C++ tile_size.h 的既有策略）。
    if head_dim <= 64:
        ...
```

python/sglang/kernels/ops/attention/flash_attn/cute/block_sparse_utils.py

mask-only 稀疏模式能否编译运行的关键：producer/consumer 需要编译期守卫，避免 full-block 列表为 None 时 CuTe 对 None 下标导致编译失败；守卫键统一为 full_block_cnt 以保证与上游一致性。

```
# produce_block_sparse_loads 中的 mask-only 特判分支。normalize 保证 full 的
# count 与 index 要么同时存在、要么同时为 None；mask_empty 是运行时值，CuTe
# 仍会编译两条动态分支，因此若不在此处特判，full-list 一侧会对 None 做下标
# 运算而直接编译失败——这里不是性能优化，而是可编译性的前提。
if const_expr(blocksparse_tensors.full_block_cnt is None):
    kv_producer_state = load_block_list(
        curr_mask_block_idx,
        mask_begin,
        mask_end,
        first_block_preloaded=False,
        kv_producer_state=kv_producer_state,
```

```
            load_K=load_K,
            load_V=load_V,
            pipeline_k=pipeline_k,
            pipeline_v=pipeline_v,
            intra_wg_overlap=intra_wg_overlap,
        )
        if const_expr(intra_wg_overlap) and not mask_empty:
            kv_producer_state = finish_overlap_v_load(
                curr_mask_block_idx,
                mask_begin,
                mask_end,
                load_V,
                pipeline_v,
                kv_producer_state,
            )
        return kv_producer_state

