PR #32383 完整报告

sgl-project/sglang

Optimize EmbeddingGemma prefill performance

合并时间：2026-07-27 17:34

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32383

执行摘要

- 一句话：优化 EmbeddingGemma prefill 性能，BCG 加速 4.37 倍
- 推荐动作：建议精读。该 PR 展示了如何将 encoder-only 模型适配到 SGLang 的 BCG 框架中，包括残差融合、批次 tokenization、注意力窗口正确性等关键设计。layernorm.py 中的融合 RMSNorm 内核设计值得复用。server_args.py 中的自动配置逻辑也可作为其他特殊模型配置的参考。

功能与动机

支持 Google EmbeddingGemma 文本嵌入模型，该模型基于双向 Gemma 3 编码器，需要特殊的 prefill 处理（禁止 prefix cache 和 chunked prefill，必须使用 BCG）。原实现使用 eager 模式，性能不佳。此 PR 通过批处理 tokenization、残差融合和 BCG 最大化其 prefill 性能，使其适用于生产部署。

实现拆解

1. 模型适配：在 gemma3_causal.py 中新增 _build_sentence_transformer_projector 函数加载 SentenceTransformer 的 Dense 投影器权重，并修改 Gemma3DecoderLayer.forward 接口增加 residual 参数，使残差可在层间传递供融合 RMSNorm 使用。
2. 残差融合 RMSNorm：在 layernorm.py 中为 Gemma3RMSNorm 所有 forward_* 方法添加 residual 参数；CUDA 路径使用 gemma_fused_add_rmsnorm 内核原地更新 x 和 residual，减少内存带宽。
3. 注意力后端正交：在 flashattention_backend.py 中解耦 causal 与 window_size，当 attn_type 为 DECODER_BIDIRECTIONAL 时对称设置 sliding window 左右宽度，确保编码器正确应用双向局部注意力。
4. 服务器自动配置：在 server_args.py 的 _handle_model_capability_adjustments 中检测 is_embedding_gemma 后自动设置 is_embedding=True、enable_tokenizer_batch_encode=True、prefill_only_disable_kv_cache=True（仅 Hopper/Blackwell），并调整 BCG 默认 max_bs 到 16384 tokens。
5. Tokenizer 适配：在 tokenizer_manager.py 的 _tokenize_texts 中为 EmbeddingGemma 请求自动追加 EOS token（若缺失），确保与 SentenceTransformer 行为一致。

6. 文档与配置：新增 docs_new/cookbook/autoregressive/Google/EmbeddingGemma.mdx 提供部署指南，更新 docs.json 注册页面。

关键文件：

- python/sglang/srt/models/gemma3_causal.py（模块 模型定义；类别 source；类型 data-contract；符号 _build_sentence_transformer_projector）：核心模型文件，添加 SentenceTransformer 投影器支持，修改解码器层 forward 接口以支持残差传递，实现双向注意力正确性。
- python/sglang/srt/layers/layernorm.py（模块 归一化层；类别 source；类型 core-logic；符号 forward_native, forward_cpu, forward_cuda, forward_npu）：融合残差 RMSNorm，提供 CUDA 内核 gemma_fused_add_rmsnorm 以在归一化前原地累加残差，减少内存开销。
- python/sglang/srt/server_args.py（模块 服务器配置；类别 source；类型 core-logic）：自动配置 EmbeddingGemma 服务器参数：启用 embedding 模式、batch tokenization、FA3 raw-K/V 路径、BCG 预填充尺寸。
- python/sglang/srt/layers/attention/flashattention_backend.py（模块 注意力后端；类别 source；类型 core-logic）：调整 attention 窗口计算以支持双向滑动窗口，确保 EmbeddingGemma 编码器正确使用 local attention。
- python/sglang/srt/managers/tokenizer_manager.py（模块 标记器；类别 source；类型 core-logic）：为 EmbeddingGemma 请求自动追加 EOS token，确保与 SentenceTransformer 行为一致。
- docs_new/cookbook/autoregressive/Google/EmbeddingGemma.mdx（模块 文档；类别 other；类型 core-logic）：新增 EmbeddingGemma 部署 cookbook，提供启动命令、性能默认值和 curl 示例。
- docs_new/docs.json（模块 文档配置；类别 config；类型 configuration）：注册 EmbeddingGemma cookbook 到文档索引。

关键符号：_build_sentence_transformer_projector, Gemma3RMSNorm.forward_native, Gemma3RMSNorm.forward_cuda, Gemma3DecoderLayer.forward, Gemma3Model.forward

关键源码片段

python/sglang/srt/models/gemma3_causal.py

核心模型文件，添加 SentenceTransformer 投影器支持，修改解码器层 forward 接口以支持残差传递，实现双向注意力正确性。

```
class Gemma3DecoderLayer(nn.Module):
    # ... 其他方法 ...
    def forward(
        self,
        positions: torch.Tensor,
        hidden_states: torch.Tensor,
        position_embeddings_global: torch.Tensor,
        position_embeddings_local: torch.Tensor,
```

```
        forward_batch: ForwardBatch,
        residual: Optional[torch.Tensor] = None, # 残差输入，来自上一层的输出或 residual
        **kwargs,
    ) -> tuple[torch.FloatTensor, Optional[tuple[torch.FloatTensor, torch.FloatTensor]]]:
        # 保持 residual 在层间传递，使得下一个 RMSNorm 可以融合加法
        if residual is None:
            residual = hidden_states
            hidden_states = self.input_layernorm(hidden_states)
        else:
            hidden_states, residual = self.input_layernorm(hidden_states, residual)

        # 应用全局 RoPE 到非滑动层
        if self.self_attn.is_sliding:
            position_embeddings = position_embeddings_local
        else:
            position_embeddings = position_embeddings_global

        hidden_states = self.self_attn(
            positions=positions,
            hidden_states=hidden_states,
            position_embeddings=position_embeddings,
            forward_batch=forward_batch,
            **kwargs,
        )
        hidden_states = self.post_attention_layernorm(hidden_states)
        # 残差加法已融入 RMSNorm，此处不再单独加
        hidden_states, residual = self.pre_feedforward_layernorm(
            hidden_states, residual
        )
        hidden_states = self.mlp(hidden_states)
        hidden_states = self.post_feedforward_layernorm(hidden_states)
        outputs = (hidden_states, residual)
        return outputs
```

