PR #28559 完整报告

sgl-project/sglang

fix: speculative draft worker clobbering target attention backend

合并时间：2026-06-18 16:21

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28559

执行摘要

- 一句话：修复推测解码 draft worker 覆盖 target 注意力后端
- 推荐动作：值得精读。此 PR 展示了如何解决一个典型的全局状态污染 bug——通过将 per-process 配置后移至 per-object 隔离，同时保持了与旧配置的向下兼容。设计模式对于多 worker 或流水线并行的模块都具参考价值。

功能与动机

Issue #28528 报告了 B300 kimi-k2.5 + dflash 在有 prefix cache 时崩溃的根本原因：target runner 初始化后 global server_args 中记录了 mla 后端，但 draft runner 初始化将其覆盖为 flashinfer，导致 target 前向时选用了错误的 flashinfer MLA 核，触发 head dimension 断言失败。PR body 明确指出此反模式依赖多 runner 初始化顺序，需要重构。

实现拆解

1. 移除全局变量写入：在 model_runner.py 的 _get_attention_backend() 中删除对 get_global_server_args().prefill_attention_backend 和 .decode_attention_backend 的写入，避免跨 worker 污染。
2. 在 backend 对象上记录 per-runner 字符串：init_attention_backend() 完成后将 prefill_attention_backend_str 和 decode_attention_backend_str 写入 self.attn_backend 对象（新增属性）。对于 draft worker（指定 speculative_draft_attention_backend），其两个字符串均设为该 draft 后端值。
3. 在 AttentionBackend 基类声明属性：base_attn_backend.py 中新增类属性 prefill_attention_backend_str 和 decode_attention_backend_str，缺省为 None。
4. 模型 forward 改为读取 backend 对象：deepseek_v2.py 的 dispatch_attn_forward_method() 通过 get_attn_backend() 获取当前 runner 的 backend，优先使用其上的 prefill/decode_attention_backend_str，若为 None 则 fallback 到 server_args.get_attention_backends() 的默认值。
5. 新增 get_attn_backend 工具函数：从 forward_context.py 导出，用于模型代码获取当前 forward 的 attention backend 实例。

关键文件：

- python/sglang/srt/models/deepseek_v2.py（模块 模型层；类别 source；类型 core-logic；符号 dispatch_attn_forward_method）：核心修复文件，

dispatch_attn_forward_method 改为从 backend 对象读取后端字符串而非全局变量。

- python/sglang/srt/model_executor/model_runner.py（模块 模型运行器；类别 source；类型 data-contract；符号 init_attention_backend, _get_attention_backend）：修改 _get_attention_backend 移除全局写入，并在 init_attention_backend 中将字符串 stamp 到 backend 对象上。
- python/sglang/srt/layers/attention/base_attn_backend.py（模块 注意力层；类别 source；类型 configuration；符号 AttentionBackend）：为 AttentionBackend 基类新增两个类属性，作为 per-runner 字符串的存储位置。

关键符号：dispatch_attn_forward_method, init_attention_backend, _get_attention_backend

关键源码片段

python/sglang/srt/models/deepseek_v2.py

核心修复文件，dispatch_attn_forward_method 改为从 backend 对象读取后端字符串而非全局变量。

```
# deepseek_v2.py 中 dispatch_attn_forward_method 的修改
# 核心思路：每个 runner 的 attention backend 对象上记录它自己的
# prefill/decode 后端字符串，不再依赖全局 server_args 中的可变字段。

def dispatch_attn_forward_method(
    self, forward_batch: ForwardBatch
) -> AttnForwardMethod:
    # 获取当前 forward 的 attention backend 实例
    # 该实例在 init_attention_backend 时已被写入 per-runner 字符串
    backend = get_attn_backend()
    server_args = get_global_server_args()
    # server_args 提供默认值，但实际优先使用 backend 上的记录
    default_prefill_str, default_decode_str = server_args.get_attention_backends()
    prefill_backend_str = (
        backend.prefill_attention_backend_str or default_prefill_str
    )
    decode_backend_str = backend.decode_attention_backend_str or default_decode_str

    if forward_batch.forward_mode.is_decode_or_idle():
        attention_backend = decode_backend_str
    elif (
        forward_batch.forward_mode.is_target_verify()
        or forward_batch.forward_mode.is_draft_extend_v2()
    ):
        if server_args.speculative_attention_mode == "decode":
            attention_backend = decode_backend_str
        else:
            attention_backend = prefill_backend_str
    else:
        attention_backend = prefill_backend_str
```

```
    self.current_attention_backend = attention_backend
    handler = AttentionBackendRegistry.get_handler(attention_backend)
    return handler(self, forward_batch)
```