    # consumer 侧的守卫统一用 full_block_cnt 做编译期存在性信号，与
    # get_curr_blocksparse_tensors 保持一致；normalize_block_sparse_tensors 已保证
    # full 的 count/index 成对出现，所以两个字段不可能在 fixed-length 或 varlen
    # 布局中出现分歧。
    if (
        const_expr(blocksparse_tensors.full_block_cnt is not None)
        and split_full_block_cnt > 0
    ):
        full_n_block = curr_full_block_idx[full_end - 1]
```

评论区精华

核心 review 由 triple-mu 主导，整体设计获得确认："On SM90 q_stage = 1，加上 block_sparsity.py 的两处检查，(64, 64) 是 64 行路由粒度下唯一能通过的 tile 组合；升序排序确实必要，因为 block_sparse_utils.py 从最高槽位向下消费，mask_seqlen 只作用于第一个被消费的 block。" 主要交锋点：

- sort 注释与实现不符：triple-mu 指出 _topk_kernel 实际已升序输出，sort 是防御性的而非纠正性的，建议保留 sort 但改写注释，避免依赖 Triton kernel 未文档化的内部行为；作者采纳并改写。
- full-block 传 None 而非空张量：triple-mu 论证 block_sparse_utils.py:73-79 显式支持 None，可让整个 full-block 分支编译期消除并省掉两次张量分配；作者采纳，并因此暴露并修复了 producer/consumer 对 None 的编译期访问。
- 变量遮蔽风险：_flash_attn_fwd 中新增 sparse_kv 局部变量与原布尔 sparse_kv（参与 compile key）同名，triple-mu 警告"900 行函数里一个名字两个含义"会静默污染缓存键；作者改名 sparse_block_size_kv。
- 64x64 特例作用域：triple-mu 指出特例分支在 head_dim 分支之前，会匹配到从未使用 tile_m=64 的 192/256 head_dim；作者加上 head_dim == 128 限制并补回归测试。

- 守卫键一致性：consumer 守卫原来挂在 curr_full_block_idx is not None，与上游 get_curr_blocksparse_tensors 的 full_block_cnt is not None 只在“同时置空”时成立；作者改为统一用 full_block_cnt，并确认 normalize_block_sparse_tensors 已保证 count/index 成对出现。
- 测试覆盖缺口：triple-mu 指出 test_varlen_routes_each_document 两个文档单波次即完成，排序失效会静默通过；作者用新的乱序超订测试补上了该场景。
  - SM90 adapter 排序的必要性与注释准确性 (correctness): 保留显式 sort，注释改为“router 公共契约允许任意顺序，SM90 consumer 要求最大 block id 落在最高槽位，排序不依赖 top-k kernel 的内部输出顺序”。
  - full-block 传 None 优于空张量及编译期裁剪 (design): full_block_cnt/full_block_idx 均传 None，并在 CuTe producer/consumer 增加 const_expr(... is None) 特判；后续评论澄清该守卫是编译前提而非性能优化。
  - sparse_kv 局部变量遮蔽 compile key 布尔值 (style): 作者重命名为 sparse_block_size_kv，规避歧义。
  - 64x64 tile 特例的作用域限制 (design): 作者加 head_dim == 128 硬约束，并新增 head_dim=96 回归测试确认仍走 128×128。
  - consumer 编译期守卫键统一为 full_block_cnt (correctness): 作者统一改为 blocksparse_tensors.full_block_cnt，并说明 normalize_block_sparse_tensors 已保证 count/index 成对出现。
  - varlen 测试无法暴露排序失效 (testing): 作者新增 test_unsorted_ragged_tail_oversubscribes_sms：516 个 query-head tile、跨多波次、故意把 ragged block 放在最低槽位，用零 Q/K + 单位 V 使输出幅度偏差无法被 assert_close 忽略。
  - 模块 docstring 与 README 遗漏 FlashInfer-only 描述 (documentation): 作者在 follow-up commit 8133e9f7 中更新 docstring 并补上 None 安全的依据注释（broadcast pattern 保留 None 且参与 compile key）。

风险与影响

- 风险：
  - 首次编译运行的新 kernel 配置：reviewer 明确警告 FwdConfig(64, 64, True, True) 对应 num_wg_mma == 1，此前 _tile_size_fwd_sm90 只产出 128/192 且无 caller 传 tile_mn，该配置在本仓库从未被编译运行过。作者通过让原有 test_full_budget_reproduces_dense、test_ragged_tail_reproduces_dense（8192+37，正是排序保护的目标路径）等数值测试落到 H100 CI 上缓解，但 64×64 tile 在真实负载下的 kernel 稳定性仍需关注。
  - mask-only 与 compile key 的隐式耦合：mask-only 调用不与 mask+full 复用编译 kernel，依赖 block_sparse_broadcast_pattern 把 None 保留进 compile key 的隐含行为，而 compile_key 本身没有 full-block 位；未来若有人“顺手”把 None 改成空张量，会导致错误 kernel 被静默复用。PR 内已用注释记录该依赖，但仍是结构性脆弱点。
  - SM90 消费顺序契约：排序正确性建立在“从高槽位向下消费 + 首个 block 被掩码”的 kernel 内部约定上，上游 kernel 若改消费方向，排序保护即失效；外部文档并未固化这一契约。

    - CI 未真正运行：作者与 reviewer 都明确指出 approval 时缺 run-ci 标签，84 个测试 job 全部跳过；BBuf 最后执行了 /rerun-failed-ci，但合并前完整 CI 结果未被确认。
    - 900 行大函数集中改动：interface.py 的 _flash_attn_fwd 是超长函数，本次新增 sparse_block_size_kv 读取与 tile 选择调用，变量遮蔽问题虽已修复，但后续维护仍需谨慎。
- 影响：
    - 用户侧：H100/H200 上部署 MiniMax-H3 的用户可直接使用 --attention-backend subblock_sparse_attn，端到端提速 1.26x-2.01x（长视频收益更大：15 s Ref2AV 达 2.01x），SSIM 0.82-0.98 的可接受精度损失换得显著吞吐提升。
    - 系统侧：sglang/kernels 的 SM90 CuTe-DSL block-sparse 路径新增 mask-only 模式与 64x64 tile 配置，所有使用该 kernel 的调用都会受影响；但 reviewer 确认“(64, 64) mask 在 SM90 上此前会在 normalize_block_sparse_config 直接报错，没有既有 caller 受影响”，因此向后兼容性有保障。
    - 团队侧：SM100 路径逐字保留，双架构共享同一套 SubBlock router 与调度配置，后续架构扩展（如 SM100a 变体）只需在 _get_subblock_sparse_attention_runner 增加分支；此前被跳过的 SM90 数值测试现在进入 H100 CI，提升了回归覆盖质量。
    - 风险标记：首次编译 FwdConfig(64,64) 配置，mask-only compile key 隐式依赖，CI 未在合并前完整运行，SM90 消费顺序契约未固化，900 行大函数集中改动

关联脉络

- PR #34148 引入 SM100 SubBlock 稀疏注意力（PR body 直接引用）：本 PR 明确声明是其 follow-up：把同一 SubBlock 路由方案从 SM100 扩展到 SM90，SM100 路径逐字保留。
- PR #34581 [Diffusion] Optimizing MiniMax-H3 for consumer-level GPUs: INT8 Linear + pluggable DiT attention backends: 同一 MiniMax-H3 diffusion 优化线：引入可插拔 DiT attention backends 体系，本 PR 的 subblock_sparse_attn 正是该体系下新增的 SM90 支持。
- PR #35339 [diffusion] Per-request lossy accelerations: Cache-DiT, CFG gating, attention backend override: 后续演进：把有损加速改为按请求开关并支持 attention backend 覆盖，与 SubBlock 这类有损注意力共同构成 MiniMax-H3 感知质量 / 吞吐权衡的完整能力栈。