PR #29677 完整报告

sgl-project/sglang

[AMD] perf: compact Triton extend-attention for ragged prefill (AMD/HIP-only)

合并时间：2026-08-07 05:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29677

执行摘要

- 一句话：AMD 扩展注意力紧凑网格化，混合 prefill 吞吐翻倍
- 推荐动作：值得精读。三个最值得关注的设计决策：(1) 用“与 flash-attn varlen 启动方式对齐”而非“发明新技术”的框架论证优化合理性，降低 reviewer 判断风险的成本；(2) BLOCK_M 从 kernel 自动探测并沿 backend → scheduler 传递，避免用户可感知配置项与潜在错配；(3) AMD-only + 默认开启 + 可 env 关闭的分层策略，在跨厂商风险与收益之间做了务实取舍。建议工程团队在引入同类平台特定优化时复用该模式；后续可关注 compact grid 是否被推广到 CUDA/NPU。

功能与动机

PR body 明确指出：legacy grid 是矩形且按 max_len_extend 铺开，在混合 batch 中“every short decode/partial-prefill row pays tile work proportional to the longest extend row”，造成大量空转 tile。而 flash-attn varlen 内核（aiter 后端）通过 cu_seqlens 做 ragged-aware 调度，工作量与实际 query tile 数成正比，因此“this closes a triton-vs-flash-attn gap, it is not a new technique”。同时 PR 强调 ragged mixed-prefill 是生产常态（不同 prompt 长度共存、mixed chunking 把 decode 行与长 prefill chunk 放同一 attention call），这是 AMD 上真实 serving 性能瓶颈。

实现拆解

本 PR 按以下 5 步落地：

1. 内核层：新增 compact query-tile grid（python/sglang/kernels/ops/attention/extend_attention.py）。新增 _compact_extend_q_tiles_per_head()，对 batch 内每个请求计算 cdiv(extend_len_i, BLOCK_M) 后求和得到真实 tile 数；当 compact_tiles >= legacy_tiles（如长度均匀的 batch）时返回 None，触发退回 legacy 矩形 grid。extend_attention_fwd() 在 launch 时读取 SGLANG_TRITON_COMPACT_EXTEND_ATTENTION 环境变量（运行期读取而非 import 时，便于测试用 temp_set_env 切换），并只在 _is_hip 为真时进入 compact 分支，grid 由 3D 矩形变为 (compact_q_tiles, head_num) 的 2D 结构。_fwd_kernel 新增 USE_COMPACT_TILE_GRID constexpr 分支，通过 qo_indptr 做 prefix-sum 线性扫描把一维 tile id 反查回 (cur_seq, cur_block_m)；该 O(batch_size) 标量循环相比省下的整块 tile 计算代价极低。
2. Backend 接线与 BLOCK_M 自动探测（triton_backend.py + scheduler.py）。TritonAttnBackend.forward_extend() 把 forward_batch.extend_seq_lens_cpu（逐请求

实际 extend 长度）透传给 extend_attention_fwd()，compact 路径依赖它做精确 tile 计算。同时 TritonAttnBackend.__init__() 调用 _get_block_sizes_for_extend_attention(head_di m, head_dim) 自动探测 kernel 实际使用的 BLOCK_M 并暴露为 extend_attention_block_m；scheduler.py 在构造 PrefillAdder 时把该值传入 prefill_tile_block_m。这消除了初版 SGLANG_PREFILL_TILE_BLOCK_M 固定默认 64 的隐患——DeepSeek MLA 的 head_dim=192 时 kernel 实际用 128，固定默认值会导致调度估算与真实 kernel 错配。

