PR #23198 完整报告

sgl-project/sglang

[diffusion] Fix --warmup-resolutions hang with --enable-cfg-parallel

合并时间：2026-04-23 13:39

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23198

执行摘要

- 一句话：修复 CFG-parallel 模式下 warmup 分辨率预热导致的 30 分钟静默挂起。
- 推荐动作：建议精读 scheduler.py 和 input_validation.py 的变更，关注设计决策如占位符常量的使用和验证逻辑的添加，这些体现了防御性编程和代码可维护性的权衡。

功能与动机

根据 PR body 描述，修复一个 30 分钟的静默挂起问题，当使用 --enable-cfg-parallel --warmup --warmup-resolutions 启动时。根因是合成的 warmup Req 没有正确设置 do_classifier_free_guidance，导致 CFG-parallel 模式下 rank 1 跳过正向传递，最终在 scheduler.step() 中崩溃，并被 gloo 广播超时掩盖。

实现拆解

1. 修改 scheduler 预热请求合成逻辑：在 python/sglang/multimodal_gen/runtime/managers/scheduler.py 的 prepare_server_warmup_reqs 方法中，当 enable_cfg_parallel 为 True 时，设置 do_classifier_free_guidance=True 和 negative_prompt=DEFAULT_PLACEHOLDER_PROMPT，确保 rank 1 有有效工作；当 CFG-parallel 关闭时，保持原有行为不变。
2. 添加输入验证防护：在 python/sglang/multimodal_gen/runtime/pipelines_core/stages/input_validation.py 的 forward 方法中，新增检查：如果服务器启用 cfg-parallel 但请求未启用 CFG，则抛出 ValueError，提供详细错误信息，防止后续挂起。
3. 新增单元测试覆盖：创建 python/sglang/multimodal_gen/test/unit/test_cfg_parallel_warmup.py，包含四个 CPU-only 测试，验证 scheduler 修复和输入验证防护的正确性，确保回归防护。

关键文件：

- python/sglang/multimodal_gen/runtime/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 prepare_server_warmup_reqs）：核心调度器文件，修改了预热请求合成逻辑，直接修复挂起问题。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/input_validation.py（模块 输入验证；类别 source；类型 core-logic；符号 forward）：输入验证阶段文件，新增防护逻辑，防止非 CFG 请求在 cfg-parallel 服务器上导致挂起。

- python/sglang/multimodal_gen/test/unit/test_cfg_parallel_warmup.py（模块 测试套件；类别 test；类型 test-coverage；符号 _make_bare_scheduler, _make_input_validation_stage, _make_validation_server_args, TestWarmupReqCfgParallel）：新增单元测试文件，覆盖 scheduler 修复和输入验证防护，确保代码正确性和回归防护。

关键符号：prepare_server_warmup_reqs, forward

关键源码片段

python/sglang/multimodal_gen/runtime/managers/scheduler.py

核心调度器文件，修改了预热请求合成逻辑，直接修复挂起问题。

```
# 在 scheduler.py 中，新增模块级常量并修改 prepare_server_warmup_reqs 方法
# Placeholder negative_prompt used in synthesized warmup Reqs when --enable-cfg-parallel is on.
# A non-empty, real word (vs "" or " ") so every tokenizer backend emits a predictable,
# non-degenerate token sequence — rank 1's uncond branch then produces a valid tensor.
DEFAULT_PLACEHOLDER_PROMPT = "warmup"

def prepare_server_warmup_reqs(self):
    if self.server_args.warmup and not self.warmed_up and self.server_args.warmup_resolutions is not None:
        self._warmup_total = len(self.server_args.warmup_resolutions)
        self._warmup_processed = 0
        task_type = self.server_args.pipeline_config.task_type
        requires_warmup_image = task_type.accepts_image_input()
        warmup_input_path = None
        if requires_warmup_image:
            warmup_input_path = self._prepare_shared_warmup_image_path()

        for resolution in self.server_args.warmup_resolutions:
            width, height = _parse_size(resolution)
            # CFG-parallel splits cond/uncond across ranks, so rank 1 needs a real uncond pass.
            # Force do_classifier_free_guidance + non-empty negative_prompt when cfg-parallel is on,
            # so the synthesized warmup Req exercises both ranks' denoising paths.
            # When cfg-parallel is off, the Req construction is byte-identical to the pre-fix behavior.
            req_kwargs = dict(
                data_type=task_type.data_type(),
                width=width,
                height=height,
                prompt="",
            )
            if requires_warmup_image:
                req_kwargs["negative_prompt"] = ""
                req_kwargs["image_path"] = [warmup_input_path]
            if self.server_args.enable_cfg_parallel:
                req_kwargs["negative_prompt"] = DEFAULT_PLACEHOLDER_PROMPT # 使用全局常量
```

```
                req_kwargs["do_classifier_free_guidance"] = True # 确保启用 CFG
            req = Req(**req_kwargs)
            req.set_as_warmup(self.server_args.warmup_steps)
            self.waiting_queue.append((None, req))
        self.warmed_up = True
```

