PR #34888 完整报告

sgl-project/sglang

Split TRTLLM MHA decode batches by KV sequence length

合并时间：2026-08-20 15:44

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34888

执行摘要

- 一句话：TRTLLM 解码按 KV 长度拆分，混合上下文迭代降 10%
- 推荐动作：值得精读。重点看 _run_fixed_q_len_decode 的排序 - 分组 - 拼回模式，理解其如何在 CUDA graph 捕获下保持输出顺序；同时关注 Fridge003 提出的默认值问题，后续可结合生产数据评估是否将默认 splits 提升到 2 甚至 4。风险控制上，建议补充 splits=3/4 及大 batch 的回归测试。

功能与动机

PR body 指出 TRTLLM-GEN decode attention 在 KV 序列长度差异大的 batch 中会因长请求形成 wave 尾部而付出 tail-latency 开销：TRTLLM-GEN decode attention can pay a large tail-latency tax when requests in the same batch have very different KV sequence lengths。GB300 上 profile 显示 8.377/9.822 ms 的 attention/step delta，其中 7.601 ms 来自 SM100 variable-sequence FMHA 主核；BS44/rank 下未拆分时每请求仅一个 KV CTA 并选择 persistent kernel，长请求成为共享 wave 尾部，拖累短请求，因此需要按 KV 长度拆分以提高 CTA 并行度。

实现拆解

1. 环境变量接入（python/sglang/srt/environ.py）：新增 SGLANG_TRTLLM_MHA_DECODE_SEQ_LEN_SPLITS = EnvInt(1)，并注释说明取值大于 1 时按 KV 长度排序分组；默认 1 保证旧行为完全不变。
2. 后端配置校验（python/sglang/srt/layers/attention/trtllm_mha_backend.py）：在 __init__ 末尾读取环境变量并校验至少为 1，否则抛 ValueError；存为 self.decode_seq_len_splits。
3. 核心分组函数 _run_fixed_q_len_decode：抽出原先 forward_decode 中的固定 query 长度 TRTLLM MHA 调用，包装为闭包 run_group，透传 bmm1_scale、bmm2_scale、window_left、sinks、kv_cache_sf、multi_ctas_kv_counter_buffer 等参数，兼容 NVFP4 KV-cache 缩放与 NEXTN/MTP 的 q_len_per_req；num_splits == 1 时保持单次调用。
4. 排序、分组、拼回：num_splits > 1 时先 torch.argsort(seq_lens) 按 KV 长度排序，再用 torch.tensor_split 均分索引；对每组通过 index_select 收集 query、block_tables、seq_lens，调用 run_group，最后 index_copy_ 按原索引写回 output_by_request，保证输出顺序与输入请求一致。

5. 调用点替换与测试配套：forward_decode 中普通 decode 路径改为调用 _run_fixed_q_len_decode，ragged-query 路径不变；frozen-KV MTP 目标验证也复用同一 helper。测试文件 test/registered/attention/unittests/dense/test_trtllm_mha.py 用 envs.SGLANG_TRTLLM_MHA_DECODE_SEQ_LEN_SPLITS.override(...) 覆盖 splits=2 的 eager decode（第一个 dense 用例）和 splits=2 的 frozen-KV MTP CUDA-graph 回放，未新增测试方法或模块。

关键文件：

- python/sglang/srt/layers/attention/trtllm_mha_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 _run_fixed_q_len_decode, run_group）：核心实现文件：新增 _run_fixed_q_len_decode 按 KV 长度排序分组，并替换 forward_decode 中的固定 query 长度 decode 调用，是本 PR 的主体逻辑。
- python/sglang/srt/environ.py（模块 环境配置；类别 source；类型 configuration）：新增开关环境变量 SGLANG_TRTLLM_MHA_DECODE_SEQ_LEN_SPLITS，控制解码拆分数，默认 1 保持旧行为。
- test/registered/attention/unittests/dense/test_trtllm_mha.py（模块 单元测试；类别 test；类型 test-coverage）：测试配套：用 env override 将首个 dense decode 用例和 frozen-KV MTP CUDA-graph 用例切换为 splits=2，验证分组解码正确性。

