PR #32315 完整报告

sgl-project/sglang

[AMD] Speed up DSV4 MoE weight loading from mmap views

合并时间：2026-08-02 14:47

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32315

执行摘要

- 一句话：AMD MoE 权重加载提速：H2D 前紧凑化 mmap view，加载快 5.6x
- 推荐动作：值得精读。核心是理解 mmap-backed storage 与 torch view 对 H2D 拷贝性能的影响，以及如何用最小侵入的开关 + 针对性单测验证。关注 _copy_weight_view_before_h2d 的判断条件设计（storage_offset、untyped_storage().nbytes()）和 CI 分层验证策略。

功能与动机

PR body 明确：DSV4-Pro TP8 每 rank 需要 140,544 次小 MoE H2D 拷贝，而 rank-local tensor view 仍引用 safetensors mmap 的大块存储，导致 TP0/TP7 在 H2D 上花 27-32 分钟、其他 rank 约 4 分钟。目标是在不改变默认行为的前提下，通过可选的 H2D 前紧凑化消除这些超大 storage 带来的额外拷贝开销。

实现拆解

1. 在 python/sglang/srt/layers/moe/fused_moe_triton/layer.py 新增 _copy_weight_view_before_h2d 与 _maybe_copy_weight_view_before_h2d。前者判断 CPU 张量是否满足「连续、storage offset 为 0、storage 恰好等于自身大小」，不满足则 clone(memory_format=torch.contiguous_format)；后者受 SGLANG_MOE_COPY_WEIGHT_VIEWS_BEFORE_H2D 开关控制。
2. 在 _load_w13、_load_w2、_load_per_channel_weight_scale 的 expert_data.copy_(loaded_weight) 前插入 _maybe_copy_weight_view_before_h2d 调用，覆盖路由专家与 per-channel scale 的加载路径。
3. 在 python/sglang/srt/environ.py 注册默认关闭的 EnvBool 环境变量，保持默认行为不变，降低回归风险。
4. 在 AMD 相关 CI workflow（nightly-test-amd-rocm720.yml、pr-test-amd-rocm720.yml、nightly-test-amd.yml）中为 DSV4-flash/pro/pro-mtp、Qwen3.5-397B/235B、GPT-OSS-20B/120B 等大模型任务添加 -e SGLANG_MOE_COPY_WEIGHT_VIEWS_BEFORE_H2D=1，将新路径纳入 nightly 与部分 PR 验证。
5. 新增单元测试 test_copy_weight_views_before_h2d.py，覆盖精确 storage（应跳过）、零 offset 大 storage view、非零 offset view、非连续张量四类场景，并注册到 CPU 基 CI。

关键文件：

- python/sglang/srt/layers/moe/fused_moe_triton/layer.py（模块 权重加载；类别 source；类型 core-logic；符号 _copy_weight_view_before_h2d, _maybe_copy_weight_view_before_h2d, _load_w13, _load_w2）：核心改动：新增 H2D 前紧凑化 helper，并接入 w13/w2/per-channel 三条 MoE 权重加载路径。
- test/registered/unit/layers/moe/test_copy_weight_views_before_h2d.py（模块 单元测试；类别 test；类型 test-coverage；符号 _assert_independent_copy, test_skips_copy_for_exact_storage, test_copies_zero_offset_storage_view, test_copies_nonzero_offset_storage_view）：新增单测覆盖四种 storage/view 场景，验证 helper 行为。
- python/sglang/srt/environ.py（模块 环境配置；类别 source；类型 configuration）：新增 SGLANG_MOE_COPY_WEIGHT_VIEWS_BEFORE_H2D 开关，默认关闭，控制新路径。
- .github/workflows/nightly-test-amd-rocm720.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：在 DSV4-flash/pro/pro-mtp、Qwen3.5、GPT-OSS 等 AMD nightly 任务中启用新环境变量。
- .github/workflows/pr-test-amd-rocm720.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：在 PR CI 的 DSV4 精度任务中启用新路径，便于提前发现问题。
- .github/workflows/nightly-test-amd.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：在默认 AMD nightly 的 Qwen3.5 精度任务中启用。

关键符号：_copy_weight_view_before_h2d, _maybe_copy_weight_view_before_h2d, _load_w13, _load_w2, _load_per_channel_weight_scale

关键源码片段

python/sglang/srt/layers/moe/fused_moe_triton/layer.py

核心改动：新增 H2D 前紧凑化 helper，并接入 w13/w2/per-channel 三条 MoE 权重加载路径。

```
# 判定是否需要独立拷贝：仅当张量非连续、或 storage offset 非零、
# 或底层 storage 大于张量自身（典型是 mmap 大文件上的 view）时才 clone。
def _copy_weight_view_before_h2d(loaded_weight: torch.Tensor) -> torch.Tensor:
    """Copy a CPU tensor view into independent contiguous storage."""
    if loaded_weight.device.type != "cpu":
        return loaded_weight
    tensor_bytes = loaded_weight.numel() * loaded_weight.element_size()
    needs_copy = not (
        loaded_weight.is_contiguous()
        and loaded_weight.storage_offset() == 0
        and loaded_weight.untyped_storage().nbytes() == tensor_bytes
    )
    if not needs_copy:
        return loaded_weight
    return loaded_weight.clone(memory_format=torch.contiguous_format)


def _maybe_copy_weight_view_before_h2d(loaded_weight: torch.Tensor) -> torch.Tensor:
```

```
    # 默认关闭，仅当环境变量开启时启用紧凑化，保证默认路径零行为变化
    if not envs.SGLANG_MOE_COPY_WEIGHT_VIEWS_BEFORE_H2D.get():
        return loaded_weight
    return _copy_weight_view_before_h2d(loaded_weight)


def _load_w13(
    self,
    expert_data: torch.Tensor,
    shard_dim: int,
    shard_id: str,
    loaded_weight: torch.Tensor,
    tp_rank: int,
    is_bias: bool = False,
):
    # ... 经过 tp sharding 与 transpose/narrow 后，loaded_weight 往往是
    # 大 mmap storage 上的小 view；在 expert_data.copy_ 前先紧凑化，
    # 避免 H2D 阶段按底层大 storage 整体搬运。
    loaded_weight = _maybe_copy_weight_view_before_h2d(loaded_weight)
    expert_data.copy_(loaded_weight)
```

