PR #43843 完整报告

vllm-project/vllm

[Misc] Support local image encoding in benchmarks

合并时间：2026-06-02 23:15

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43843

执行摘要

- 一句话：在基准测试中支持客户端侧图像编码
- 推荐动作：值得精读，特别是如何复用 vllm.multimodal.utils 中的工具实现客户端编码，以及如何在不破坏现有行为的前提下逐步添加特性。设计决策清晰，适合作为多模态基准测试扩展的参考。

功能与动机

当基准测试客户端和服务端运行在不同机器或容器中时，默认的引用方式（发送本地路径或 URL）会因服务端无法访问而失败。此 PR 通过客户端侧编码为 base64 data URL 来解决。

实现拆解

1. 修改 vllm/benchmarks/datasets/datasets.py 中的 process_image 函数，新增 ensure_client_side_data 关键字参数，利用 vllm.multimodal.utils.fetch_image 和 encode_image_url 将本地路径、file:// 和 HTTP(S) URL 图像编码为 base64 PNG data URL，同时保持已有 data:image/ URL 不变。
2. 在 _process_content_part、_process_interleaved_content 和新增的 _process_image_files 函数中透传该参数，确保交错内容场景同样支持。
3. 在 get_samples 入口处通过 getattr(args, 'custom_ensure_client_side_data', False) 获取命令行参数并传递给 CustomImageDataset.sample。
4. 新增 CLI 参数 --custom-ensure-client-side-data，类型为 store_true。
5. 在测试文件 tests/benchmarks/test_custom_image_dataset.py 中新增辅助函数和三个用例，覆盖本地文件、远程 URL 和 data URL 的编码行为。
6. 更新文档 docs/benchmarking/cli.md，说明新标志的用法和注意事项。

关键文件：

- vllm/benchmarks/datasets/datasets.py（模块 基准测试；类别 source；类型 core-logic；符号 process_image, _process_content_part, _process_interleaved_content, _process_image_files）：核心逻辑变更：修改 process_image 函数支持客户端编码，新增 _process_image_files 等辅助函数，添加 CLI 参数并透传至采样流程。
- tests/benchmarks/test_custom_image_dataset.py（模块 测试套件；类别 test；类型 test-coverage；符号 _write_png, _decode_data_url, _assert_png_data_url,

test_custom_image_dataset_encodes_image_media_when_requested）：新增三个测试用例和辅助函数，验证客户端编码对本地文件、远程 URL、data URL 以及交错内容场景的正确性。

- docs/benchmarking/cli.md（模块 文档；类别 docs；类型 documentation）：更新文档，说明 --custom-ensure-client-side-data 标志的用法。

关键符号：process_image, _process_content_part, _process_interleaved_content, _process_image_files, test_custom_image_dataset_encodes_image_media_when_requested, test_custom_image_dataset_encodes_interleaved_image_media, test_custom_image_dataset_rejects_invalid_image_media

关键源码片段

vllm/benchmarks/datasets/datasets.py

核心逻辑变更：修改 process_image 函数支持客户端编码，新增 _process_image_files 等辅助函数，添加 CLI 参数并透传至采样流程。

```
# vllm/benchmarks/datasets/datasets.py (head 简化版 )

def process_image(
    image: Any,
    *,
    ensure_client_side_data: bool = False,
) -> Mapping[str, Any]:
    """
    ...
    - If ensure_client_side_data is True, local and HTTP(S) image references
      are loaded and encoded as base64 image data URLs. Existing data:image
      URLs are kept unchanged.
    """
    if isinstance(image, dict) and "bytes" in image:
        image = Image.open(BytesIO(image["bytes"]))
    if isinstance(image, Image.Image):
        image = convert_image_mode(image, "RGB")
        with io.BytesIO() as image_data:
            image.save(image_data, format="JPEG")
            image_base64 = base64.b64encode(image_data.getvalue()).decode("utf-8")
        return {
            "type": "image_url",
            "image_url": {"url": f"{image_base64}"},
        }
    if isinstance(image, str):
        image_url = (
            image
            if image.startswith(("http://", "https://", "file://", "data:image/"))
            else f"file://{image}"
        )
        # 当需要客户端侧数据且不是已有 data URL 时，通过 fetch_image 加载并编码为 base64 PNG
```

```
        if ensure_client_side_data and not image_url.startswith("data:image/"):
            try:
                fetched_image = fetch_image(image_url)
                image_url = encode_image_url(fetched_image)
            except Exception as e:
                raise ValueError(f"Invalid image URL: {image_url}") from e
        return {"type": "image_url", "image_url": {"url": image_url}}
    raise ValueError(
        f"Invalid image input {image}. Must be a PIL.Image.Image, "
        "str (URL, file path, or base64 data URL), or dictionary with raw image bytes."
    )
```

