PR #26512 完整报告

sgl-project/sglang

Fix FA DRAFT_EXTEND_V2 cache extent

合并时间：2026-05-28 15:56

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26512

执行摘要

- 一句话：修复 FlashAttention DRAFT_EXTEND_V2 缓存范围错误
- 推荐动作：值得精读。该 PR 展示了注意力后端中缓存范围元数据的精细语义差异，特别是 DRAFT_EXTEND_V2 中 seq_lens 与有效缓存长度不一致时的正确处理方式。设计决策如 per-request 求和取最大值而非简单双 max 求和，体现了对偏斜分布的考量，值得在其他注意力后端实现中参考。

功能与动机

DRAFT_EXTEND_V2 模式下，forward_batch.seq_lens 表示前缀长度（新 extend token 写入前的缓存长度），而 FlashAttention 在 forward_extend 中通过 set_kv_buffer 已将新 K 写入缓存。原实现错误地将 seq_lens 视为完整缓存长度，导致注意力内核读取的缓存范围仅覆盖前缀，遗漏刚写入的 extend 行，产生约 0.55 最大绝对差（vs HF 参考实现）。

实现拆解

1. Eager 路径 init_forward_metadata（文件 flashattention_backend.py，约 506-533 行）：在原有的 is_extend_or_draft_extend_or_mixed 分支内，先判断 is_draft_extend_v2()。若是，则计算逐请求的有效缓存长度 effective_cache_seqlens = seqlens_in_batch + forward_batch.extend_seq_lens，并基于 per-request 求和取最大值 max_i(prefix[i] + extend[i]) 作为 max_seq_len_k（避免使用 max(prefix) + max(extend) 导致的过度分配）；否则保持原逻辑直接使用 seqlens_in_batch。然后将 metadata.cache_seqlens_int32、max_seq_len_k、cu_seqlens_k 全部基于 effective_cache_seqlens 计算。
2. CUDA Graph 重放路径 init_forward_metadata_replay_cuda_graph（文件 flashattention_backend.py，约 2290-2335 行）：在 is_draft_extend_v2() 分支中，新增逻辑从 spec_info 获取 extend_seq_lens_tensor 和 extend_seq_lens_cpu（若不存在则推测默认值）。然后计算 effective_cache_seqlens = seq_lens + extend_seq_lens_tensor，并将 cache_seqlens_int32、max_seq_len_k、cu_seqlens_k 以及后续的 max_seq_pages 和 page_table 全部基于前缀加 extend 的范围重新计算，使得收集到的页表覆盖注意力内核读取的所有列。
3. 非 DRAFT_EXTEND_V2 路径保持不变：原有 EXTEND 模式不受影响，因为其 seq_lens 已是完整缓存长度。

关键文件：

- python/sglang/srt/layers/attention/flashattention_backend.py（模块 注意力层；类别 source；类型 core-logic；符号 init_forward_metadata, init_forward_metadata_replay_cuda_graph）：核心变更文件，修复了 FlashAttention 后端中 DRAFT_EXTEND_V2 模式下的缓存范围元数据计算错误，涉及 eager 和 CUDA Graph 重放两条路径。

