PR #32489 完整报告

sgl-project/sglang

Add local ZIP uploader for whl releases

合并时间：2026-07-27 15:04

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32489

执行摘要

- 一句话：新增本地 ZIP 上传器，自动化 whl Release 发布与索引更新
- 推荐动作：值得精读 shell 脚本中的幂等设计和 Python 索引更新逻辑，可作为自动化发布工具的实现参考。但注意测试覆盖被移除，建议在生产前补充 CI 中的集成测试。

功能与动机

当前 sglang 发布时，除标准 whl 包外还需要附带其他大型文件（如模型缓存、工具包等），缺乏一个统一的、可恢复的上传渠道。该 PR 提供了一个安全的本地 ZIP 上传器，能够将文件作为 Release 资源发布并自动更新 HTML 索引，支持通过文件名、大小、SHA256 和 GitHub 摘要进行恢复验证。

实现拆解

1. Shell 上传脚本（upload_zip_to_whl.sh）：接收 ZIP 路径和版本信息，计算 SHA256，检查 GitHub Release 是否已存在完全一致的资源（名称、大小、摘要），避免重复上传；若不存在则创建 Release 并上传资源，最后调用 Python 索引更新脚本。
2. Python 索引更新脚本（update_others_whl_index.py）：在 gh-pages 分支的根 index.html 中添加 others/ 链接（幂等去重），在 others/index.html 中追加新资源条目（包含 SHA256 锚点），通过检查 identity 字符串避免重复条目。
3. 文档更新（README.md）：详细描述脚本的用法、前置条件、示例和恢复机制。
4. 测试（已被移除）：第一个提交包含单元测试，但第二个提交将其删除，因此最终版本无测试文件。

关键文件：

- scripts/release/update_others_whl_index.py（模块 发布脚本；类别 source；类型 core-logic；符号 parse_args, update_root_index, update_others_index, main）：核心 Python 脚本，实现索引的幂等更新逻辑。
- scripts/release/upload_zip_to_whl.sh（模块 发布脚本；类别 infra；类型 core-logic；符号 usage, die, warn, cleanup）：主上传脚本，包含完整的发布创建、资源上传和恢复验证逻辑。
- scripts/release/README.md（模块 文档；类别 docs；类型 documentation）：新增使用文档，确保其他开发者能正确使用上传工具。

关键符号：parse_args, update_root_index, update_others_index, main, usage, die, warn, cleanup, api_get_optional, compute_sha256, load_and_validate_release, create_release_and_asset, push_index_updates

关键源码片段

scripts/release/update_others_whl_index.py

核心 Python 脚本，实现索引的幂等更新逻辑。

```
# 更新根 index.html，确保包含 others/ 链接
# 如果已存在且仅有一条则跳过；若多条则去重后再添加
ROOT_LINK = '<a href="others/">others</a><br>'

def update_root_index(repo_dir: pathlib.Path) -> bool:
    index_path = repo_dir / "index.html"
    content = index_path.read_text(encoding="utf-8")
    lines = content.splitlines(keepends=True)
    # 统计 ROOT_LINK 出现次数
    root_link_count = sum(line.rstrip("

") == ROOT_LINK for line in lines)
    if root_link_count == 1:
        return False # 已存在且唯一，无变更
    if root_link_count > 1:
        # 去重：移除所有 ROOT_LINK 行
        content = "".join(line for line in lines if line.rstrip("

") != ROOT_LINK)
    if content and not content.endswith("\n"):
        content += "\n"
    index_path.write_text(f"{content}{ROOT_LINK}\n", encoding="utf-8")
    return True

# 更新 others/index.html，追加新资源条目（含 SHA256 锚点）
def update_others_index(
    repo_dir: pathlib.Path,
    asset_url: str,
    filename: str,
    tag: str,
    sha256: str,
) -> bool:
    index_dir = repo_dir / "others"
    index_path = index_dir / "index.html"
    # 对用户提供的数据做 HTML 转义，防止注入
    escaped_url = html.escape(asset_url, quote=True)
    escaped_filename = html.escape(filename, quote=True)
    escaped_tag = html.escape(tag, quote=True)
    # identity 用于检测是否已存在相同资源
    identity = f'href="{escaped_url}#sha256='
```

```
    entry = (
        f'<a href="{escaped_url}#sha256={sha256.lower()}">'
        f"{escaped_filename}</a> ({escaped_tag})<br>\n"
    )
    if index_path.exists():
        content = index_path.read_text(encoding="utf-8")
        if not content.startswith(OTHERS_HEADER):
            raise ValueError(f"{index_path} 头部不符合预期")
    else:
        content = OTHERS_HEADER # 初始 HTML 头部
    if identity in content:
        return False # 幂等：已存在则跳过
    index_dir.mkdir(parents=True, exist_ok=True)
    # 将新条目插入到头部之后，保留原有内容
    updated = f"{OTHERS_HEADER}{entry}{content[len(OTHERS_HEADER):]}"
    index_path.write_text(updated, encoding="utf-8")
    return True
```

