PR #34564 完整报告

sgl-project/sglang

[diffusion] Stream and parallelize bit-exact video output saves

合并时间：2026-08-12 20:03

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34564

执行摘要

- 一句话：流式分块转换并并行保存多路视频输出，显存降约 95%、耗时减半
- 推荐动作：值得精读，尤其是 _try_save_cuda_videos_direct 的显存 /CPU 预算启发式和 _sendfile_all 的部分写处理，是典型的「零拷贝流式 + 资源感知并行」设计。建议关注两点：共享 memfd 缓冲在并行线程间的互斥窗口是否覆盖整个使用周期，以及非 Linux 平台回退路径的 CI 覆盖。若后续推广到更多 diffusion 模型，可考虑把 chunk 预算做成可配置项。

功能与动机

PR body 明确说明痛点：旧路径在编码前一次性物化整段视频的浮点乘法结果和两套 uint8 布局，临时显存与 pinned-host 内存随视频时长线性增长；num_outputs_per_prompt > 1 时每路独立结果只能串行编码。实测数据驱动了本次改造：1344×768 下五秒转换临时显存峰值约 1.92 GB，十五秒 RGB24 主机缓冲约 1.13 GB，双输出保存约 1.30 s；改造后分别降到约 108 MB、21.7 MB 与 0.69 s。

实现拆解

1. 分块转换预算：在 python/sglang/multimodal_gen/runtime/entrypoints/utils.py 中新增常量 _MAX_CUDA_VIDEO_CONVERSION_CHUNK_BYTES = 128 MiB 和辅助函数 _cuda_video_conversion_chunk_frames。它按「每帧临时字节 = 3 × H × W × (元素字节数 + 2)」反推每次能安全处理的帧数，从源头控制 CUDA 转换的临时显存峰值，不再让 frames = (video * 255).clamp_(0, 255).to(torch.uint8) 一次性物化整段视频。
2. 流式喂给 ffmpeg：_try_save_cuda_video_direct 的 ffmpeg 输入从 /proc/self/fd/{fd} 改为 pipe:0，进程由 subprocess.run 改为 Popen；新增 _sendfile_all 把 memfd 中的原始帧分块 os.sendfile 到 ffmpeg stdin，并循环处理部分写入。每块转换、拷入、同步 CUDA 流后立即送走，memfd 缓冲只保留一个 chunk。异常路径会 kill 子进程并读取 stderr 转储，随后统一回退 imageio。
3. 并行多路保存：新增 _try_save_cuda_videos_direct，前置校验包括 CUDA 张量、3 通道、.mp4 扩展名、多路同设备；显存侧按最大的两个 chunk 临时字节是否超过当前空闲显存 25% 决定是否并行，CPU 侧按两个 x264 自动线程数之和是否超过 os.sched_getaffinity 可用核数决定。满足条件时用 ThreadPoolExecutor(max_workers=2) 并发执行 save_one，否则返回 None 交由串行路径。
4. 回退与接线：save_outputs 对并行结果中 False 的输出逐一路回退到既有 _try_save_cuda_video_direct 串行路径（再失败才走 post_process_sample），测试用

monkeypatch 验证了 [True, False] 场景下失败项确实走串行且不触发帧物化。

5. MiniMax-H3 校验并行化：.../minimax_h3/video_adapter.py 的 validate_final_outputs_sync 把逐路 ffprobe 改为 ThreadPoolExecutor(max_workers=min(4, len(output_paths))) + pool.map，保持输出索引顺序，不一致时仍按原样报错，媒体元数据探测耗时从约 73–82 ms 降到 38–40 ms。
6. 测试配套：python/sglang/multimodal_gen/test/unit/test_output_saving.py 新增 test_multiple_videos_use_parallel_direct_save_with_serial_fallback，覆盖并行入口调用、路径与 fps 参数传递、失败回退；PR 同时做了 5 秒 /15 秒合成视频字节级比对和 4×H200 MiniMax-H3 双输出 MP4/ 解码 RGB/ 解码 PCM 一致性验证。

关键文件：

- python/sglang/multimodal_gen/runtime/entrypoints/utils.py（模块 输出保存；类别 source；类型 core-logic；符号 _cuda_video_conversion_chunk_frames, _sendfile_all, _try_save_cuda_video_direct, _try_save_cuda_videos_direct）：核心改动文件：实现分块转换、sendfile 流式写入和并行多路保存，是本 PR 的性能与内存优化主体。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/minimax_h3/video_adapter.py（模块 媒体校验；类别 source；类型 data-contract；符号 probe_output, validate_final_outputs_sync）：MiniMax-H3 多输出媒体校验从串行 ffprobe 改为并行探测，保持输出索引顺序与错误语义，零拷验证耗时从约 73–82 ms 降到 38–40 ms。
- python/sglang/multimodal_gen/test/unit/test_output_saving.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_multiple_videos_use_parallel_direct_save_with_serial_fallback, parallel_save）：新增并行直存 + 串行回退的单元测试，用 monkeypatch 验证失败项回退路径、参数传递和不触发帧物化。

