PR #45163 完整报告

vllm-project/vllm

[Model] Add DiffusionGemma Support

合并时间：2026-06-12 13:17

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45163

执行摘要

- 一句话：为 vLLM 添加 DiffusionGemma 离散扩散模型支持
- 推荐动作：
  - 值得精读：diffusion_gemma.py 中的混合因果注意力实现和 DiffusionGemmaSampler 是理解扩散推理的核心。
  - 关注设计决策：复用推测解码数据路径度量（metrics.py）是一种可复用的工程模式，适用于其他非自回归模型。
  - 参考价值：gemma4 工具解析器的批量 token 处理增强，对处理多 token 发射的流式场景有借鉴意义。

功能与动机

支持 Google 发布的 DiffusionGemma 离散扩散语言模型，使用户能够在 vLLM 中运行该模型进行推理。该模型采用块扩散（block diffusion）生成方式，与标准自回归模型不同，需要新的模型实现和推理路径。PR body 中未提供具体 issue，但引用了 Google 的 recipe 和 Docker 镜像。

实现拆解

1. 配置类：在 vllm/transformers_utils/configs/diffusion_gemma.py 中新增 DiffusionGemmaConfig 和 DiffusionGemmaTextConfig，处理 MoE 启用、K=V 共享等特定配置，并定义 canvas_length 和 self_conditioning_size 参数。
2. 模型核心：在 vllm/model_executor/models/diffusion_gemma.py 中新增完整的模型实现，包括 DiffusionGemmaSelfConditioning（门控 MLP 自条件化模块）、文本编码器 / 解码器（混合因果 / 双向注意力）、扩散采样器 DiffusionGemmaSampler，以及多模态处理信息。
3. 配置注册：在 vllm/model_executor/models/config.py 中添加 DiffusionGemmaModelForBlockDiffusionConfig，自动设置默认注意力后端（排除 FlashInfer，启用 use_non_causal），创建 DiffusionConfig，并限制 max_num_seqs 为 8 以防止 OOM。
4. 度量复用：在 vllm/v1/spec_decode/metrics.py 中为 SpecDecodingLogging 和 SpecDecodingProm 添加 is_diffusion 标志和 _log_diffusion 方法，复用推测解码的数据路径但以扩散语义输出。在 vllm/benchmarks/serve.py 中添加 DiffusionMetrics 数据类和

fetch_diffusion_metrics 函数，支持基准测试报告。

5. 工具解析适配：增强 vllm/tool_parsers/gemma4_tool_parser.py 以支持 DiffusionGemma 的批量 token 发射，允许一次 delta 中开始多个工具调用，并添加对 required/named tool choice 的支持（跳过结构化输出引导解码）。
6. 测试配套：新增 tests/kernels/attention/test_mixed_causal_attn.py 测试混合因果注意力正确性，增强 tests/tool_parsers/test_gemma4_tool_parser.py 测试批量工具调用场景。

关键文件：

- vllm/model_executor/models/diffusion_gemma.py（模块 扩散模型；类别 source；类型 core-logic；符号 DiffusionGemmaSelfConditioning, init, forward, DiffusionGemmaProcessingInfo）：新增模型核心实现，包含 DiffusionGemmaSelfConditioning、混合因果 / 双向注意力、扩散采样器等，是 PR 的核心变更。
- vllm/transformers_utils/configs/diffusion_gemma.py（模块 扩散配置；类别 source；类型 core-logic；符号 _init_text_config, DiffusionGemmaTextConfig, init, DiffusionGemmaConfig）：新增 DiffusionGemmaConfig 和 DiffusionGemmaTextConfig 配置类，用于加载 HuggingFace 模型配置。
- vllm/v1/spec_decode/metrics.py（模块 推测解码；类别 source；类型 core-logic；符号 init, _log_diffusion）：为扩散模型添加专用度量记录和 Prometheus 计数器，复用推测解码的数据路径但以扩散语义输出。
- vllm/benchmarks/serve.py（模块 基准测试；类别 source；类型 core-logic；符号 DiffusionMetrics, fetch_diffusion_metrics）：添加 DiffusionMetrics 数据类和 fetch_diffusion_metrics 函数，支持在基准测试中获取扩散度量。
- vllm/model_executor/models/config.py（模块 模型注册；类别 source；类型 data-contract；符号 DiffusionGemmaModelForBlockDiffusionConfig, verify_and_update_config）：注册 DiffusionGemma 模型配置类，设置默认注意力后端、canvas_length 和 max_num_seqs 内存限制。
- vllm/tool_parsers/gemma4_tool_parser.py（模块 工具解析；类别 source；类型 core-logic；符号 _handle_tool_call_end）：增强 Gemma4 工具解析器以支持 DiffusionGemma 的批量 token 发射，允许同时开始多个工具调用。

