

PR #56729 完整报告

vllm-project/vllm

[Security] Cap Qwen-VL video sampling knobs

合并时间: 2026-09-14 18:22

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56729>

执行摘要

- 一句话: Qwen-VL 视频采样加服务端上限, 防止请求放大解码量
- 推荐动作: 值得精读: 改动虽小 (+10/-4), 但它是 request-controlled 参数落到资源消耗路径上的一类典型安全收口, 理解 `media_io_kwargs` → `VideoMediaIO` → `compute_frames_index_to_sample` 这条链路对排查多模态输入侧问题很有价值。重点看两处设计决策: 一是「在采样器内部 clamp」与 Isotr0py 建议的「把 `fps/max_frames` 变成不可被请求覆盖的 static kwargs」之间的取舍 (后者被明确延后为 follow-up refactor); 二是评审要求删除新增测试后, Qwen 封顶逻辑处于无测试保护状态。追这条线的读者应关注同系列 glm5next/Molmo2/OpenPangu 封顶 PR 与后续 kwargs 机制重构的落地情况。

功能与动机

PR body 明确指出问题根因: 「Qwen2-VL and Qwen3-VL video samplers ignore `num_frames` and bound sampling on request-controlled `max_frames` and `fps`. A client can raise those knobs through `media_io_kwargs` and force the frontend to decode every source frame.」即采样上限完全由请求参数决定, 缺少服务端天花板, 属于可被客户端放大的资源消耗问题 (安全类)。修复方式是复用 GLMGA 采样器已有的类级上限写法 (`_MAX_FRAMES` / `_MAX_FPS`), 保证「请求可以调低采样数, 但不能越过 768 帧或 30 fps」。

实现拆解

1. 新增服务端上限常量 (入口层): 在 `vllm/multimodal/video.py` 的 `Qwen3VLVideoBackend` 与 `Qwen2VLVideoBackend` 两个类中各加入 `_MAX_FRAMES`: `ClassVar[int] = 768` 与 `_MAX_FPS`: `ClassVar[int] = 30`, 与既有 `TestGLMGASamplingCaps` 所覆盖的 GLMGA 采样器写法保持一致, 把「上限」从散落的字面量提升为类级、可被子类覆盖但不随请求变化的契约。
2. Qwen3-VL 采样封顶: 在 `Qwen3VLVideoBackend.compute_frames_index_to_sample` 中, 把 `fps = target.fps` 改为 `fps = min(target.fps, cls._MAX_FPS)`, 把 `max_frames = kwargs.get("max_frames", 768)` 改为 `min(kwargs.get("max_frames", cls._MAX_FRAMES), cls._MAX_FRAMES)`。由于后续 `num_frames = min(max(num_frames, min_frames), max_frames, total_frames_num)` 会再与 `max_frames` 夹紧, `min_frames` 即便被请求调大也仍被 `max_frames` 兜住, 不会绕过封顶。

3. Qwen2-VL 采样封顶：在 `Qwen2VLVideoBackend.compute_frames_index_to_sample` 中对 `max_frames` 做同样的 min 封顶（在向下取整到 `temporal_patch_size` 之前），并把帧率封顶内联进采样帧数计算 $n = \text{total_frames_num} / \text{original_fps} * \min(\text{target.fps}, \text{cls_MAX_FPS})$ ，随后仍走原有的 min/max 夹紧与 $\text{math.floor}(n / \text{temporal_patch_size}) * \text{temporal_patch_size}$ ，因此对 HF transformers 逐帧对齐的索引语义没有改变。
4. 覆盖边界说明（非代码改动）：PR body 说明收敛点是 `compute_frames_index_to_sample`，OpenCV、TorchCodec、PyNvVideoCodec、DeepStream 四条解码路径都经由该方法；`video/jpeg` 的 `load_base64` 走 `num_frames`、不经这些采样器；Rust 前端直接拒绝 `media_io_kwargs`，不受影响。
5. 测试与配套：作者原本在 `tests/multimodal/test_video.py` 增加了用例，评审（Isotr0py）要求删除，最终 diff 只包含源码文件（`additions=10 / deletions=4`，`changed_files_count=1`）。PR body 以「Existing TestGLMGASamplingCaps tests already cover the same clamping pattern」作为覆盖依据，但这是 GLMGA 后端而非 Qwen 后端的既有测试，本 PR 未新增 Qwen 专属断言。

关键文件：

- `vllm/multimodal/video.py`（模块 视频采样；类别 source；类型 core-logic；符号 `Qwen3VLVideoBackend`, `Qwen2VLVideoBackend`, `compute_frames_index_to_sample`, `_MAX_FRAMES`）：唯一变更文件；`Qwen3VLVideoBackend` 与 `Qwen2VLVideoBackend` 新增类级 `_MAX_FRAMES/_MAX_FPS` 并对请求侧 `max_frames`、`fps` 做 min 封顶，是所有视频解码后端（OpenCV/TorchCodec/PyNvVideoCodec/DeepStream）共用的采样收敛点。