关键符号：_cuda_video_conversion_chunk_frames, _sendfile_all, _try_save_cuda_video_direct, _try_save_cuda_videos_direct, save_one, probe_output, validate_final_outputs_sync

评论区精华

本 PR 没有公开的 review 评论（review 与 issue 评论均为 0），评审证据主要来自 PR body 的自测：5 秒 /15 秒合成视频的 MP4 字节级比对、4×H200 MiniMax-H3 Ref2VA 双输出的 MP4 容器 / 解码 RGB/ 解码 PCM 一致性，以及注入 sendfile 失败后的回退与临时文件清理验证。虽然没有评审交锋，但实现中「先算预算再决定是否并行」的做法本身就体现了对 OOM 和 CPU 争用的防御性设计。

- 评审以自测与 CI 为主，无公开评论 (other): 作者用字节级一致性与回退注入测试证明流式路径与旧路径等价，CI 全绿后合入。

风险与影响

- 风险：
  1. 平台与内核依赖：新路径强依赖 os.memfd_create 与 os.sendfile，两者仅在 Linux 可用；_try_save_cuda_video_direct 在缺失时返回 False 走 imageio 回退，非 Linux 不

受影响，但回退路径的实际覆盖需要 CI 验证。

2. OOM 保护是启发式：并行判定只看转换 chunk 的临时字节是否低于空闲显存 25%，没有把 _CudaMemfdVideoBuffer 本身、WAV 文件和 ffmpeg 内部缓冲计入；极端低显存场景下仍可能 OOM。分块预算 128 MiB 也只是按元素大小估算，float32 与 bfloat16 的系数差异会影响实际峰值。
3. 并行编码的资源竞态：x264 的 -threads 由 _x264_auto_thread_count(height) 自动推导，与推理进程共享 CPU；检查基于 os.sched_getaffinity 的当前可用核数，但同机其他租户的瞬时负载无法感知，可能造成编码与推理互相拖慢。
4. 共享 memfd 缓冲的并发访问：全局 _cached_cuda_video_buffer 由 _cuda_video_buffer_cache_lock 保护，但若 _acquire_cuda_video_buffer 在返回后即释放锁，两个并行线程可能拿到同一块缓冲并相互覆盖。建议确认锁覆盖整个 copy/sendfile 窗口，或在并行路径为每路分配独立缓冲（这一点从当前源码窗口无法完全确认）。
5. ffmpeg 管道早退：Popen 后若 ffmpeg 因参数或编码错误提前退出，继续向 stdin 写可能触发 BrokenPipeError；代码在 finally 中 kill 与 wait，并用 stderr 临时文件收集原因，但这类错误在真实编码失败时仍可能暴露在回退日志中。 - 影响：对用户：长视频生成的显存足迹显著下降（1344×768 五秒转换临时显存约 1.92 GB → 108 MB，十五秒 RGB24 主机缓冲约 1.13 GB → 21.7 MB），num_outputs_per_prompt > 1 的保存墙钟时间约减半（1.30 s → 0.69 s），且输出与旧路径字节级一致，属于透明优化。 对系统：并行编码会短暂双倍占用 x264 线程与 CPU 核，已用 affinity 上限约束；多路输出同时 ffprobe 也只存在于 MiniMax-H3 校验阶段。 对团队：直接保存路径从同步 subprocess.run 切换为 Popen + 管道，复杂度上升，后续需要保持回退路径的测试覆盖；共享 memfd 缓冲的线程安全是一个需要跟踪的隐患。

- 风险标记：平台依赖 memfd/sendfile，并行编码 CPU 竞态，OOM 保护为启发式，共享 memfd 缓冲并发访问待确认

关联脉络

- PR #34359 [Diffusion] Support native and PEFT MiniMax H3 LoRAs: 同一 MiniMax-H3 输出链路，多输出校验需求与本 PR 的并行 ffprobe 直接衔接。
- PR #34508 [Diffusion][LTX-2] Allocate AdaLN outputs from one contiguous slab: 同为 diffusion 推理内存足迹优化的性能 PR，体现对设备内存的系统性收敛。
- PR #34412 [Diffusion] Improve bit-exact fusion fallback diagnostics: 两者都在 bit-exact 语义上做验证与诊断，本 PR 保证新旧输出字节级一致，与 34412 的回退诊断互为补充。