PR #25002 完整报告

sgl-project/sglang

[spec_v2] Enable trtllm_mha draft-extend CUDA graph with v2 semantics

合并时间：2026-06-05 08:50

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25002

执行摘要

- 一句话：为 trtllm_mha 启用 spec_v2 draft-extend CUDA graph
- 推荐动作：值得精读。本 PR 展示了在复杂推测解码路径中启用 CUDA graph 的完整思路：白名单控制、metadata 语法适配、合理精简 graph 内部计算量以避免浪费，以及对应的测试合约更新。对于理解 speculative v2、TRTLLM backend 以及 CUDA graph 的正确使用很有参考价值。

功能与动机

V2 draft-extend 路径此前无法启用 trtllm_mha 的 CUDA graph，因为在白名单中缺失 TRTLLMHAAttnBackend。且旧 replay 逻辑会计算完整的 softmax+topk/reduce，但这些结果在 V2 worker 中不被使用（worker 会重新对选中行做 softmax+fast_topk），造成无效计算。

实现拆解

1. 白名单扩展：在 eagle_worker_v2.py 的 supports_cuda_draft_extend_graph 条件中添加 TRTLLMHAAttnBackend，并按照 review 建议将孤立 isinstance 合并为元组判断。
2. Metadata 分支调整：在 trtllm_mha_backend.py 中，将 .is_draft_extend() 调用改为 .is_draft_extend(include_v2=True)，让 V2 也进入 draft-extend 的 metadata 构建分支。并在 _apply_cuda_graph_metadata 内新增 is_draft_extend_v2() 分支：V2 使用 spec_info.num_tokens_per_req 作为统一步长填充 cu_seqlens_q 和 max_seq_len_q，不再沿用 V1 的 num_accept_tokens 变长语义。
3. Graph 输出精简：在 eagle_draft_extend_cuda_graph_runner.py 的 replay 方法中，针对 DRAFT_EXTEND_V2 模式，跳过 topk_p 和 topk_index 的切片赋值，只保留 next_token_logits 和 hidden_states 的输出（graph 内仍通过 torch.amax 锚定全 logits 以满足 CUDA graph 生命周期要求，但不产生 top-k 输出）。
4. 测试适配：在 speculative_draft_extend_runner.py 中新增 _assert_draft_extend_v2_outputs_close 函数，仅比较 logits 和 hidden_states，不再断言 topk 字段；并在 dense/MLA 的两条 V2 测试用例中将其挂载为 assert_outputs_close。

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）：核心：在 _build_cuda_graph_metadata 和

_apply_cuda_graph_metadata 中扩展 draft_extend 条件以包含 V2，并新增 V2 专属的 metadata 填充逻辑。

- python/sglang/srt/speculative/eagle_worker_v2.py（模块 推测解码；类别 source；类型 dependency-wiring）：入口：在此文件中导入 TRTLLMHAAttnBackend 并将其加入白名单，同时内部 _draft_extend_for_decode 添加注释说明 graph 输出语义。
- python/sglang/srt/speculative/eagle_draft_extend_cuda_graph_runner.py（模块 CUDA Graph 运行器；类别 source；类型 core-logic）：核心逻辑：在 replay 中对 V2 模式跳过 topk 复制，避免冗余计算。
- python/sglang/test/kits/attention_unittest/runner_modes/speculative_draft_extend_runner.py（模块 测试；类别 test；类型 test-coverage；符号 _assert_draft_extend_v2_outputs_close）：测试保障：新增针对 V2 的断言函数，不检查 topk 字段；在 dense 和 MLA 两条 V2 测试用例中注册该断言。

关键符号：_assert_draft_extend_v2_outputs_close

关键源码片段

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

核心：在 _build_cuda_graph_metadata 和 _apply_cuda_graph_metadata 中扩展 draft_extend 条件以包含 V2，并新增 V2 专属的 metadata 填充逻辑。

```
# python/sglang/srt/layers/attention/trtllm_mha_backend.py（片段）

# 在 _build_cuda_graph_metadata 中，将 draft_extend 分支条件扩展为 include_v2
elif forward_mode.is_draft_extend(include_v2=True): # ← 同时覆盖 V1 和 V2
    num_tokens_per_bs = num_tokens // bs
    metadata.cache_seqlens_int32 = self.draft_extend_metadata["cache_seqlens"][:bs]
    metadata.cu_seqlens_q = self.draft_extend_metadata["cu_seqlens_q"][:bs + 1]
    metadata.cu_seqlens_k = self.draft_extend_metadata["cu_seqlens_k"][:bs + 1]
    metadata.max_seq_len_q = num_tokens_per_bs
    metadata.page_table = self.draft_extend_metadata["page_table"][:bs, :]
    # ... bind swa page table
    self.draft_extend_metadata[bs] = metadata

# 在 _apply_cuda_graph_metadata 中，之前的条件也改为 include_v2
elif forward_mode.is_draft_extend(include_v2=True):
    metadata = self.draft_extend_metadata[bs]
    metadata.cache_seqlens_int32.copy_(seq_lens)
    metadata.max_seq_len_k = seq_lens_cpu.max().item()
    # ... cu_seqlens_k cumsum

    # V2 与 V1 分流：V2 使用 num_tokens_per_req 作为一致步长
    if forward_mode.is_draft_extend_v2():
        num_tokens_per_bs = spec_info.num_tokens_per_req
        if num_tokens_per_bs <= 0:
            # 捕获阶段使用合成输入，fallback 推断步长
            num_tokens_per_bs = int(
```

```
                spec_info.num_accept_tokens[:bs].max().item()
            )
        metadata.max_seq_len_q = num_tokens_per_bs
        # cu_seqlens_q 填充为等差数列：0, step, 2*step, ...
        metadata.cu_seqlens_q[1:].copy_(
            torch.arange(
                num_tokens_per_bs,
                bs * num_tokens_per_bs + 1,
                num_tokens_per_bs,
                dtype=torch.int32,
                device=metadata.cu_seqlens_q.device,
            )
        )
    else:
        # V1 分支不变：使用 num_accept_tokens 变长填充
        extend_lens = spec_info.num_accept_tokens[:bs]
        if spec_info.num_accept_tokens_cpu:
            metadata.max_seq_len_q = max(spec_info.num_accept_tokens_cpu)
        else:
            metadata.max_seq_len_q = 1
        metadata.cu_seqlens_q[1:].copy_(
            torch.cumsum(extend_lens, dim=0, dtype=torch.int32)
        )
```

