PR #26947 完整报告

sgl-project/sglang

[diffusion] Speed up PNG image output saving

合并时间：2026-06-02 00:43

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26947

执行摘要

- 一句话：PNG 输出改用 Pillow 并降低压缩级别加速
- 推荐动作：值得快速合并。变更是局部化、可回退、测试覆盖完善（新增 68 行测试）。设计决策清晰：对 PNG 专用路径优化，不改动其他格式；压缩级别暴露给用户并可配置。

功能与动机

对于 PNG 图像响应，服务器当前在 imageio PNG 编码上消耗了大量请求尾延迟。PR body 中给出测量数据：直接 720p RGB PNG 编码探测显示 imageio/Pillow 默认压缩约 252ms，Pillow compress_level=1 约 59ms，compress_level=0 约 31ms。目标是将 PNG 编码延迟降低数倍以改善尾延迟。

实现拆解

1. 添加 Pillow 导入：在 python/sglang/multimodal_gen/runtime/entrypoints/utils.py 中增加 from PIL import Image 导入。
2. 抽取 _save_image_frame 函数：将原 post_process_sample 中的图像保存逻辑（单帧和多帧分支）统一为 _save_image_frame 函数，该函数根据文件扩展名决策：
   - 对于 .png 文件，使用 Pillow 的 Image.fromarray().save() 并设置 compress_level 参数（默认 1，若用户指定 output_compression 且非 75，则映射到 0-9 范围）。
   - 对于其他格式，回退到原有的 imageio.imwrite 路径。
3. 调整调用点：将 post_process_sample 中原来的 imageio.imwrite 调用替换为对 _save_image_frame 的调用，传入 quality 和 output_compression 参数。
4. 新增单元测试文件：python/sglang/multimodal_gen/test/unit/test_output_saving.py 包含两个参数化测试：
   - test_png_output_saving_preserves_pixels：验证不同 output_compression 设置下保存的 PNG 文件像素完全一致。
   - test_png_output_saving_uses_fast_pillow_path：通过 monkeypatch 使 imageio.imwrite 抛出异常、监视 Pillow.save 调用，确保 PNG 输出确实走 Pillow 路径且 compress_level 正确。

关键文件：

- python/sglang/multimodal_gen/runtime/entrypoints/utils.py（模块 图像输出；类别 source；类型 core-logic；符号 _save_image_frame）：核心变更文件：新增 _save_image_frame 函数、Pillow 导入、修改 post_process_sample 中图像保存调用。
- python/sglang/multimodal_gen/test/unit/test_output_saving.py（模块 测试；类别 test；类型 test-coverage；符号 _rgb_frame, test_png_output_saving_preserves_pixels, test_png_output_saving_uses_fast_pillow_path, fail_imageio_imwrite）：新增的单元测试文件，覆盖像素保持性和 Pillow 路径验证，确保变更正确性。

关键符号：_save_image_frame, test_png_output_saving_preserves_pixels, test_png_output_saving_uses_fast_pillow_path

关键源码片段

python/sglang/multimodal_gen/runtime/entrypoints/utils.py

核心变更文件：新增 _save_image_frame 函数、Pillow 导入、修改 post_process_sample 中图像保存调用。

```
def _save_image_frame(
    path: str, frame: np.ndarray, quality: int | None, output_compression: int | None
) -> None:
    # 根据文件扩展名决定编码后端
    ext = os.path.splitext(path)[1].lower()
    if ext == ".png":
        # 默认压缩级别 1（平衡速度与体积），若用户指定且非默认值 75
        # 则按比例映射到 0-9
        compress_level = 1
        if output_compression is not None and output_compression != 75:
            compress_level = max(0, min(9, round(output_compression / 100 * 9)))
        # 去掉单通道冗余维度
        if frame.ndim == 3 and frame.shape[-1] == 1:
            frame = frame[..., 0]
        # 使用 Pillow 保存，速度远快于 imageio
        Image.fromarray(frame).save(path, format="PNG", compress_level=compress_level)
    else:
        # 非 PNG 格式（如 JPEG）沿用原 imageio 路径
        imageio.imwrite(path, frame, quality=quality)
```