关键符号：`Qwen3VLVideoBackend.compute_frames_index_to_sample`,
`Qwen2VLVideoBackend.compute_frames_index_to_sample`

关键源码片段

`vllm/multimodal/video.py`

唯一变更文件；`Qwen3VLVideoBackend` 与 `Qwen2VLVideoBackend` 新增类级 `_MAX_FRAMES/_MAX_FPS` 并对请求侧 `max_frames`、`fps` 做 min 封顶，是所有视频解码后端（OpenCV/TorchCodec/PyNvVideoCodec/DeepStream）共用的采样收敛点。

```
# vllm/multimodal/video.py (head 版本，两个 Qwen-VL 采样后端的关键实现)
#
# 修复主线：把「服务端上限」提升为类级常量，请求侧传入的 max_frames / fps
# 只能在其之下调整，无法越过 768 帧、30 fps 的硬顶。该写法对齐已有的 GLMGA 采样器。
```

```
@VIDEO_LOADER_REGISTRY.register(
    "qwen3_vl",
    video_processor=("Qwen3VLVideoProcessor", "Cosmos3EdgeVideoProcessor"),
)
class Qwen3VLVideoBackend(VideoBackend):
    # 服务端硬上限：类属性，不随请求参数变化
    _MAX_FRAMES: ClassVar[int] = 768
    _MAX_FPS: ClassVar[int] = 30

    @classmethod
```

```

def compute_frames_index_to_sample(
    cls,
    source: VideoSourceMetadata,
    target: VideoTargetMetadata,
    **kwargs,
) -> list[int]:
    total_frames_num = source.total_frames_num
    original_fps = source.original_fps
    # target.fps 来自请求 (media_io_kwargs 可覆盖), 这里先取 min 封顶:
    # 请求可以调低帧率, 但调不高过 _MAX_FPS
    fps = min(target.fps, cls._MAX_FPS)
    max_frame_idx = source.total_frames_num - 1
    min_frames = kwargs.get("min_frames", 4)
    # 修复前是裸的 kwargs.get("max_frames", 768), 请求可传任意大值;
    # 修复后先与 _MAX_FRAMES 取小, 保证采样帧数不突破服务端上限
    max_frames = min(kwargs.get("max_frames", cls._MAX_FRAMES), cls._MAX_FRAMES)

    # 与 HF transformers 的 Qwen3-VL 采样保持一致: 先按帧率推导采样帧数,
    # 再用 [min_frames, max_frames] 与视频总帧数夹紧;
    # 即便请求调大 min_frames, 也仍被 max_frames 兜住
    num_frames = int(total_frames_num / original_fps * fps)
    num_frames = min(max(num_frames, min_frames), max_frames, total_frames_num)
    indices = np.linspace(0, max_frame_idx, num_frames).round().astype(int).tolist()
    return indices

@VIDEO_LOADER_REGISTRY.register(
    "qwen2_vl",
    video_processor="Qwen2VLVideoProcessor",
)
class Qwen2VLVideoBackend(VideoBackend):
    """Qwen2-VL / Qwen2.5-VL fps 采样后端, 索引需与 HF transformers 逐帧对齐。"""

    _MAX_FRAMES: ClassVar[int] = 768
    _MAX_FPS: ClassVar[int] = 30

    @classmethod
    def compute_frames_index_to_sample(
        cls,
        source: VideoSourceMetadata,
        target: VideoTargetMetadata,
        **kwargs,
    ) -> list[int]:
        total_frames_num = source.total_frames_num
        original_fps = source.original_fps
        temporal_patch_size = kwargs.get("temporal_patch_size", 2)
        min_frames = kwargs.get("min_frames", 4)
        # 与 Qwen3-VL 相同的封顶逻辑, 作用在向下取整到 temporal_patch_size 之前
        max_frames = min(kwargs.get("max_frames", cls._MAX_FRAMES), cls._MAX_FRAMES)

```

```

# 帧率未知 (VFR / 流式 / 损坏容器) 时直接报错, 避免除零
if original_fps <= 0:
    raise ValueError(
        "Qwen2-VL video sampling needs a known source fps, but the "
        "container reported 0 (variable or unknown frame rate)."
    )

max_frames = (
    math.floor(min(max_frames, total_frames_num) / temporal_patch_size)
    * temporal_patch_size
)
# fps 的封顶发生在计算采样帧数时: 先 min 再进入 min/max 夹紧与取整
n = total_frames_num / original_fps * min(target_fps, cls._MAX_FPS)
n = min(max(n, min_frames), max_frames, total_frames_num)
n = math.floor(n / temporal_patch_size) * temporal_patch_size

# torch.arange 与 transformers 的 float32 索引计算保持一致;
# 尾部可能越界一帧, 统一 clamp 到最后一帧
indices = torch.arange(0, total_frames_num, total_frames_num / n).int()
return torch.clamp(indices, max=total_frames_num - 1).tolist()