python/sglang/srt/speculative/eagle_worker_v2.py

入口：在此文件中导入 TRTLLMHAAttnBackend 并将其加入白名单，同时内部 _draft_extend_for_decode 添加注释说明 graph 输出语义。

```
# python/sglang/srt/speculative/eagle_worker_v2.py（片段）

from sglang.srt.layers.attention.trtllm_mha_backend import TRTLLMHAAttnBackend # 新增导入

# ... 在 init_cuda_graphs 中
supports_cuda_draft_extend_graph = (_is_cuda or _is_musa) and isinstance(
    self.draft_extend_attn_backend,
    (
        TritonAttnBackend,
        TRTLLMMLABackend,
        TRTLLMHAAttnBackend, # 新增
        TokenspeedMLABackend,
    ),
)

# 在 _draft_extend_for_decode 中，graph 输出只锚定 logits，top-k 由 worker 后算
# The draft-extend graph only anchors full logits; selected-row topk is
# owned by the worker for both graph and eager paths.
```

python/sglang/srt/speculative/eagle_draft_extend_cuda_graph_runner.py

核心逻辑：在 replay 中对 V2 模式跳过 topk 复制，避免冗余计算。

```
# python/sglang/srt/speculative/eagle_draft_extend_cuda_graph_runner.py（片段）

# 在 replay 方法末尾，unpadding 部分
if unpadding_bs is not None:
    out_copy = out
    # 构造只含 logits 和 hidden_states 的输出，不拷贝 topk
    out = LogitsProcessorOutput(
        next_token_logits=out.next_token_logits[:unpadding_bs],
        hidden_states=out.hidden_states[:unpadding_bs],
    )
    # 对于 V2 模式，graph 内部已通过 torch.amax 锚定 logits 但未输出 topk
    if self.forward_mode != ForwardMode.DRAFT_EXTEND_V2:
        out.topk_p = out_copy.topk_p[:raw_bs]
        out.topk_index = out_copy.topk_index[:raw_bs]
return out
```

评论区精华

Review 中 merrymercy 建议将 eagle_worker_v2.py 中孤立的 or isinstance(..., TRTLLMHAAttnBackend) 改为用元组统一判断，类似已有的其他 backend 写法。作者采纳并修改。

- isinstance 调用风格 (style): 作者接受并修改，简化代码。

风险与影响

- 风险：
  1. 兼容性：is_draft_extend(include_v2=True) 会同时匹配 V1 和 V2，需确认原 V1 的 draft_extend 路径在 metadata 构建和 replay 中行为不变（patch 中在 _apply_cuda_graph_metadata 内部用 is_draft_extend_v2() 分流，不影响 V1 逻辑）。
  2. 边界条件：V2 的 num_tokens_per_req 可能在 capture 阶段为 0（合成输入），代码中 fallback 到 num_accept_tokens[:bs].max().item()，若所有 accept_tokens 也为 0 可能导致异常，但实际 capture 输入设计保证了至少有一个 token。
  3. 测试覆盖：新增的 V2 断言不再校验 topk，如果未来修改了 worker 与 graph 的 top-k 交付契约，测试可能无法捕获回归。但当前设计明确将 top-k 计算后置到 worker，因此测试只验证 graph 实际锚定的输出是合理的。 - 影响：对用户：当使用 trtllm_mha 作为 draft-extend attention backend 且开启 spec_v2 时，draft-extend 步骤将自动享受 CUDA graph 加速，同时消除之前 graph 中无用的 top-k 计算，提升推理吞吐。不影响已有 V1 行为或其它 backend 路径。 对系统：无新增配置项，启动时自动生效。需要 CUDA 环境且 trtllm_mha 可用。
- 风险标记：路径兼容性，边界条件依赖捕获输入，测试契约变更

关联脉络

- PR #27300 fix(spec): complete CustomSpecAlgo duck-typing interface and guard against drift: 同为 speculative decoding 模块的接口完善与测试增强，反映团队对该模块

正确性的持续关注。