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

新增的单元测试文件，覆盖像素保持性和 Pillow 路径验证，确保变更正确性。

```
# 创建一个固定的小 RGB 图像用于测试
# 包含多种颜色值以确保编码 / 解码后像素不变
_rgb_frame = lambda: np.array([
    [[0, 32, 255], [64, 128, 192], [255, 224, 16]],
    [[9, 17, 33], [127, 128, 129], [240, 12, 88]],
], dtype=np.uint8)


# 测试 1: 像素保持 — 不同 compression 下保存的 PNG 解码后应与原始帧相同
```

```
@pytest.mark.parametrize("output_compression", [None, 0, 75])
def test_png_output_saving_preserves_pixels(tmp_path, output_compression):
    frame = _rgb_frame()
    output_path = tmp_path / f"sample_{output_compression}.png"
    frames = post_process_sample(frame, DataType.IMAGE, fps=1,
                                 save_file_path=str(output_path),
                                 output_compression=output_compression)
    assert output_path.exists()
    np.testing.assert_array_equal(frames[0], frame)
    np.testing.assert_array_equal(np.array(Image.open(output_path)), frame)

# 测试 2: 验证 PNG 输出确实使用 Pillow 路径，且 compress_level 正确
# 通过 monkeypatch 使 imageio.imwrite 失败、监视 Pillow.save 调用
@pytest.mark.parametrize(("output_compression", "expected_compress_level"),
                         [(None, 1), (0, 0), (75, 1)])
def test_png_output_saving_uses_fast_pillow_path(
        tmp_path, monkeypatch, output_compression, expected_compress_level):
    frame = _rgb_frame()
    output_path = tmp_path / f"sample_{output_compression}.png"
    # 确保 imageio 不被调用
    monkeypatch.setattr(output_utils.imageio, "imwrite",
                        lambda *a,**kw: (_ for _ in ()).throw(AssertionError("should use Pillow")))
    save_calls = []
    original_save = Image.Image.save
    def save_spy(self, fp, format=None, **params):
        save_calls.append((format, params.get("compress_level")))
        return original_save(self, fp, format=format, **params)
    monkeypatch.setattr(Image.Image, "save", save_spy)
    post_process_sample(frame, DataType.IMAGE, fps=1,
                        save_file_path=str(output_path),
                        output_compression=output_compression)
    assert save_calls == [("PNG", expected_compress_level)]
```

评论区精华

PR 无 review 评论，讨论主要体现为 PR body 中的性能测量数据和设计决策说明。author 明确了 PNG 保持无损，改变压缩级别不影响像素值；并对比了不同压缩级别的编码耗时。

- 暂无高价值评论线程

风险与影响

- 风险：低风险。变更仅影响 PNG 输出编码路径，其他格式（JPEG 等）仍使用 imageio。PNG 压缩级别映射逻辑对极端输入（如 output_compression 为负数或极大值）有 max(0, min(9, ...)) 钳位保护。但注意：默认 compress_level 从 imageio 的默认（约 6 或更高）改为了 1，会导致存储文件变大，但不影响像素值。如果下游依赖特定文件大小的自动化系统（如期望固定字节数），可能受影响。

- 影响：用户：PNG 输出响应速度提升数倍（尾延迟降低 70%+），体验显著改善；文件大小可能增大。 系统：降低 PNG 编码阶段的 GPU/CPU 等待时间，提升服务吞吐。 团队：引入 Pillow 作为新依赖（但通常已预装），后续维护需注意 PNG 相关兼容性。
- 风险标记：文件大小变大，依赖引入（Pillow）

关联脉络

- PR #26926 [diffusion] feat: improve cosmos3 serve API support: 同为 diffusion 模块，对 Cosmos3 服务的支持改进，与本 PR 的 PNG 输出优化共享测试环境和性能关注