关键符号：init_forward_metadata, init_forward_metadata_replay_cuda_graph

关键源码片段

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

核心变更文件，修复了 FlashAttention 后端中 DRAFT_EXTEND_V2 模式下的缓存范围元数据计算错误，涉及 eager 和 CUDA Graph 重放两条路径。

```
# python/sglang/srt/layers/attention/flashattention_backend.py # Eager 路径中的关键分支 (init_forward_metadata) elif forward_batch.forward_mode.is_extend_or_draft_extend_or_mixed(     include_draft_extend_v2=True ):     # DRAFT_EXTEND_V2: seq_lens 仅为前缀长度，实际 KV 缓存需包含新写入的 extend 内容     if forward_batch.forward_mode.is_draft_extend_v2():         # 逐请求计算有效缓存长度：prefix_len + extend_len         effective_cache_seqlens = (             seqlens_in_batch +             forward_batch.extend_seq_lens         )         seq_lens_cpu = forward_batch.seq_lens_cpu         extend_seq_lens_cpu = forward_batch.extend_seq_lens_cpu         if extend_seq_lens_cpu is not None:             extend_cpu_tensor = torch.as_tensor(                 extend_seq_lens_cpu,                 dtype=seq_lens_cpu.dtype             )             # 使用 per-request 求和后的最大值，避免 max(prefix) + max(extend) 在偏斜分布下过度分配             effective_max_seq_len_k = int(                 (seq_lens_cpu + extend_cpu_tensor).max().item()             )         else:             effective_max_seq_len_k = int(effective_cache_seqlens.max().item())     else:         # 非 DRAFT_EXTEND_V2，seq_lens 即完整缓存长度         effective_cache_seqlens = seqlens_in_batch         effective_max_seq_len_k = int(forward_batch.seq_lens_cpu.max().item())     # 所有元数据基于 effective 值计算，而非原始的 seqlens_in_batch     metadata.cache_seqlens_int32 = effective_cache_seqlens.to(torch.int32)     metadata.max_seq_len_k = effective_max_seq_len_k     metadata.cu_seqlens_k = torch.nn.functional.pad(         torch.cumsum(effective_cache_seqlens, dim=0, dtype=torch.int32),         (1, 0),     ) # CUDA Graph 重放路径中的关键分支 (init_forward_metadata_replay_cuda_graph) elif forward_mode.is_draft_extend_v2():     metadata = self.draft_extend_metadata[bs]     # 从 spec_info 获取 extend 长度（兼容属性可能不存在的情况）     extend_seq_lens_tensor = getattr(spec_info, "extend_seq_lens_tensor", None)     extend_seq_lens_cpu = getattr(spec_info, "extend_seq_lens_cpu", None)     if extend_seq_lens_tensor is not None:         pass # 使用已有的 extend_seq_lens_tensor     else:         # fallback: 推算默认 extend 长度         default_extend = forward_batch.seq_lens[0].item() if forward_batch else 1         extend_seq_lens_tensor = torch.full(             (bs,), default_extend, dtype=torch.int32, device=forward_batch.seq_lens.device         )         extend_seq_lens_cpu = [default_extend] * bs     # 有效缓存长度 = 前缀 + extend
```

```
effective_cache_seqlens = seq_lens.to(torch.int32) + extend_seq_lens_tensor
metadata.cache_seqlens_int32.copy_(effective_cache_seqlens) if
extend_seq_lens_cpu is not None: extend_cpu_tensor = torch.as_tensor(
extend_seq_lens_cpu, dtype=seq_lens_cpu.dtype ) metadata.max_seq_len_k
= int( (seq_lens_cpu + extend_cpu_tensor).max().item() ) else:
metadata.max_seq_len_k = int(effective_cache_seqlens.max().item())
metadata.cu_seqlens_k[1:].copy_( torch.cumsum(metadata.cache_seqlens_int32,
dim=0, dtype=torch.int32) )
```

评论区精华

该 PR 无 review 评论，仅由作者自行合并。PR body 中详细描述了 bug 根因、修复方案和精度验证结果。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 回归风险极低：修复仅修改 DRAFT_EXTEND_V2 分支逻辑，非 V2 的 EXTEND 路径完全不变。PR 提供了 FA3/FA4 在 *eager* 和 CUDA Graph 下的精度验证，最大绝对差从 ~0.55 降至 ≤ atol，且非 V2 EXTEND 回归测试通过。
  2. 性能无影响：变更仅在 CPU 端元数据初始化路径上，未改动内核，无运行时开销。
  3. 边界情况：当 *extend_seq_lens_cpu* 为 None 时使用 fallback 默认值（推测自 forward_batch.seq_lens 或 1），可能在某些非预期场景下覆盖不全，但已通过属性安全获取。
- 影响：
  1. 用户影响：修复了 EAGLE v2 等多层 *draft-extend* 模型使用 FlashAttention 时的数值错误，影响面限于 DRAFT_EXTEND_V2 模式用户，非 V2 用户无感知。
  2. 系统影响：无部署变更，无需配置更新。
  3. 团队影响：为后续注意力后端单元测试矩阵提升（PR 提及的 follow-up 测试 PR）奠定正确性基础。 - 风险标记：核心路径变更，无直接测试文件配套

关联脉络

- PR #26513 Fix FlashInfer SWA EXTEND-with-prefix *correctness* in *merge_state* path: 同为注意力后端正确性修复，涉及 EXTEND 模式的语义理解，属于同一注意力基础设施改进线路。
- PR #24737 Support Flashinfer Cute-DSL MLA attention: 引入 FlashInfer Cute-DSL 后端，本 PR 修复了 FlashAttention 后端的问题，两者互补覆盖不同注意力后端。
- PR #26515 Allow Optional key/value in unified_attention_with_output split-op (MLA *absorb* fix): 同为注意力后端的 bugfix，修复 CUDA Graph 场景下的崩溃问题，与本 PR 有相似的技术关注点。