test/registered/unit/layers/moe/test_copy_weight_views_before_h2d.py

新增单测覆盖四种 storage/view 场景，验证 helper 行为。

```
import torch

from sglang.srt.layers.moe.fused_moe_triton.layer import (
    _copy_weight_view_before_h2d,
)
from sglang.test.ci.ci_register import register_cpu_ci

register_cpu_ci(est_time=2, suite="base-a-test-cpu")


def _assert_independent_copy(source: torch.Tensor, result: torch.Tensor) -> None:
    # 校验拷贝结果：值相等、新对象、连续、offset 为 0、storage 恰好等于自身大小
    assert torch.equal(result, source)
    assert result is not source
    assert result.is_contiguous()
    assert result.storage_offset() == 0
    assert result.untyped_storage().nbytes() == result.numel() * result.element_size()


def test_skips_copy_for_exact_storage():
    # 已是精确独立 storage 的张量应原样返回，不产生额外拷贝
    source = torch.arange(16).reshape(4, 4)
    assert _copy_weight_view_before_h2d(source) is source

def test_copies_zero_offset_storage_view():
    # 零 offset 但底层 storage 更大的 view 需要拷贝
```

```
    backing = torch.arange(32).reshape(8, 4)
    source = backing.narrow(0, 0, 2)
    assert source.storage_offset() == 0
    assert source.untyped_storage().nbytes() > source.numel() * source.element_size()
    _assert_independent_copy(source, _copy_weight_view_before_h2d(source))


def test_copies_nonzero_offset_storage_view():
    # 非零 offset 的 view 需要拷贝
    backing = torch.arange(32).reshape(8, 4)
    source = backing.narrow(0, 6, 2)
    assert source.storage_offset() != 0
    _assert_independent_copy(source, _copy_weight_view_before_h2d(source))


def test_copies_noncontiguous_tensor():
    # 非连续张量（如 transpose）需要拷贝
    source = torch.arange(16).reshape(4, 4).transpose(0, 1)
    assert not source.is_contiguous()
    _assert_independent_copy(source, _copy_weight_view_before_h2d(source))
```

评论区精华

amd-bot 在 CI 状态回复中强调 “PR CI is incomplete — do not read green as verified”：AMD 的 stage-b/stage-c 因 wait-for-stage-a-amd 超时全部跳过，新路径真实价值只在 nightly 验证；同时指出现有 4 个失败（XPU/NPU/NVIDIA/CPU）都不触及本 PR 路径。yctseng0211 补充 stage-b-test-1-gpu-large-amd 的失败将由 PR#32862 解决。

- PR CI 未覆盖 AMD 路径，AMD stage 全部跳过 (testing): 本 PR 的新路径只通过了 CPU 单测与 nightly 验证，PR CI 绿灯不能视为 AMD 已验证；该 CI 基础设施问题由后续 PR（#32862）修复。
- stage-b-test-1-gpu-large-amd 失败将由 PR#32862 解决 (infra): 该失败与本 PR 改动无关，属于 CI runner 基础设施问题，由独立 PR 修复。

风险与影响

- 风险：启用后每个 MoE 权重 view 会先 CPU clone，新增约 10 秒 wall time 与临时内存峰值；对于本就精确 storage 的张量会跳过。新逻辑只接入了 fused_moe_triton/layer.py 的 _load_w13/_load_w2/_load_per_channel_weight_scale，其他 MoE 后端或非 triton 路径不受影响，效果覆盖不全。untyped_storage().nbytes() 比较对 mmap 大 storage 判断正确，但零维张量等边界未在单测中覆盖。PR CI 的 AMD 阶段全部跳过，本路径在 CI 绿灯下仍未得到 GPU 级验证，依赖 nightly（后续由 PR#32862 保障）。
- 影响：默认关闭，对现有用户无行为变化。对 AMD 平台大模型部署（DSV4-Pro、Qwen3.5-397B/235B、GPT-OSS）启用后模型加载时间大幅缩短（5.6x / 1.9-7.8x），显著减少夜间验证超时；同时为后续所有依赖 mmap 权重加载的大型 MoE 模型提供了可复用的优化开关。团队收益：nightly CI 稳定性提升、runner 占用时间减少。

- 风险标记：默认关闭降低回归面，PR CI 未覆盖 AMD 路径，CPU clone 增加内存峰值，仅覆盖 fused_moe_triton 路径

关联脉络

- PR #32862 （评论提及，标题未知）：评论中 yctseng0211 提到该 PR 将修复 stage-b-test-1-gpu-large-amd 失败，与本 PR 的 CI 状态相关。