3. 调度器配套：tile-budget admission（schedule_policy.py，默认关闭）。新增 estimate_prefill_extend_tile_metrics() 与 _check_prefill_tile_budget()，在非 chunked、chunked、DLLM 三条 add 路径插入检查：当候选请求使 batch 的 compact（或 legacy）q_tiles_per_head 超过 SGLANG_PREFILL_TILE_BUDGET 时返回 AddReqResult.OTHER 停止接纳。_IS_HIP 与默认 budget=0 双重门控，非 AMD 平台完全 no-op；test_prefill_tile_budget_always_allows_first_request 保证首个请求永远放行，避免冷启动死锁。
4. 正确性配套：MLA split-KV verify 快路径拒绝（verify_splitkv.py）。在 can_handle() 中新增 q_head_dim != v_head_dim 拒绝分支：MLA（如 DeepSeek 576 vs 512）使用共享 latent KV 与 absorbed-attention 布局，split-KV verify 内核不支持该形状会 GPU fault，现在直接退回 extend_attention_fwd()。该改动来自本 PR 演进中用 DeepSeek 模型实测暴露的问题。
5. 测试与 CI。新增 2 个 kernel 测试（compact tile 计数、compact vs legacy 数值等价性 test_extend_attention_compact_grid）、4 个调度测试（估算、admission 语义、首请求放行、非 HIP no-op）和 1 个 verify fallback 测试，全部注册进 base-b/CUDA 与 stage-b AMD 套件。PR body 给出 GSM8K 精度 0.940（>0.94 通过，与 AITER 参考 0.946 相当）、3 种 serving 工况端到端吞吐对比，以及 6 个模型家族的 kernel 微基准（mixed 场景 11.6x~14.9x、ragged 场景 4.7x~5.8x，且逐位精确 max abs diff = 0.0）。

关键文件：

- python/sglang/kernels/ops/attention/extend_attention.py（模块 注意力内核；类别 infra；类型 infrastructure；符号 _compact_extend_q_tiles_per_head）：核心 kernel 变更：新增 compact tile 计数函数与 2D compact grid launch，_fwd_kernel 内新增 tile 反查分支，是性能收益的根源。
- python/sglang/srt/managers/schedule_policy.py（模块 调度策略；类别 source；类型 core-logic；符号 _ceil_div, estimate_prefill_extend_tile_metrics, _admitted_extend_lens, _tile_admission_metric_key）：调度器核心 admission 逻辑：新增 tile-budget 估算与检查，控制 prefill batch 的 tile 构成，是让 kernel 优化在系统层面生效的配套。
- python/sglang/srt/layers/attention/triton_backend.py（模块 注意力后端；类别 source；类型 dependency-wiring）：把 forward_batch.extend_seq_lens_cpu 透传给 extend_attention_fwd，并在 __init__自动探测 extend_attention_block_m 供调度器使用。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic）：构造 PrefillAdder 时从 attention backend 读取 BLOCK_M 并传入，打通自动探测链路。

- python/sglang/srt/environ.py（模块 环境配置；类别 source；类型 core-logic）：新增 SGLANG_TRITON_COMPACT_EXTEND_ATTENTION（默认 True）、SGLANG_PREFILL_TILE_BUDGET（默认 0）等环境变量定义，集中管理公共路径的配置。
- test/registered/attention/test_triton_attention_kernels.py（模块 内核测试；类别 test；类型 test-coverage；符号 test_compact_extend_attention_tile_count, test_extend_attention_compact_grid）：新增 compact tile 计数与 compact vs legacy 数值等价性测试，后者是会真实执行新分支的关键回归用例。
- test/registered/unit/managers/test_prefill_adder.py（模块 调度测试；类别 test；类型 test-coverage；符号 _adder_with_extend_lens, test_estimate_prefill_extend_tile_metrics, test_compact_prefill_tile_budget_admits_more_than_legacy, test_prefill_tile_budget_always_allows_first_request）：覆盖调度器 tile-budget admission 的四个行为面：估算、compact 优于 legacy、首请求放行、非 HIP no-op。
- python/sglang/kernels/ops/attention/verify_splitkv.py（模块 验证内核；类别 infra；类型 infrastructure）：can_handle 新增 MLA head_dim != v_head_dim 拒绝，避免 split-KV verify 快路径 GPU fault，与本 PR 的 DeepSeek 实测直接相关。
- test/registered/attention/test_verify_splitkv.py（模块 验证内核；类别 test；类型 test-coverage；符号 test_fallback_mla_head_dim_mismatch）：新增 test_fallback_mla_head_dim_mismatch 锁定 MLA 形状下 can_handle 拒绝行为，防止 GPU fault 回归。

关键符号：_compact_extend_q_tiles_per_head, extend_attention_fwd, estimate_prefill_extend_tile_metrics, _check_prefill_tile_budget, _admitted_extend_lens, _candidate_tile_metrics, _tile_admission_metric_key, can_handle, test_extend_attention_compact_grid, test_compact_prefill_tile_budget_admits_more_than_legacy

