PR #34563 完整报告

sgl-project/sglang

[diffusion] Optimize bit-exact H3 reference video ingress

合并时间：2026-08-12 20:02

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34563

执行摘要

- 一句话：H3 参考视频按主机单次解码并共享映射，存储降 87.5%
- 推荐动作：值得精读。重点关注 _decode_reference_video_shared 中“两轮 collective 保证决策一致”的设计：第一轮广播 decode 状态与失败，第二轮广播 mapping 状态与失败，任何异常都整体回退而不是让某个 rank 单独失败；这是多 rank 共享内存类优化中容易踩坑的地方。其次关注 text/visual 两个 stage 对 MINIMAX_H3_PREPARED_REFERENCE_VIDEO_EXTRA_KEY 生命周期的 finally 清理，以及 _MINIMAX_H3_SINGLE_RANK_TEXT_ENCODE_EXTRA_KEY 在 DP 广播路径上禁用共享的细节。

功能与动机

PR body 明确指出："Reference-video preprocessing previously ran the same CPU FFmpeg graph independently on every rank. A five-second 1344×768 reference therefore duplicated hundreds of MiB of transformed RGB pages per worker and repeated the same decode work." 即 CPU 侧的解码与内存占用在多 rank 场景被线性放大，需要将解码结果在主机内共享，同时保持既有的 24 FPS、Lanczos 缩放、RGB24 变换与 bit-exact 输出契约不变。

实现拆解

1. 解码入口重构：minimax_h3_decode_reference_video_frames 不再固定 pipe:1 + subprocess.run(capture_output=True)，新增 share_across_replicas 参数，分别路由到 _decode_reference_video_local 或 _decode_reference_video_shared。ffmpeg 滤镜链（fps、lanczos、setsar=1、rgb24）完全不变，保证位精确输出。
2. 本地解码优化：_decode_reference_video_local 在 Linux 上优先用 os.memfd_create 创建匿名文件描述符，让 ffmpeg 通过 pipe:<fd> 直写，再经 _write_reference_video_to_fd 用 lseek 获取实际写入字节数，mmap(ACCESS_WRITE) 映射为 numpy 数组；这避免了 communicate() 对数百 MiB stdout 的 chunk 聚合与 bytes join。memfd 不可用（如容器 seccomp 禁用）时回退到旧的 pipe:1 路径。
3. 跨 replica 共享解码：_decode_reference_video_shared 在 world_size > 1 且 Linux 且 /proc/self/fd 可用时启用。先经 _all_gather_world_objects 收集各 rank 主机名，_reference_video_host_leader（带 lru_cache）确定每台主机的 leader；leader 用 memfd 解码并发布 /proc/<pid>/fd/<fd> 路径。随后所有 rank（含 leader）打开该路径并

以 mmap(ACCESS_COPY) 映射同一份物理页。整个流程分两轮 collective：第一轮广播 decode 状态与错误，第二轮广播 mapping 状态，任何主机失败或任何 rank 映射失败都会全体抛错或全体回退 local，避免 collective 分叉。

4. 调用方契约与生命周期：text_encoding.py::_encode_ref2va 在 replica_world > 1 且非单 rank 文本编码时启用共享，并通过 _MINIMAX_H3_SINGLE_RANK_TEXT_ENCODE_EXTRA_KEY 标记避免在单 rank 编码路径上无谓开启共享；visual_encoding.py::_encode_reference_video 仅在 video_vae.parallel_tiling 为真时共享。两个 stage 在成功或异常路径都会 pop MINIMAX_H3_PREPARED_REFERENCE_VIDEO_EXTRA_KEY 释放大映射。
5. 测试配套：test_minimax_h3_media.py 新增 test_video_transform_can_share_one_host_decode、test_shared_video_transform_falls_back_when_proc_fd_is_blocked、test_shared_video_transform_propagates_any_host_decode_failure 三个单测，分别覆盖共享解码、/proc 被阻断时回退、跨主机解码失败传播，并复用 FakeGroup + monkeypatch 验证两次 all_gather_object 的执行次数。