关键符号：_run_fixed_q_len_decode, run_group, forward_decode

关键源码片段

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

核心实现文件：新增 _run_fixed_q_len_decode 按 KV 长度排序分组，并替换 forward_decode 中的固定 query 长度 decode 调用，是本 PR 的主体逻辑。

```
def _run_fixed_q_len_decode(
    self,
    query: torch.Tensor,
    kv_cache,
    block_tables: torch.Tensor,
    seq_lens: torch.Tensor,
    *,
    bmm1_scale,
    bmm2_scale,
    window_left: int,
    sinks: Optional[torch.Tensor],
    q_len_per_req: int = 1,
    kv_cache_sf=None,
) -> torch.Tensor:
    '''运行固定 query 长度 decode，可按 KV 长度排序并分组。

    分组后每组调用一次 `flashinfer.decode.trtllm_batch_decode_with_kv_cache`，
    再把输出按照原始请求顺序拼回，避免长 KV 请求拖慢短 KV 请求的波前收尾。
    '''
```

```
def run_group(group_query, group_block_tables, group_seq_lens):
    # 单组请求的 TRTLLM MHA 调用；非 1 时透传每请求的 query 长度参数
    kwargs = {}
    if q_len_per_req != 1:
        kwargs['q_len_per_req'] = q_len_per_req
    return flashinfer.decode.trtllm_batch_decode_with_kv_cache(
        query=group_query,
        kv_cache=kv_cache,
        workspace_buffer=self.workspace_buffer,
        block_tables=group_block_tables,
        seq_lens=group_seq_lens,
        max_seq_len=self.max_context_len,
        bmm1_scale=bmm1_scale,
        bmm2_scale=bmm2_scale,
        window_left=window_left,
        sinks=sinks,
        skip_softmax_threshold_scale_factor=envs.SGLANG_SKIP_SOFTMAX_DECODE_
        THRESHOLD_SCALE_FACTOR.get(),
        out_dtype=self.q_data_type,
        kv_cache_sf=kv_cache_sf,
        multi_ctas_kv_counter_buffer=self._multi_ctas_kv_counter_buffer,
        **kwargs,
    )

num_requests = seq_lens.shape[0]
# 拆分数量上限为请求数，避免空组
num_splits = min(self.decode_seq_len_splits, num_requests)
if num_splits == 1:
    # 默认路径：一次调用，保持与旧行为完全一致
    return run_group(query, block_tables, seq_lens)

# 按 KV 序列长度升序排序，等分成 num_splits 个索引组
order = torch.argsort(seq_lens)
# query 先还原为按请求组织的形状，便于分组切片与回填
query_by_request = query.view(
    num_requests, q_len_per_req, query.shape[-2], query.shape[-1]
)
output_by_request = torch.empty(
    query_by_request.shape,
    dtype=self.q_data_type,
    device=query.device,
)
for indices in torch.tensor_split(order, num_splits):
    group_output = run_group(
        query_by_request.index_select(0, indices).reshape(
            -1, query.shape[-2], query.shape[-1]
        ),
        block_tables.index_select(0, indices),
        seq_lens.index_select(0, indices),
```

```
        )
        # 按原始索引写回，保证输出顺序与输入请求一致
        output_by_request.index_copy_(
            0,
            indices,
            group_output.view(-1, q_len_per_req, query.shape[-2], query.shape[-1]),
        )
    return output_by_request.view(-1, query.shape[-2], query.shape[-1])
```