python/sglang/srt/layers/layernorm.py

融合残差 RMSNorm，提供 CUDA 内核 gemma_fused_add_rmsnorm 以在归一化前原地累加残差，减少内存开销。

```
class Gemma3RMSNorm(MultiPlatformOp):
    # ...
    def forward_native(self, x, residual: Optional[torch.Tensor] = None):
        # 若传入 residual，先原地累加再归一化
        if residual is not None:
            residual = x + residual
            x = residual
        output = self._norm(x.float())
        # Gemma3 风格：  (x * w).to(float16)
        output = output * (1.0 + self.weight.float())
        output = output.type_as(x)
```

```
        # 若存在 residual，返回 ( 归一化结果 , 累加后的 residual) 供下一层使用
        return output if residual is None else (output, residual)

    def forward_cuda(self, x, residual: Optional[torch.Tensor] = None):
        if residual is not None:
            # 使用融合内核原地更新 x 和 residual
            # x 变为归一化输出，residual 变为 x + residual
            gemma_fused_add_rmsnorm(x, residual, self.weight.data, self.eps)
            return x, residual
        if x.dim() == 2:
            return gemma_rmsnorm(x, self.weight.data, self.eps)
        return self.forward_native(x)
```

python/sglang/srt/server_args.py

自动配置 EmbeddingGemma 服务器参数：启用 embedding 模式、batch tokenization、FA3 raw-K/V 路径、BCG 预填充尺寸。

```
# EmbeddingGemma 检测与自动配置
if getattr(model_config, "is_embedding_gemma", False):
    # 标记为嵌入模式，启用 FA raw-K/V 快速路径
    self.is_embedding = True
    self.disable_radix_cache = True
    self.chunked_prefill_size = -1
    # 启用批量 tokenization，使 BCG 能捕获完整预填充批次
    self.enable_tokenizer_batch_encode = True
    requested_prefill_backend = (
        self.prefill_attention_backend or self.attention_backend
    )
    if (
        is_cuda()
        and (is_sm90_supported() or is_sm100_supported())
        and requested_prefill_backend in (None, "fa3", "fa4")
    ):
        # Hopper/Blackwell 上启用无 KV 缓存预填充路径
        self.prefill_only_disable_kv_cache = True
        self._validate_prefill_only_disable_kv_cache_args()

    self.cuda_graph_config.decode.backend = Backend.DISABLED
    if is_cuda() and self.cuda_graph_config.prefill.backend != Backend.DISABLED:
        self.cuda_graph_config.prefill.backend = Backend.BREAKABLE
        prefill_config = self.cuda_graph_config.prefill
        cuda_graph_config_locked = getattr(
            self, "_cuda_graph_config_locked", set()
        )
        # 默认提升 BCG 最大 token 数到 16K，覆盖 8 条 2K 请求
        if (Phase.PREFILL, "max_bs") not in cuda_graph_config_locked:
            prefill_config.max_bs = max(
                prefill_config.max_bs or 0,
                model_config.context_len,
```

```
            16384,
        )
        if (Phase.PREFILL, "bs") not in cuda_graph_config_locked:
            prefill_config.bs = (
                self._generate_prefill_cuda_graph_batch_sizes(
                    prefill_config.max_bs
                )
            )
```

评论区精华

无 Review 讨论。PR 作者在评论中提及由于 CI 阻塞而绕过，但会持续监控。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. gemma3_causal.py 中 Gemma3DecoderLayer.forward 接口新增 residual 参数，所有内部调用点已更新，但若存在外部继承或未覆盖的子类可能遗漏。
  2. layernorm.py 中 forward_cuda 使用 gemma_fused_add_rmsnorm 内核，该内核要求 tensor 连续且 token-major，不满足时可能静默回退或产生错误。
  3. server_args.py 中自动设置 prefill_only_disable_kv_cache 仅对 Hopper/Blackwell 生效，其他 GPU 回退到 paged-KV 路径，可能引入性能回归。
  4. 整个 PR 未添加新的单元测试，回归覆盖依赖现有 CI，边缘情况（如非 GPU 平台、老 GPU）风险较高。 - 影响：对用户：首次支持高性能 EmbeddingGemma 服务，默认启用 BCG 和 FA3，吞吐量达 496k tok/s。对系统：修改了注意力后端、归一化层和 tokenizer 管理器，但所有变更通过 is_embedding_gemma 标志隔离，不影响其他模型。对团队：新增 EmbeddingGemma cookbook 降低用户部署门槛。 - 风险标记：残差融合接口变更，双向注意力正确性依赖，BCG 失效回退，缺少测试覆盖

关联脉络

- PR #32375 model: support EmbeddingGemma: 此 PR 在 #32375 引入的基本支持上优化 prefill 性能，包括 BCG 适配和残差融合。