关键文件：

- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/minimax_h3/reference_encoding.py（模块 媒体解码；类别 source；类型 data-contract；符号 _decode_reference_video_local, _decode_reference_video_shared, _write_reference_video_to_fd, _all_gather_world_objects）：核心实现文件：新增 memfd + mmap 本地解码路径、跨 replica 共享解码路径，以及两轮 all_gather 保证集体决策一致；同时为 minimax_h3_prepared_reference_videos 增加 share_across_replicas 契约。
- python/sglang/multimodal_gen/test/unit/test_minimax_h3_media.py（模块 媒体测试；类别 test；类型 test-coverage；符号 run, test_video_transform_can_share_one_host_decode, FakeGroup, test_shared_video_transform_falls_back_when_proc_fd_is_blocked）：测试配套：覆盖共享解码、/proc 被阻断时回退、跨主机失败传播三个关键场景，并验证两次 all_gather 的执行次数，是保证集体操作一致性的重要防线。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/minimax_h3/stages/text_encoding.py（模块 文本编码；类别 source；类型 data-contract；符号 MiniMaxH3TextEncodingStage.forward, MiniMaxH3TextEncodingStage.run_grouped_requests, MiniMaxH3TextEncodingStage._encode_ref2va）：调用方契约变更：在文本编码阶段根据 replica world size 与单 rank 编码标记决定是否开启跨 replica 共享，并在异常与单 rank 路径清理 prepared video extra key。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/minimax_h3/stages/visual_encoding.py（模块 视觉编码；类别 source；类型 data-contract；符号 MiniMaxH3VisualEncodingStage.forward, MiniMaxH3VisualEncodingStage._encode_reference_video）：调用方契约变更：视觉编码阶段仅在 video VAE 并行 tiling 时启用共享解码，并在 forward 的 finally 中释放大映射，防止后续 stage 长时间持有数百 MiB 的请求本地视图。

关键符号：minimax_h3_decode_reference_video_frames, _decode_reference_video_local, _decode_reference_video_shared,

_write_reference_video_to_fd, _all_gather_world_objects, _reference_video_host_leader,minimax_h3_prepared_reference_videos, MiniMaxH3TextEncodingStage._encode_ref2va, MiniMaxH3VisualEncodingStage._encode_reference_video

关键源码片段

python/sglang/multimodal_gen/test/unit/test_minimax_h3_media.py

测试配套：覆盖共享解码、/proc 被阻断时回退、跨主机失败传播三个关键场景，并验证两次 all_gather 的执行次数，是保证集体操作一致性的重要防线。

```
@pytest.mark.skipif(not sys.platform.startswith("linux"), reason="requires Linux memfd")
def test_shared_video_transform_propagates_any_host_decode_failure(monkeypatch):
    """任一 host 的 leader 解码失败时，所有 rank 必须同步抛错、不再进入映射 collective"""
    class FakeGroup:
        world_size = 4
        rank_in_group = 0
        cpu_group = object()

    gather_index = 0

    def all_gather_object(outputs, value, **_kwargs):
        nonlocal gather_index
        if gather_index == 0:
            # 第一轮：主机名分布为 host-a x2 + host-b x2
            outputs[:] = ["host-a", "host-a", "host-b", "host-b"]
        else:
            # 第二轮：host-b 的 leader 携带解码失败信息
            outputs[:] = [
                value,
                None,
                (None, 0, "CalledProcessError: remote decode failed"),
                None,
            ]
        gather_index += 1

    monkeypatch.setattr(reference_encoding, "get_world_group", FakeGroup)
    monkeypatch.setattr(torch.distributed, "all_gather_object", all_gather_object)
    monkeypatch.setattr(
        reference_encoding,
        "_write_reference_video_to_fd",
        lambda _command, _fd: 1,
    )
    reference_encoding._reference_video_host_leader.cache_clear()
    try:
        with pytest.raises(RuntimeError, match="remote decode failed"):
            reference_encoding._decode_reference_video_shared(["ffmpeg"])
    finally:
```

```
        reference_encoding._reference_video_host_leader.cache_clear()

    # 失败在共享状态交换后立即解决，任何 rank 都不会进入被跳过的映射 collective
    assert gather_index == 2
```