关键源码片段

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

核心 kernel 变更：新增 compact tile 计数函数与 2D compact grid launch，_fwd_kernel 内新增 tile 反查分支，是性能收益的根源。

```
def _compact_extend_q_tiles_per_head(
    *,
    batch_size: int,
    max_len_extend: int,
    total_extend_tokens: int,
    block_m: int,
    extend_seq_lens_cpu=None,
) -> int | None:
    # legacy 矩形 grid 的 tile 数是 batch_size * cdiv(max_len_extend, BLOCK_M),
    # 在 ragged mixed batch 里每个短请求行都要按最长行付 tile 开销。而
    # flash-attn varlen 内核（aiter 后端）通过 cu_seqlens 做 ragged-aware 调度,
    # 工作量只与实际 query tile 数成正比。这里就是在补齐 triton 与 flash-attn
    # 的这个启动方式差距，而不是发明新技术。
```

```
    if batch_size <= 1 or max_len_extend <= 0:
        return None # 单请求或空 batch 没有压缩空间，保持 legacy grid

    legacy_tiles = batch_size * triton.cdiv(max_len_extend, block_m)
    if legacy_tiles <= 0:
        return None

    if extend_seq_lens_cpu is not None:
        if isinstance(extend_seq_lens_cpu, torch.Tensor):
            extend_seq_lens_cpu = extend_seq_lens_cpu.tolist()
        if len(extend_seq_lens_cpu) < batch_size:
            return None # 长度信息不完整时退回 legacy，避免错误 grid
        compact_tiles = sum(
            triton.cdiv(max(0, int(extend_seq_lens_cpu[i])), block_m)
            for i in range(batch_size)
        )
    else:
        # 退化路径：拿不到逐请求长度时，若长度均匀则直接放弃压缩
        if total_extend_tokens == batch_size * max_len_extend:
            return None
        compact_tiles = (total_extend_tokens + batch_size * (block_m - 1)) // block_m

    # 只有真正减少 launch 工作量时才切换，否则继续用矩形 grid
    if compact_tiles <= 0 or compact_tiles >= legacy_tiles:
        return None
    return int(compact_tiles)

# 仅在 HIP 平台启用，且允许环境变量覆盖（0 强制关闭，1 强制打开）。
# 非 AMD 平台完全保持 legacy 矩形 grid，保证 CUDA 路径零改动。
use_compact_tile_grid = _is_hip and envs.SGLANG_TRITON_COMPACT_EXTEND_ATTENTION.
get()
compact_q_tiles = None
if use_compact_tile_grid:
    compact_q_tiles = _compact_extend_q_tiles_per_head(
        batch_size=batch_size,
        max_len_extend=max_len_extend,
        total_extend_tokens=q_extend.shape[0],
        block_m=BLOCK_M,
        extend_seq_lens_cpu=extend_seq_lens_cpu,
    )
use_compact_tile_grid = compact_q_tiles is not None
if use_compact_tile_grid:
    # 2D grid：每个 program 处理一个真实 query tile，tile 到 (seq, block) 的
    # 映射由 kernel 内部通过 qo_indptr 的 prefix-sum 反查完成
    grid = (compact_q_tiles, head_num)
else:
    grid = (batch_size, head_num, triton.cdiv(max_len_extend, BLOCK_M))
```