python/sglang/srt/model_executor/model_runner.py

修改 _get_attention_backend 移除全局写入，并在 init_attention_backend 中将字符串 stamp 到 backend 对象上。

```
# model_runner.py 中 init_attention_backend 的尾部新增
# 以及 _get_attention_backend 的改动

def init_attention_backend(self):
    # ... 原有的分支逻辑（pdmux / TBO / 默认） ...
    # 全部走完 _get_attention_backend 后，将 per-runner 的字符串 stamp 到 backend 上
    self.attn_backend.prefill_attention_backend_str = (
        self.prefill_attention_backend_str
    )
    self.attn_backend.decode_attention_backend_str = (
        self.decode_attention_backend_str
    )

def _get_attention_backend(self, init_new_workspace: bool = False):
    # ... 对于 draft worker 特殊分支
    if self.is_draft_worker and draft_attn_backend:
        # draft worker 只有一个后端，prefill 和 decode 都用同一字符串
        self.prefill_attention_backend_str = draft_attn_backend
        self.decode_attention_backend_str = draft_attn_backend
        return self._get_attention_backend_from_str(
            draft_attn_backend, init_new_workspace=init_new_workspace,
        )
    # ... 原有 hybrid 判断逻辑不变 ...
    # 删除以下全局写入:
    # (get_global_server_args().prefill_attention_backend,
    # get_global_server_args().decode_attention_backend) = (...)
    return attn_backend
```

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

为 AttentionBackend 基类新增两个类属性，作为 per-runner 字符串的存储位置。

```
# base_attn_backend.py 中 AttentionBackend 类的声明
class AttentionBackend(ABC):
    # ... 原有文档和结构 ...

    # 新增类属性：由 ModelRunner.init_attention_backend 在构造后设置
    # 每个 runner 记录的 prefill/decode 后端字符串，
    # 用于 dispatch_attn_forward_method 读取，
    # 避免跨 runner 的全局变量污染。
    prefill_attention_backend_str: Optional[str] = None
    decode_attention_backend_str: Optional[str] = None
```

评论区精华

该 PR 的 review 评论和讨论较少（共 4 条 comment，主要是 CI rerun 和 bot 消息），未出现深度技术争议。PR body 已清晰解释了 root cause 和 fix 方向。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。核心变更在于将后端字符串的传播方式从全局变量改为 backend 对象属性，并通过 get_attn_backend() 函数获取实例，路径透明可控。主要风险点：
  - 若其他模型或代码路径仍依赖旧的全局变量（如直接引用 server_args.prefill_attention_backend），可能产生未同步问题。但 PR body 指出 --speculative-draft-attention-backend 路径不受影响，且本次删除了全局写入，其他读取点可能需要并行审计。
  - get_attn_backend() 实现依赖正确的前向上下文初始化，若在上下文未设置时调用可能返回 None，当前已通过 fallback 到默认值缓解。
  - 影响：影响范围集中在 speculative decoding（dflash）场景下的 DeepSeek 系列模型（以及同样使用 dispatch_attn_forward_method 的模型）。非 spec decode 场景行为不变。用户无需修改启动参数，修复向后兼容。
  - 风险标记：核心路径变更，缺少测试覆盖，依赖初始化顺序

关联脉络

- PR #28528 [Bug] B300 kimi-k2.5 + dflash crash when there is prefix cache: 本 PR 修复的 issue，与 #28528 中的崩溃现象直接关联。