评论区精华

本 PR 无 Review 评论（comments_count = 0、review_comments_count = 0），由作者直接合入。从实现注释与 PR body 可以提炼出关键设计决策：任一主机解码失败必须在第一轮状态交换后立即传播，使所有 rank 同时退出，绝不进入后续 mapping collective；/proc fd 遍历可能被 hidepid 或容器策略拒绝，此时全体回退到 rank-local 解码，保证优化永远不改变原有请求的成功与否或 RGB 字节（bit-exact 契约）。这些决策通过三个针对性单测固化，测试还断言失败场景 gather_index == 2，即不会多执行一次集体通信。

- 暂无高价值评论线程

风险与影响

- 风险：
  - 集体通信一致性：_decode_reference_video_shared 内部有两次 all_gather_object，要求所有 rank 对 sys.platform、/proc/self/fd 存在性、world_size 的判定完全一致；若环境差异导致部分 rank 进入共享路径、部分进入 local 路径，会发生集体操作挂起。目前条件均为各 rank 应当一致的静态环境判断，风险可控但值得在异构容器环境中验证。
  - 平台与容器兼容性：依赖 memfd_create、/proc/<pid>/fd 跨进程打开、hidepid/ 容器 seccomp 策略。代码对每种失败都提供了回退，但回退本身依赖第一次 collective 成功执行。
  - 映射生命周期：mapping 对象存放在 batch.extra，由 text/visual 两个 stage 的 finally 或异常分支 pop 释放；leader 的 fd 在多个异常出口显式 os.close。若未来新增第三个 RGB 消费方，可能提前释放或重复映射。
  - 内存语义：非 leader 使用 ACCESS_COPY 写时复制映射，若消费方原地修改 frames 不会污染共享页；leader 走 ACCESS_WRITE 则反之，测试中仅验证了可写性，未验证跨 rank 修改隔离。
  - 性能回退：共享路径多出两轮 collective，对小视频或单卡场景反而可能更慢；代码通过 world_size <= 1 与 parallel_tiling 条件规避了无谓开销。
- 影响：
  - 用户侧：MiniMax H3 ref2va 多 GPU 部署的参考视频预处理内存占用显著下降（单条 5 秒 1344×768 参考视频解码存储减少约 2.60 GB，峰值 RSS 减半），降低 OOM 风险；多机多卡下收益更明显，且输出保持 bit-exact。
  - 系统侧：减少 CPU 重复解码与 Python 侧大缓冲拷贝，释放内存带宽；mmap 共享物理页使同一主机的多个 worker 进程只保留一份 RGB 数据。
  - 团队侧：确立了“主机内单次解码 + 跨进程映射 + 集体回退”的模式，为 diffusion 其他视频链路（如 SANA、Cosmos）的 ingress 优化提供了可复用范式；同时新增的 extra key 生命周期约定需要后续开发者在新增多媒体 stage 时注意。

- 风险标记：多 rank 集体通信一致性，memfd 与 /proc 平台依赖，映射生命周期与 fd 泄漏，共享路径回退分支，bit-exact 契约回归风险

关联脉络

- PR #34564 [diffusion] Stream and parallelize bit-exact video output saves: 同为 diffusion bit-exact 视频 I/O 的流式与并行优化，一个优化输出保存、一个优化输入解码，可对照理解 bit-exact 契约在 SGLang Diffusion 中的处理方式。
- PR #34359 [Diffusion] Support native and PEFT MiniMax H3 LoRAs: 该 PR 同样改动 MiniMax H3 reference_encoding 链路与 media 相关测试，属于 H3 ref2va 功能线的延续。
- PR #34412 [Diffusion] Improve bit-exact fusion fallback diagnostics: 同属 bit-exact 主题，关注融合回退行为与诊断，与本 PR 的 fallback 设计有共同的方法论。