python/sglang/srt/managers/schedule_policy.py

调度器核心 admission 逻辑：新增 tile-budget 估算与检查，控制 prefill batch 的 tile 构成，是让 kernel 优化在系统层面生效的配套。

```
def estimate_prefill_extend_tile_metrics(
    extend_lens: List[int], block_m: int
) -> Dict[str, Union[int, float, List[int], None]]:
    # 估算一个 prefill batch 的 extend-attention query tile 数。
    # legacy 按最长请求铺矩形 grid，compact 按每个请求实际长度求和，
    # 两者差值就是调度器可用来做 admission 决策的收益空间。
    normalized_lens = [max(0, int(length)) for length in extend_lens]
    q_tiles = [
        _ceil_div(length, block_m) if length > 0 else 0 for length in normalized_lens
    ]
    legacy_tiles = len(q_tiles) * max(q_tiles) if q_tiles else 0
    compact_tiles = sum(q_tiles)
    saved_tiles = legacy_tiles - compact_tiles
    saved_ratio = saved_tiles / legacy_tiles if legacy_tiles else None
    return {
        'block_m': int(block_m),
        'request_count': len(normalized_lens),
        'extend_lens': normalized_lens,
        'q_tiles_per_request': q_tiles,
        'max_extend_len': max(normalized_lens) if normalized_lens else 0,
        'sum_extend_len': sum(normalized_lens),
        'legacy_q_tiles_per_head': legacy_tiles,
        'compact_q_tiles_per_head': compact_tiles,
        'saved_q_tiles_per_head': saved_tiles,
        'saved_q_tile_ratio': saved_ratio,
    }


def _check_prefill_tile_budget(
    self, candidate_extend_len: int
) -> Optional[AddReqResult]:
    # AMD-only：非 HIP 平台即使设置了 env budget 也保持原调度行为。
    if not _IS_HIP or PREFILL_TILE_BUDGET <= 0:
        return None

    # 第一个请求永远放行，避免冷启动时把初始请求挡在门外。
    if not self.can_run_list:
        return None

    metrics = self._candidate_tile_metrics(candidate_extend_len)
    candidate_metric = int(metrics.get(self._tile_admission_metric_key()) or 0)

    # compact 模式统计真实 query tile 数，同样的 budget 能接纳更多短请求；
    # legacy 模式按矩形 grid 估算则更容易触顶拒绝。
    if candidate_metric <= PREFILL_TILE_BUDGET:
        return None
```

```
    return AddReqResult.OTHER
```

评论区精华

Review 中最有价值的交锋集中在 5 点：

- CUDA 路径零改动是硬约束：hubertlu-tw 在早期 diff（把 CUDA Hopper 的 Lq <= 128 分支放宽到 <= 256）上直接要求 “Please make sure that CUDA's code path is not changed.” 最终实现将所有新逻辑收敛到 _is_hip/_IS_HIP 门控之后，CUDA 套件全绿作为确认。
- 公共路径全局变量最小化：yichiche 提出 “In common paths ... please minimize the use of global variables. If they are truly needed, put them in environ.py.” 最终 SGLANG_TRITON_COMPACT_EXTEND_ATTENTION、SGLANG_PREFILL_TILE_BUDGET、SGLANG_PREFILL_TILE_BUDGET_MODE 全部收敛到 environ.py 的 Envs 类。同时 yichiche 质疑 USE_COMPACT_TILE_GRID 参数是否必要，该参数因 compact 分支需要 kernel 内逻辑而保留。
- “硬件无关 vs AMD-only” 的设计张力：valechen 回应 yichiche：“I believe this optimization is generic, hardware-agnostic, and not library-dependent. But I was also told to make it AMD only change. What do you recommend?” 最终拍板保持 AMD-only（PR body 明确 “to avoid impacting other vendors”），但默认开启而非 opt-in——既规避跨厂商回归风险，又让 AMD 用户无感受益。
- Flag 负担与 BLOCK_M 自动探测：HaiShaw 建议 “make some of these FLAGS to tunable kargs, or encode them for different arch ... Leaving to UI increases the user burden”。valechen 的回应是删掉 SGLANG_PREFILL_TILE_BLOCK_M 这个用户 knob，改为从 kernel 自动探测并沿 backend → scheduler → PrefillAdder 链路传递——这是最值得学习的一处收敛。
- HIP CI 覆盖缺口被追平：amd-bot 两次报告“核心代码未被任何 PR-CI 测试真正执行”（早期 rate limit 与 gate 超时导致 AMD 矩阵被跳过），并指出唯一验证新分支的测试只有 HIP 上有意义；最终 michaelzhang-ai 确认 AMD 全矩阵在真实 MI300/MI35X 硬件跑通，4 个相关测试全部通过，闭合了“CUDA 绿但 HIP 未验”的缺口。
  - CUDA 路径保持零改动 (correctness): 最终所有新增逻辑都收敛到 _is_hip / _IS_HIP 门控之后，CUDA 套件全绿确认零改动。
  - 公共路径全局变量最小化 (design): env 定义全部收敛到 environ.py 的 Envs 类；USE_COMPACT_TILE_GRID 因 compact 分支需要 kernel 内逻辑而保留。
  - 优化本质硬件无关但被限定 AMD (design): 拍板保持 AMD-only 并默认开启，避免影响其他厂商；为后续泛化留下伏笔。
  - Flag 负担与 BLOCK_M 自动探测 (design): 删除 SGLANG_PREFILL_TILE_BLOCK_M，改为从 _get_block_sizes_for_extend_attention() 自动探测并沿 backend → scheduler 传递；保留 SGLANG_PREFILL_TILE_BUDGET（默认 0= 禁用）供实验。
  - HIP CI 覆盖缺口 (testing): AMD 全矩阵真实运行，_is_hip=True 分支被真实执行，覆盖缺口关闭。