python/sglang/multimodal_gen/runtime/pipelines_core/stages/input_validation.py

输入验证阶段文件，新增防护逻辑，防止非 CFG 请求在 cfg-parallel 服务器上导致挂起。

```
# 在 input_validation.py 的 forward 方法中，新增验证逻辑
# Reject requests that do not enable CFG on a server launched with --enable-cfg-parallel.
# CFG-parallel splits cond/uncond across ranks, so rank 1 has no work and returns None for
noise_pred,
# which crashes scheduler.step() ~30 minutes later under a gloo broadcast timeout.
if server_args.enable_cfg_parallel and not batch.do_classifier_free_guidance:
    neg_prompt_state = (
        "not set"
        if batch.negative_prompt is None
        else "empty" if batch.negative_prompt == "" else "set"
    )
    raise ValueError(
        f"Server was launched with --enable-cfg-parallel but this request does not use classifier-
        free guidance "
        f"(do_classifier_free_guidance={batch.do_classifier_free_guidance}, guidance_scale=
        {batch.guidance_scale}, "
        f"true_cfg_scale={batch.true_cfg_scale}, negative_prompt={neg_prompt_state}). "
        f"CFG-parallel splits cond/uncond across ranks and requires both to be active. "
        f"Either disable --enable-cfg-parallel or ensure the request enables CFG "
        f"(set guidance_scale > 1.0 or true_cfg_scale > 1.0, with a non-empty negative_prompt or
        negative_prompt_embeds)."
    ) # 提供详细错误信息，帮助用户调试
```

评论区精华

review 中主要讨论了 negative_prompt 占位符的设计选择。mickqian 询问占位符是否为 typo，mispa-ms 解释使用真实单词（如 'warmup'）是为了确保 tokenizer 产生可预测序列，避免退化。最终采纳 mickqian 的建议，提取为全局常量 DEFAULT_PLACEHOLDER_PROMPT，提升代码可维护性。

- 占位符常量设计 (design): 提取 DEFAULT_PLACEHOLDER_PROMPT 常量，提升代码可维护性和一致性。

风险与影响

- 风险：
  1. 回归风险：修改了 scheduler 的核心预热逻辑，可能影响其他 warmup 场景，但新增测试覆盖了关键路径。

2. 兼容性风险：输入验证新增拒绝非 CFG 请求，可能影响之前允许的配置，但这是防御性措施，且错误信息清晰。
3. 性能风险：无显著性能影响，修复确保 CFG-parallel 正确工作，避免资源浪费。 - 影响：对用户：修复了挂起问题，提升服务可靠性和用户体验；对系统：确保 CFG-parallel 模式在 diffusion 模块中正常运行，避免 30 分钟超时和崩溃；对团队：提供了单元测试和清晰验证逻辑，便于后续维护和扩展。 - 风险标记：核心路径变更，输入验证增强

关联脉络

- 暂无明显关联 PR