test/registered/attention/unittests/dense/test_trtllm_mha.py

测试配套：用 env override 将首个 dense decode 用例和 frozen-KV MTP CUDA-graph 用例切换为 splits=2，验证分组解码正确性。

```
# 在既有 dense decode 用例上叠加 env override，验证 splits=2 的 eager 路径
for case_index, case in enumerate(self.DECODE_CASES):
    splits = 2 if case_index == 0 else 1
    with self.subTest(
        case=case.name, backend=case.backend
    ), envs.SGLANG_TRTLLM_MHA_DECODE_SEQ_LEN_SPLITS.override(splits):
        run_dense_attention_case(
            self,
            case,
            head_dim=self.HEAD_DIM,
            hidden_size=self.HIDDEN_SIZE,
        )

# frozen-KV MTP 的 CUDA-graph 回放固定使用 splits=2，覆盖多 token 目标验证路径
for case in self.FROZEN_KV_MTP_RUNNER_CASES:
    with self.subTest(
        case=case.name, backend=case.backend
    ), envs.SGLANG_TRTLLM_MHA_DECODE_SEQ_LEN_SPLITS.override(2):
        run_dense_frozen_kv_mtp_cuda_graph_runner_case(
            self,
            case,
            head_dim=self.HEAD_DIM,
            hidden_size=self.HIDDEN_SIZE,
        )
```

评论区精华

Fridge003 在 PR 评论中提出了唯一的设计问题：*Is there any downside of enabling num_splits > 1, otherwise we can set default num_split to 2 or 4*。该问题询问启用多个 splits 的潜在代价，并建议若代价不大可将默认值设为 2 或 4。作者没有在 PR 内给出文字回复，最终合入时仍保持默认 1（opt-in），说明该决策倾向保守，默认不改变已有行为。这为后续基于生产数据调整默认值保留了空间。其余评论为 CI 重跑命令（/tag-and-rerun-ci、/rerun-failed-ci），无实质技术讨论。

- num_splits 默认值与 downside (design): PR 合入时保留默认 1，未在 PR 内明确回复；合并者最终批准，说明接受 opt-in 式保守默认。

风险与影响

- 风险：
  1. 正确性 / 回归风险：改动位于 *decode attention* 核心路径；尽管默认关闭，开启后涉及排序与多次 kernel 启动，若 TRTLLM MHA 对 batch 形状有隐藏假设（如 CTA 数量对齐），可能引入数值或稳定性问题。测试覆盖 splits=2，未覆盖 splits>2 与极大 batch。
  2. 性能风险：torch.argsort、tensor_split、index_select、index_copy_ 在每层 *decode* 都会执行，对超大 batch 会增加 host 侧与数据搬移开销；对均匀长度 batch 有约 +0.20% 的轻微回退。分组过细会导致 kernel 启动次数线性增长，需在生产中调参。
  3. 兼容性：涉及 FlashInfer TRTLLM 接口的直接调用，后续依赖上游 API 变化；*environ.py* 新增环境变量不影响其他后端。
  4. 影响面：仅影响 trtllm_mha_backend.py 的固定 *query* 长度 *decode* 路径（含 frozen-KV MTP），*ragged-query* 与 *prefill* 不受影响。 - 影响：用户：默认无行为变化；在混合上下文场景开启环境变量后可见解码迭代延迟下降约 10%，生产 split-count 调优（split-3 到 split-4）TPS 提升约 4.9%，ITL 改善约 4.03%。系统：新增一个后端级 opt-in 配置，无 *schema* 或部署变更；分组逻辑仅影响 TRTLLM MHA 固定 *query* 长度 *decode* 路径。团队：提供了“排序拆批缓解波前尾部”的可复用范式，测试中的 *env override* 模式也可推广到其他 *attention* 后端的调优验证。
- 风险标记：核心 *decode* 路径变更，新环境变量默认关闭，排序与 *scatter* 额外开销，测试仅覆盖 splits=2

关联脉络

- PR #36219 [Performance] Tune FlashInfer EXTEND for DP prefill: 同属近期解码 / 注意力性能调优系列，关注混合上下文下的尾部开销；一个是 prefill EXTEND 预热，一个是 *decode* 拆批，可对照阅读批处理波前优化思路。