PR #28956 完整报告

sgl-project/sglang

Pack aux hidden states into a preallocated buffer

合并时间：2026-07-28 19:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28956

执行摘要

- 一句话：预分配缓冲消除 Dflash 辅隐藏状态 HBM 尖峰
- 推荐动作：值得精读：AuxHiddenStatePacker 的设计模式可推广到其他需要累积张量并减少内存峰值的场景。关注 copies_on_append 属性的使用和兼容性机制。

功能与动机

Capturing aux hidden states (Eagle3/DFlash) has a transient ~2x HBM spike: the model collects K separate [tokens, hidden] tensors and LogitsProcessor concatenates them into [tokens, K * hidden], holding both at once.

实现拆解

1. 新建 aux_hidden_states.py 定义 AuxHiddenStatePacker 类和 AuxHiddenStates 等类型别名。Packer 通过 append 将每层捕获写入预分配 [tokens, K * hidden] 缓冲，finalize 返回 packed tensor。
2. 修改 communicator.py 中的 prepare_attn_and_capture_last_layer_outputs 方法，参数类型从 Optional[List[torch.Tensor]] 改为 Optional[AuxHiddenStateAccumulator]，并利用 copies_on_append 属性避免不必要的克隆。
3. 修改 logits_processor.py，导入 AuxHiddenStates 和 pack_aux_hidden_states，将 aux_hidden_states 参数类型改为 Optional[AuxHiddenStates]，并在 _get_pruned_states 中处理 packed tensor 和 list 两种形式。
4. 修改 deepseek_v2.py，导入并使用 AuxHiddenStatePacker 替换列表，初始化时传入层数，最后调用 finalize 返回 packed buffer。
5. 全量修改类型标注以支持两种表示共存，未迁移的模型继续使用旧列表路径。

关键文件：

- python/sglang/srt/layers/aux_hidden_states.py（模块 捕获管理；类别 source；类型 core-logic；符号 AuxHiddenStatePacker, init, append, len）：新增核心文件，定义了 AuxHiddenStatePacker 类和类型别名，是本次性能优化的核心实现。
- python/sglang/srt/layers/logits_processor.py（模块 Logits 处理；类别 source；类型 dependency-wiring；符号 LogitsProcessor.forward, LogitsProcessor._get_pruned_states）：修改 aux_hidden_states 参数类型并调整 _get_pruned_states 以兼容 packed 和 list 两种输入。

- python/sglang/srt/models/deepseek_v2.py（模块 DeepSeek V2 模型；类别 source；类型 data-contract；符号 DeepseekV2Model.forward）：在 DeepSeek-V2 前向中使用 AuxHiddenStatePacker 替换列表收集捕获，并调用 finalize 返回 packed 结果。
- python/sglang/srt/layers/communicator.py（模块 通信层；类别 source；类型 dependency-wiring；符号 LayerCommunicator.prepare_attn_and_capture_last_layer_outputs）：修改 prepare_attn_and_capture_last_layer_outputs 方法，支持 AuxHiddenStateAccumulator 参数并利用 copies_on_append 跳过不必要的 clone。

关键符号：AuxHiddenStatePacker.init, AuxHiddenStatePacker.append, AuxHiddenStatePacker.finalize, pack_aux_hidden_states, LogitsProcessor.forward, LogitsProcessor._get_pruned_states, DeepseekV2Model.forward, LayerCommunicator.prepare_attn_and_capture_last_layer_outputs

关键源码片段

python/sglang/srt/layers/aux_hidden_states.py

新增核心文件，定义了 AuxHiddenStatePacker 类和类型别名，是本次性能优化的核心实现。

```
"""Aux hidden states captured for Eagle3/DFlash draft models."""

from typing import List, Optional, Union

import torch

# 两种表示共存：已迁移到 AuxHiddenStatePacker 的模型传递一个 packed [tokens, K * hidden] tensor,
# 其余模型仍然传递一个 K 个 tensor 的列表。
AuxHiddenStates = Union[torch.Tensor, List[torch.Tensor]]


class AuxHiddenStatePacker:
    """替换模型收集 Eagle3/DFlash 捕获时使用的 `[]` 列表。

    每个 `.append()` 直接写入预分配的 `[tokens, K * hidden]` 缓冲区，
    避免列表路径在 `torch.cat` 时产生的瞬时 ~2x HBM。
    假设所有捕获共享 leading shape 和 feature size。
    """

    # `append` 会拷贝，因此生产者无需为后续 mutate 而 clone。
    copies_on_append = True

    def __init__(self, num_captures: int) -> None:
        self._num_captures = int(num_captures)
        self._buffer: Optional[torch.Tensor] = None
        self._feature_size: Optional[int] = None
        self._idx = 0

    def append(self, hidden: torch.Tensor) -> None:
```

```
        feature_size = int(hidden.shape[-1])
        if self._buffer is None:
            # 第一次 append 时根据 hidden 的形状和总捕获次数分配完整缓冲区
            self._feature_size = feature_size
            self._buffer = hidden.new_empty(
                (*hidden.shape[:-1], feature_size * self._num_captures)
            )
        start = self._idx * self._feature_size
        # 将当前捕获的内容拷贝到预分配缓冲区中的对应位置
        self._buffer[..., start : start + self._feature_size].copy_(hidden)
        self._idx += 1

    def __len__(self) -> int:
        return self._idx

    def finalize(self) -> torch.Tensor:
        """返回 packed 缓冲区；调用者应通过 `len()` 预先处理空情况。"""
        assert (
            self._buffer is not None and self._idx == self._num_captures
        ), f"captured {self._idx} of {self._num_captures} aux hidden states"
        return self._buffer


# 模型向下游捕获路径传递的类型：普通列表或就地写入的 packer
AuxHiddenStateAccumulator = Union[List[torch.Tensor], AuxHiddenStatePacker]


def pack_aux_hidden_states(aux_hidden_states: AuxHiddenStates) -> torch.Tensor:
    """统一将两种形式转换为 packed tensor（若已是 tensor 则直接返回）。"""
    if isinstance(aux_hidden_states, torch.Tensor):
        return aux_hidden_states
    return torch.cat(aux_hidden_states, dim=-1)
```

评论区精华

未产生实质性代码审查讨论；合并者 hnyls2002 对原始提交进行了重构（提取独立模块、精简注释、增加断言），并通过 CI rerun 验证了测试通过。

- 向后兼容的接口设计 (design): 采纳，未迁移模型无感知。

风险与影响

- 风险：
  1. 若 num_captures 与实际 append 次数不一致，finalize 断言会抛出异常，开发阶段可及早暴露错误。
  2. 旧列表路径完全保留，未迁移模型无影响。
  3. copies_on_append 属性可能被误用：若 accumulator 不拷贝，调用者必须保证后续不会修改已追加的张量。

4. 仅限于 DeepSeek-V2，其他模型如使用类似捕获路径未更改，但后续可迁移。 - 影响：对 DeepSeek-V2 用户：减少推理过程中的峰值显存，降低 OOM 风险，尤其长序列场景。接口向后兼容，无需修改配置。对系统：捕获路径的代码可读性略有提升（类型标注更精确）。对团队：为 Eagle3/DFlash 的进一步优化奠定基础。 - 风险标记：内存峰值优化，向后兼容，断言检查

关联脉络

- 暂无明显关联 PR