tests/benchmarks/test_custom_image_dataset.py

新增三个测试用例和辅助函数，验证客户端编码对本地文件、远程 URL、data URL 以及交错内容场景的正确性。

```
# tests/benchmarks/test_custom_image_dataset.py（头版新增部分）

@pytest.mark.benchmark
def test_custom_image_dataset_encodes_image_media_when_requested(
    tmp_path: Path,
    monkeypatch: pytest.MonkeyPatch,
) -> None:
    image_a = tmp_path / "chart_a.png"
    image_b = tmp_path / "chart b.png"
    _write_png(image_a, color=(255, 0, 0))
    _write_png(image_b, color=(0, 255, 0))
    data_url = ""
    remote_url = "https://example.com/chart.png"
    original_fetch_image = datasets_module.fetch_image

    # 模拟远程图片获取，返回蓝色 1x1 像素
    def fake_fetch_image(image_url: str) -> Image.Image:
        if image_url == remote_url:
            return Image.new("RGB", (1, 1), color=(0, 0, 255))
        return original_fetch_image(image_url)

    monkeypatch.setattr(datasets_module, "fetch_image", fake_fetch_image)

    jsonl = tmp_path / "images.jsonl"
    _write_jsonl(
        jsonl,
        [
            {
                "prompt": "Compare the charts.",
                "image_files": [
                    str(image_a),
                    image_b.as_uri(),
                    remote_url,
```

```
                data_url,
            ],
        }
    ],
)

dataset = CustomImageDataset(dataset_path=str(jsonl), disable_shuffle=True)
samples = dataset.sample(
    tokenizer=_Tokenizer(),
    num_requests=1,
    output_len=32,
    ensure_client_side_data=True,
)

assert len(samples) == 1
assert isinstance(samples[0].multi_modal_data, list)
image_urls = [part["image_url"]["url"] for part in samples[0].multi_modal_data]

# 前三个应为 PNG data URL
_assert_png_data_url(image_urls[0])
_assert_png_data_url(image_urls[1])
_assert_png_data_url(image_urls[2])
# data URL 保持不变
assert image_urls[3] == data_url
```

评论区精华

- **复用 vllm.multimodal.utils：Reviewer 建议使用已有的 fetch_image 和 encode_image_url 替代手动实现。贡献者采纳并移除了自定义 URL 解析函数。
- **客户端编码范围的争议：最初提议对所有非 data URL 进行编码，但贡献者担心对已为 data URL 的输入会引入额外解码—重编码开销。最终达成一致：保留 data:image/ 原样，仅对本地路径、file:// 和 HTTP(S) URL 进行客户端编码。
- **标志命名：初始名称 --custom-ensure-client-side-data 由 reviewer 建议，贡献者确认修改。
  - 复用 vllm.multimodal.utils 替代手动处理 (design): 使用 vllm.multimodal.utils 中的工具实现客户端编码。
  - 客户端编码范围：data URL 的处理 (design): 保留 data:image/ URL 不变，仅对本地路径、file:// 和 HTTP(S) URL 进行编码。
  - 标志命名 (style): 标志名确定为 --custom-ensure-client-side-data。

风险与影响

- 风险：风险较低，因为新标志默认关闭，不影响现有用户。但需注意：若 fetch_image 调用失败（例如文件不存在、网络不可达），process_image 会抛出 ValueError，可能中断基准测试运行。依赖 vllm.multimodal.utils 已在项目中使用，未引入新依赖。

- 影响：影响范围局限于使用 custom_image 数据集且需要客户端处理图像的用户。默认行为不变；启用新标志后，请求体中的图像 URL 会被替换为 base64 data URL，导致请求体显著增大（但这是预期行为）。服务器无需额外适配。文档和测试更新确保可维护性。
- 风险标记：客户端编码失败可能中断基准测试，默认不启用

关联脉络

- 暂无明显关联 PR