```

评论区精华

两处关键评审意见都来自 Isotr0py, 且都改变了最终产物:

1. 测试被要求移除: 对 tests/multimodal/test_video.py 的评论为「Let's remove these added tests.」, 作者回复「This should have been fixed」, 最终提交中该测试文件不再出现。结果是本次安全修复没有专属回归测试。
2. 设计上更优解被延后: 对 vllm/multimodal/video.py 第 456 行 (Qwen2-VL 的采样帧数计算行) 评论「I think a better solution is making fps and max_frames static kwargs, which can't be overridden by arguments from request. But anyway, let's leave it to be done in followup refactoring.」——评审认可当前 clamp 可用, 但认为根本解法是让这些参数不可被请求覆盖, 属于被显式记下的后续重构债。
3. claude[bot] 的自动评审因来自 fork 被禁用, 最终由 Isotr0py 执行 /ci run (Buildkite #88769) 并 APPROVED、合入。
 - 移除本 PR 新增的 Qwen 采样上限测试 (testing): 测试被移除, Qwen 侧封顶逻辑改为依赖既有 TestGLMGASamplingCaps 的同类模式作为间接依据, 本次变更没有专属回归测试。
 - 用 static kwargs 取代运行时 clamp 的更优解 (design): 认可当前 min 封顶作为短期修复, 但根本方案是让 fps/max_frames 不可被请求覆盖; 明确记为后续重构项, 未在本 PR 落地。
 - fork PR 自动评审被禁用 (other): 由 Isotr0py 手动执行 /ci run (Buildkite #88769)、APPROVED 并合入, 未引入额外自动化审查。

风险与影响

- 风险:

1. 缺少测试覆盖（主要风险）：新增的 Qwen 封顶逻辑没有任何直接测试，原计划的 tests/multimodal/test_video.py 用例被移除；现有 TestGLMGASamplingCaps 只覆盖 GLMGA 后端。若后续「static kwargs」重构时误删这两个 min 调用，CI 不会报警。
 2. 静默行为变更：之前可通过 media_io_kwargs 合法抬高 max_frames (>768) 或 fps (>30) 的客户端，会在无任何日志 / 告警的情况下拿到被截断的采样结果，采样帧数变少可能影响长视频理解的输出质量，属于语义上的静默降级。
 3. 同类后端封顶不统一：PR body 明确列出 glm5next、Molmo2、OpenPangu 由 #55727、#56207、#56390 各自处理，本 PR 只收口 Qwen2/Qwen3-VL。在这批 PR 齐备前，media_io_kwargs 放大路径在其他采样后端仍然可用，安全姿态不一致。
 4. 相邻既有缺陷未处理：Qwen3VLVideoBackend.compute_frames_index_to_sample 仍缺少 Qwen2-VL 那样的 original_fps <= 0 保护（base/head 均如此，非本 PR 引入），total_frames_num / original_fps 在 VFR/ 损坏容器下仍有除零风险。
 5. 封顶依赖类常量：_MAX_FRAMES / _MAX_FPS 是 ClassVar，任何子类或后续改动都可覆盖；它们只约束采样索引数量，min_frames、temporal_patch_size 等相邻 kwargs 仍由请求注入（虽然被 max_frames 与总帧数夹紧）。- 影响：对用户 / 客户端：默认行为不变（默认 fps=2、max_frames 默认 768，本就在上限之内）；只有显式通过 media_io_kwargs 抬高参数的请求会感知到采样数被截断。对系统：收窄了单请求可触发的视频解码工作量上限，覆盖 chat completions、responses、embeddings、tokenize、invocations、batch 与离线 LLM 全部经 MediaConnector → VideoMediaIO → 采样器链 的入口，并同时覆盖四种解码后端，收益面广而改动面极小。对团队：确立「采样上限 = 类级常量」的收敛模式，并把「参数不可被请求覆盖」登记为后续重构方向；由于 Qwen 侧无测试，重构时需要人工补覆盖。
- 风险标记：缺少测试覆盖，请求参数被静默截断，同类采样后端封顶不统一，封顶依赖类常量可被覆盖

关联脉络

- PR #54323 [Bugfix][Multimodal] Validate base64 video payloads, matching image and audio: 同属多模态视频输入侧的健壮性 / 安全加固，改动的 vllm/multimodal/media/video.py 与本 PR 的 vllm/multimodal/video.py 处于同一条 MediaConnector → VideoMediaIO → 视频后端 链路上，可对照理解「请求侧载荷校验 / 采样侧上限」两类防线。
- PR #56366 [Bugfix][Rust Frontend][Multimodal] Align DeepSeek V4.1 and Kimi K3 media with rendered placeholders: 同属前端多模态媒体处理主线；该 PR 说明 Rust 前端与 Python 前端在多模态媒体处理上的差异，而本 PR 明确以「Rust frontend rejects media_io_kwargs and is unaffected」划分影响边界，两者对照可确认封顶的覆盖面。