scripts/release/upload_zip_to_whl.sh

主上传脚本，包含完整的发布创建、资源上传和恢复验证逻辑。

```
  # 验证现有 Release 是否与本地文件完全匹配（名称、大小、SHA256）
  load_and_validate_release() {
      local endpoint="$1"
      local expected_name="$2"
      local expected_size="$3"
      local expected_checksum="$4"
      local release_body
      local asset_rows
      local remote_name remote_size remote_url remote_digest
      local matched_size="" matched_url="" matched_digest=""
      local match_count=0
      local checksum_marker

      # 读取 Release URL
      RELEASE_URL=$(gh api "$endpoint" --jq '.html_url') || \
          die "无法读取 Release URL"
      release_body=$(gh api "$endpoint" --jq '.body // ""') || \
          die "无法读取 Release body"
      asset_rows=$( \
          gh api "$endpoint" \
              --jq '.assets[] | [.name, .size, .browser_download_url, (.digest // "")] | @tsv' \
      ) || die "无法读取 Release assets"

      # 遍历所有资源，查找名称匹配的项
      while IFS=$'\t' read -r remote_name remote_size remote_url remote_digest; do
          if [[ "$remote_name" == "$expected_name" ]]; then
              ((match_count += 1))
              matched_size="$remote_size"
```

```
            matched_url="$remote_url"
            matched_digest="$remote_digest"
        fi
    done <<<"$asset_rows"

    # 检查 SHA256 标记一致性
    checksum_marker="SHA256: \`${expected_checksum}\`"
    [[ "$release_body" == *"$checksum_marker"* ]] || \
        die "现有 Release 的 checksum 与本地 ZIP 不匹配"
    [[ "$match_count" -eq 1 ]] || \
        die "Release 必须恰好包含一个名为 ${expected_name} 的资源"
    [[ "$matched_size" == "$expected_size" ]] || \
        die "现有 Release 资源大小与本地 ZIP 不匹配"
    [[ "$matched_digest" == "sha256:${expected_checksum}" ]] || \
        die "现有 Release 资源摘要与本地 ZIP 不匹配"
    # 全部通过则返回 0，表示可用
}
```

评论区精华

无有效 review 讨论（仅有 Gemini Code Assist 的 sunset 公告）。PR 由作者自行合并。

- 暂无高价值评论线程

风险与影响

- 风险：Shell 脚本大量依赖 gh CLI 和外部网络（GitHub API），在认证不足或网络异常时可能失败；load_and_validate_release 和 create_release_and_asset 中的错误处理可能不全面；索引更新采用覆盖式写入，并发推送可能导致冲突；测试文件在最终提交中被移除，缺乏自动化验证。另外，ZIP 文件必须小于 2GiB，否则上传会失败。
- 影响：仅影响 sglang 发布流程的维护者，对系统运行时无影响。上传工具可以减少手动操作，提高发布可靠性。但引入新依赖（gh CLI、Python 3、Git），需要发布者在环境中准备。
- 风险标记：缺少测试覆盖，网络依赖，幂等性保证

关联脉络

- 暂无明显关联 PR