关键符号：DiffusionGemmaSelfConditioning, DiffusionGemmaProcessingInfo, _softcap_logits, DiffusionGemmaConfig, _init_text_config, _log_diffusion, DiffusionMetrics, fetch_diffusion_metrics, verify_and_update_config, _handle_tool_call_end

关键源码片段

vllm/model_executor/models/diffusion_gemma.py

新增模型核心实现，包含 DiffusionGemmaSelfConditioning、混合因果 / 双向注意力、扩散采样器等，是 PR 的核心变更。

```
class DiffusionGemmaSelfConditioning(nn.Module):
    """Gated MLP that processes soft embeddings from the previous denoising step.
```

```
    Structurally identical to Gemma4MLP but with self_conditioning_size
    and post_norm without learned scale.
    """
    def __init__(self, hidden_size: int, self_conditioning_size: int, eps: float = 1e-6):
        super().__init__()
        self.pre_norm = RMSNorm(hidden_size, eps=eps)
        self.post_norm = RMSNorm(hidden_size, eps=eps, has_weight=False)
        # gate / up / down projection 构成 gated MLP
        self.gate_proj = nn.Linear(hidden_size, self_conditioning_size, bias=False)
        self.up_proj = nn.Linear(hidden_size, self_conditioning_size, bias=False)
        self.down_proj = nn.Linear(self_conditioning_size, hidden_size, bias=False)

    def forward(self, inputs_embeds: torch.Tensor, soft_embeds: torch.Tensor) -> torch.Tensor:
        # 对 soft_embeds 做 pre-norm，然后 gated MLP，残差连接到 inputs_embeds
        x = self.pre_norm(soft_embeds)
        sc_signal = self.down_proj(
            F.gelu(self.gate_proj(x), approximate="tanh") * self.up_proj(x)
        )
        return self.post_norm(inputs_embeds + sc_signal)


@torch.compile(dynamic=True)
def _softcap_logits(logits: torch.Tensor, cap: float) -> torch.Tensor:
    # Soft-cap logits: 先转 fp32 做 tanh，再乘回 cap，保证数值稳定性
    # @torch.compile 可将 cast/div/tanh/mul 融合为单个 elementwise kernel
    return torch.tanh(logits.float() / cap) * cap
```

vllm/v1/spec_decode/metrics.py

为扩散模型添加专用度量记录和 Prometheus 计数器，复用推测解码的数据路径但以扩散语义输出。

```
class SpecDecodingLogging:
    def __init__(self, is_diffusion: bool = False):
        # Diffusion (dLLM) 模型复用 spec-decode 数据路径但使用扩散术语
        self.is_diffusion = is_diffusion
        self.reset()

    def log(self, log_fn=logger.info):
        if not self.num_drafts:
            return
        # ... 计算 num_drafts, num_draft_tokens, num_accepted_tokens ...
        if self.is_diffusion:
            # 扩散模型：使用扩散专用术语记录度量
            self._log_diffusion(
                log_fn,
                num_denoising_steps=num_drafts,
                num_canvas_tokens=num_draft_tokens,
                num_committed_tokens=num_accepted_tokens,
                committed_throughput=accepted_throughput,
```

```
            )
            self.reset()
            return
        # 否则按原有推测解码逻辑记录
        # ...

    def _log_diffusion(self, log_fn, num_denoising_steps, num_canvas_tokens, num_committed_
    tokens, committed_throughput):
        # 每个 "draft" 是一次去噪步骤，重新评估 canvas block 并提交部分 token
        mean_committed_per_step = (
            num_committed_tokens / num_denoising_steps if num_denoising_steps > 0 else
            float("nan")
        )
        mean_steps_per_canvas = (
            num_canvas_tokens / num_committed_tokens if num_committed_tokens > 0 else
            float("nan")
        )
        log_fn(
            "DiffusionDecoding metrics: "
            "Committed token throughput: %.2f tokens/s, "
            "Mean denoising steps per canvas: %.2f, "
            "Mean tokens committed per denoising step: %.2f, "
            "Committed: %d tokens, Denoising steps: %d, Canvas positions evaluated: %d",
            committed_throughput, mean_steps_per_canvas, mean_committed_per_step,
            num_committed_tokens, num_denoising_steps, num_canvas_tokens,
        )
```

