PR #25748 完整报告

sgl-project/sglang

loader: yield filtered MTP weights lazily to avoid OOM hang on multi-layer EAGLE

合并时间：2026-05-20 12:33

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25748

执行摘要

- 一句话：延迟 MTP 权重过滤修复 OOM 挂起
- 推荐动作：值得精读的 bugfix 典范：一行的逻辑错误（tuple 强制物化）导致整个系统在特定配置下不可用，修复后效果显著。代码改动虽小，但不熟悉迭代器模型的人容易犯同样错误。

功能与动机

PR body 明确指出：多层级 EAGLE 下每个 draft runner 独立全扫描检查点，tuple(filtered_weights) 持有全部前缀重命名的张量，TP=8 时 3 个 runner 的活跃张量导致主机触发 page reclaim，所有调度线程卡在 futex_do_wait，read_bytes=0，服务无法就绪。

实现拆解

1. 修改函数签名和返回值：将 _filter_mtp_weights 的返回值从 Tuple[Tuple[str, torch.Tensor], ...] 改为 Generator[Tuple[str, torch.Tensor], None, None]，使其成为一个生成器。
2. 移除列表并改用 yield：删除 filtered_weights = [] 和 filtered_weights.append(...)，改为直接 yield (prefix + new_name, tensor)，这样每个张量在被消费后才释放，不会全部积压在内存中。
3. 保留上游行为一致性：该函数的所有调用者（位于 _get_weights_iterator 中）已经通过 return self._filter_mtp_weights(...) 将迭代器返回给上层，上层本就以迭代方式消费，因此改为生成器后完全兼容。此外，同文件的 _get_all_weights 和 _get_weights_iterator 的非 MTP 路径已经使用生成器表达式，风格统一。
4. 文档更新：更新了 docstring 以解释惰性 yield 避免 OOM 的原因。

关键文件：

- python/sglang/srt/model_loader/loader.py（模块 模型加载器；类别 source；类型 data-contract；符号 _filter_mtp_weights）：唯一变更文件，包含核心修复：将 _filter_mtp_weights 从返回 tuple 改为生成器，避免内存爆炸。

关键符号：_filter_mtp_weights, _get_weights_iterator

关键源码片段

python/sglang/srt/model_loader/loader.py

唯一变更文件，包含核心修复：将 _filter_mtp_weights 从返回 tuple 改为生成器，避免内存爆炸。

```
# python/sglang/srt/model_loader/loader.py
# 关键函数：_filter_mtp_weights
# 原实现：filtered_weights = [] 后用 append 收集全部张量，最后 return tuple(...) 一次性物化。
# 新实现：使用 yield 逐项生成，让上游 buffered_multithread_safetensors_weights_iterator
# 的滑动窗口 buffer 真正生效，避免大 MoE 检查点时 CPU 内存爆炸。

@classmethod
def _filter_mtp_weights(
    cls, weights_iterator, prefix: str, draft_model_idx: int
) -> Generator[Tuple[str, torch.Tensor], None, None]:
    """Filter MTP weights to keep only the specified draft model layer
    and remap it to layer 0.  Yields lazily so the upstream buffered
    iterator 's sliding window actually bounds CPU memory — eager
    materialization caused page-reclaim hangs on large MoE checkpoints
    with multi-layer EAGLE."""
    for name, tensor in weights_iterator:
        match = cls._MTP_PATTERN.match(name)
        if match is not None:
            idx = int(match.group(1))
            if idx != draft_model_idx:
                continue
            # 将 MTP 层编号重映射为 layer 0
            new_name = name.replace(match.group(), "model.mtp.layers.0.")
        else:
            new_name = name
        # 关键变更：以前是 filtered_weights.append(...),
        # 最后 return tuple(filtered_weights);
        # 现在直接 yield，每个权重被消费后即可释放。
        yield (prefix + new_name, tensor)
```

评论区精华

- 暂无高价值评论线程

风险与影响

- 风险：风险极低。
- 回归：调用者已经以迭代方式消费结果，且同文件其他路径早已使用生成器，行为不变。
- 性能：惰性 yield 避免了全量拷贝，理论上减少内存和延迟。
- 兼容性：仅修改内部函数签名和实现，对外可见的接口未变。
- 影响：影响范围限定在 DefaultModelLoader._filter_mtp_weights 函数。主要受益场景：使用多层级 EAGLE 且模型权重很大的用户（如 MiMo-V2.5-Pro 等大型 MoE 模型），服务启动时间从无限挂起降至数分钟，且多线程加载器可以正常工作。对于单个 draft runner 或

小模型，影响微乎其微。

- 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #25774 drop output ids: 同属调度模块重构，涉及迭代器 / 生成器模式的使用，且修改了同一个类 DefaultModelLoader 的周边方法。
- PR #25532 [fp8] SM90 swap-AB scaled_mm dispatch (~1.16x kernel geomean, +5.8-18.5% end-to-end): 性能优化方向，两者都关注大模型部署时的内存和计算效率。
- PR #25825 [Refactor] Pass PP start_layer via model constructor instead of forward_batch.token_to_kv_pool: 同样涉及模型加载和调度模块的接口变更，共享文件层级结构。