风险与影响

- 风险：
  1. AMD 默认开启的回归面：SGLANG_TRITON_COMPACT_EXTEND_ATTENTION 默认 True，AMD 用户升级即走新路径。虽然 GSM8K 精度与 AITER 参考对齐、微基准逐位精确，但真实模型覆盖有限（DeepSeek-R1-MXFP4、Qwen2.5-7B），新增 kernel 分支与 tile 反查对 sliding window、sinks、page_size 等组合的数值等价性主要依赖单元测试。
  2. 调度 admission 语义改变：_check_prefill_tile_budget 引入后，一旦用户开启 SGLANG_PREFILL_TILE_BUDGET，prefill batch 构成会改变（多短请求、少长请求），可能影响长请求排队延迟；且它依赖 prefill_tile_block_m 与 kernel 实际 BLOCK_M 一致，自动探测链路任何一环失效都会造成估算偏差。目前默认关闭，风险可控。
  3. kernel 内反查循环：compact 分支在 kernel 内做 O(batch_size) 标量 while 循环，极端小 batch 下可能得不偿失，但有 compact_tiles >= legacy_tiles 的 fallback 兜底；该反查只读 qo_indptr 不触发设备同步，无 graph capture 风险。
  4. 模块级 is_hip() 求值：schedule_policy.py 在 import 时求值 _IS_HIP，假设进程内平台单一；混用平台或模拟场景下会误判，当前部署实践均为单平台，风险低。
  5. verify_splitkv 拒绝分支：can_handle() 新增拒绝会让 MLA 场景强制退回 extend_attention_fwd，性能可能低于理论上可支持的 verify 快路径，但避免 GPU fault 优先。 - 影响：用户与系统层面：AMD/HIP + Triton backend 用户默认获得 ragged mixed-prefill 场景 2~4 倍端到端吞吐提升，TTFT/TPOT 下降 60%~80%（open-loop 8 req/s 下从 SLA 崩溃边缘恢复到亚秒延迟）；非 AMD/NPU 用户路径零改动。该收益直接改善 AMD 上 DeepSeek 系模型的混合 prefill 服务质量。团队与运维层面：新增 3 个环境变量（1 个默认开启、2 个默认关闭）；调度器多了一个可选 admission 维度，为后续跨硬件泛化预留扩展点；BLOCK_M 自动探测消除了一类隐蔽的配置错配 bug。测试资产层面：新增的 HIP 真实硬件测试与 compact vs legacy 等价性用例成为 AMD Triton attention 内核的长期回归资产。 - 风险标记：AMD 默认开启新 kernel 路径，调度 admission 变更（默认关闭），kernel 内新增反查分支，依赖真实 HIP 硬件验证，CUDA 路径零改动约束

关联脉络

- PR #30407：评论区 yichiche 明确说明本 PR 包含了 #30407（DeepSeek MLA dispatch 修复，解决 triton spec GPU fault / segmentation fault）；commit 历史中的 “Pin DeepSeek MLA dispatch” 与 verify_splitkv MLA 拒改即来自该线。
- PR #34189 [DSV4] Fix silent KV corruption when speculative draft tokens > 4: 同属 AMD/DeepSeek 投机解码场景的 kernel 正确性修复，与本 PR 的 verify_splitkv MLA fallback、DeepSeek-R1 精度验证属于同一质量线。
- PR #34167 [DSA] Fix top-k v2 dropping non-primary ranks' output on CUDA 13.1+ (root cause for #33835): 同为 DeepSeek 系列 kernel 正确性修复，与本 PR 在 AMD/DeepSeek kernel 质量保障上是连续投入。
- PR #34161 fix: preserve GQA head mapping in Triton DCP prefill: 同为 Triton attention 内核的数值正确性修复，与本 PR 的 compact grid 等价性测试共同完善 Triton prefill 路径的回归保障。