评论区精华

- 硬编码配置讨论：benchislett 质疑硬编码 canvas_length 和默认 max_num_seqs=8 的合理性，LucasWilkinson 回应将作为快速跟进优化。
- 注意力后端配置：benchislett 建议设立专用的混合因果注意力后端配置项，LucasWilkinson 同意作为快速跟进。
- GB10 Triton 共享内存溢出：jasongwartz 报告 DGX Spark (SM120) 上 Triton 内核共享内存不足，MatthewBonanni 通过 is_device_capability_family(100) 修复。
- FP32 softcap 精度：benchislett 询问 FP32 softcap 是否严格必要，LucasWilkinson 确认是 HF 参考实现的精度要求。
- 工具解析器批量支持：bbrowning 询问增强原因，LucasWilkinson 解释 DiffusionGemma 每次发射 256 tokens 需要批量工具调用处理。
- 模型注册表占位符：benchislett 指出使用临时仓库 gg-hf-st/...，MatthewBonanni 修复为官方仓库。
  - 硬编码 canvas_length 和默认 max_num_seqs (design): LucasWilkinson 表示将跟进优化，当前为临时方案。
  - 注意力后端配置支持混合因果性 (design): LucasWilkinson 同意作为快速跟进，当前满足功能需求。

- GB10 Triton 共享内存溢出 (performance): MatthewBonanni 修复为 is_device_capability_family(100)，jasongwartz 确认修复有效。
- FP32 softcap 精度要求 (performance): LucasWilkinson 确认 FP32 是 HF 参考实现的精度要求，以保证数值稳定性。
- 工具解析器批量支持 (design): 确认增强必要性，LucasWilkinson 欢迎进一步改进。
- 模型注册表占位符 (other): MatthewBonanni 修复为 'google/diffusiongemma-26B-A4B-it'。

风险与影响

- 风险：
  - 混合因果注意力后端依赖：必须使用 FLASH_ATTN 或 TRITON_ATTN，FlashInfer 会直接报错退出，用户需要手动指定。
  - 默认 max_num_seqs=8 限制：保守配置可能限制吞吐量，但防止了多次去噪步骤的 OOM。
  - FP32 softcap 内存开销：扩散采样器产生 [seq_len, canvas_length, vocab] 的 FP32 张量，内存占用大，大 batch 下有风险。
  - 工具解析器兼容性：对 Gemma4 原有工具调用行为的影响通过 supports_required_and_named=False 保持向后兼容。
  - HF 临时仓库依赖：模型配置指向 HF 临时仓库 gg-hf-st，可能不稳定，合并时已修正为官方 google/diffusiongemma-26B-A4B-it。
  - 推测解码路径复用：SpecDecodingLogging 中扩散分支与原生分支共享统计，需确保 num_sampled_tokens_per_step == 0 等条件正确，避免空 token 问题（修复历史显示曾出现回归）。
- 影响：
  - 用户影响：用户可以运行 DiffusionGemma 进行文本和图像推理，需指定 --attention-backend TRITON_ATTN 或 FLASH_ATTN，建议使用 Docker 镜像 vllm-openai:gemma。
  - 系统影响：引入了扩散模型架构概念（diffusion_config、扩散采样器），扩展了 V1 引擎的模型类型。新增了 1363 行核心代码和 52 个变更文件，增加了构建和测试时间。
  - 团队影响：需要维护 DiffusionGemma 的后续更新和优化，该实现可能作为未来其他块扩散模型的基础。
  - 风险标记：混合因果注意力依赖特定后端，默认 max_num_seqs 限制吞吐量，FP32 softcap 内存开销大，工具解析器兼容性，依赖 HF 临时仓库

关联脉络

- PR #45376 [Bugfix] Set type/role explicitly in streaming message_start event: 该 PR 修复了 Anthropic 流式响应中缺失 type/role 字段的问题，使 DiffusionGemma 与 Claude Code 等 Anthropic SDK 兼容（评论中 bbrowning 提及）。

- PR #45396 [Frontend] Support strict mode for tool calling with ResponsesAPI: 该 PR 引入工具调用严格模式，本 PR 的 gemma4 工具解析器增强与之协同，确保 DiffusionGemma 的工具调用与最新前端兼容。