PR #33378 完整报告

sgl-project/sglang

[Feature] Add GLM Image usage report

合并时间：2026-08-05 18:59

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33378

执行摘要

- 一句话：GLM-Image 响应新增 token usage 报告
- 推荐动作：该 PR 值得精读，尤其适合关注以下设计决策：usage 在 AR/DiT 分离部署下的提取、聚合与按输出拆分机制；内部字段（cached_tokens）与公开字段（prompt_tokens_details）的边界处理；--enable-cache-report 与 LLM 侧同名参数的语义对齐方式。阅读时建议重点核对 glm_image.py 的返回类型变化和 gpu_worker.py 的合并语义，并留意合入前文档要求未落实、GPU CI 未验证两个遗留问题。

功能与动机

PR body 指出：在 AR/DiT 分离部署中，token usage 由 AR 阶段产生，但最终响应由 DiT 服务返回，此前 /v1/images/generations 只返回图像数据、延迟和内存信息，AR 阶段的 token 用量完全丢失。作者在评论中补充了四点价值：帮助用户检查 prefix cache 是否生效及缓存命中量；cached tokens 可能遵循不同的内部计费规则，暴露出来便于记账统计；使 GLM-Image 的 usage 报告与 LLM chat/completion API 保持一致；帮助调试 AR/DiT 分离部署中 token 由谁产生、由谁返回的问题。

实现拆解

实现按以下 5 步拆解：

1. 数据契约扩展：在 python/sglang/multimodal_gen/runtime/pipelines_core/schedule_batch.py 中给 Req 和 OutputBatch 新增 usage: dict[string, Any] | None 字段，作为 usage 跨阶段传递的统一载体；在 python/sglang/multimodal_gen/runtime/entrypoints/openai/protocol.py 新增 ImagePromptTokensDetails（cached_tokens）和 ImageUsage（prompt/completion/total/reasoning tokens、prompt_tokens_details、image_count）两个 pydantic 模型，ImageResponse 挂载可选 usage 字段。
2. AR 阶段提取 usage：在 python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/glm_image.py 新增 _extract_srt_usage(meta_info)，从外部 SRT AR 服务响应的 meta_info 中提取 prompt/completion/reasoning/cached tokens 并计算 total_tokens；generate_prior_tokens 与 generate_prior_tokens_batch 的返回值由二元组扩展为三元组（增加 usage 或 usages 列表），本地模式（不走外部 SRT）下 usage 恒为 None。

3. 聚合与透传：新增 _merge_srt_usage、_merge_srt_usages 对多输出（num_outputs_per_prompt > 1）的 usage 做整数累加；run_grouped_requests 将聚合结果写入 batch.usage，并通过 batch.extra["usage_by_output"] 按输出拆分，_make_sequential_request 为单个 Req 回填对应 usage；decoding.py 的 forward 把 batch.usage 透传到 OutputBatch；gpu_worker.py 的 _req_to_output_batch 保留 usage，_merge_expanded_singletons 在展开批合并时对 int 类型 usage 字段累加。
4. 响应组装与开关：python/sglang/multimodal_gen/runtime/entrypoints/openai/utils.py 的 add_common_data_to_response 中，先从内部 usage 中 pop 出 cached_tokens（默认隐藏），仅当 --enable-cache-report 开启且缓存命中数大于 0 时，才写入 prompt_tokens_details.cached_tokens；python/sglang/multimodal_gen/runtime/entrypoints/openai/image_api.py 的 _build_image_response_kwargs 为 usage 追加 image_count = 实际生成图片数。
5. 配置与测试配套：python/sglang/multimodal_gen/runtime/server_args/server_args.py 新增 enable_cache_report: bool = False 默认值与同名 CLI 参数；test/unit/test_glm_image_ar.py 新增 4 个用例，覆盖 meta_info 提取、多输出 usage 聚合、image_count 字段、cache-report 开关行为。文档未更新（review 中提出但合入前未见 docs 改动）。

关键文件：

- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/glm_image.py（模块 图像管线；类别 source；类型 data-contract；符号 _extract_srt_usage, _merge_srt_usage, _merge_srt_usages, generate_prior_tokens）：核心实现文件：新增 _extract_srt_usage / _merge_srt_usage / _merge_srt_usages，扩展 generate_prior_tokens 与 generate_prior_tokens_batch 返回值，在 run_grouped_requests 和 forward 中完成 usage 聚合与按输出拆分，是 AR/DiT 分离部署下 usage 传递的源头。
- python/sglang/multimodal_gen/test/unit/test_glm_image_ar.py（模块 测试配套；类别 test；类型 test-coverage；符号 test_srt_ar_extracts_usage_from_meta_info, test_srt_ar_forward_aggregates_usage, test_image_response_adds_image_count_to_usage, test_image_response_reports_cached_tokens_when_cache_report_enabled）：测试配套：新增 4 个用例覆盖 meta_info 提取、多输出 usage 聚合、image_count 字段、cache-report 开关行为，并调整既有用例适配新的三元组返回值。
- python/sglang/multimodal_gen/runtime/entrypoints/openai/protocol.py（模块 协议模型；类别 source；类型 core-logic；符号 ImagePromptTokensDetails, ImageUsage, ImageResponse）：协议模型层：新增 ImagePromptTokensDetails 与 ImageUsage, ImageResponse 挂载可选 usage，定义了对外的 OpenAI 兼容契约。
- python/sglang/multimodal_gen/runtime/entrypoints/openai/utils.py（模块 响应组装；类别 source；类型 core-logic；符号 add_common_data_to_response）：响应组装层：在 add_common_data_to_response 中剥离内部 cached_tokens，按 enable_cache_report 条件转为 prompt_tokens_details.cached_tokens，是对外可见性与内部信息保护的边界。
- python/sglang/multimodal_gen/runtime/entrypoints/openai/image_api.py（模块 接口入口；类别 source；类型 entrypoint；符号 _build_image_response_kwargs）：HTTP 入口

层：在 _build_image_response_kwargs 中为 usage 追加 image_count 字段，实现响应中图片数量上报。

- python/sglang/multimodal_gen/runtime/server_args/server_args.py（模块 服务配置；类别 source；类型 configuration；符号 ServerArgs, add_cli_args）：配置层：新增 enable_cache_report 默认值及 --enable-cache-report CLI 参数，用于控制 cached_tokens 是否对外暴露。
- python/sglang/multimodal_gen/runtime/pipelines_core/schedule_batch.py（模块 批处理；类别 source；类型 core-logic；符号 Req, OutputBatch）：数据结构层：Req 与 OutputBatch 新增 usage 字段，作为 usage 跨阶段透传的统一载体。
- python/sglang/multimodal_gen/runtime/managers/gpu_worker.py（模块 工作节点；类别 source；类型 core-logic；符号 _req_to_output_batch, _merge_expanded_singletons）：工作节点层：_req_to_output_batch 保留 usage，_merge_expanded_singletons 在多输出合并时对 int 型 usage 字段累加，保证展开 / 合并路径不丢数据。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/decoding.py（模块 解码阶段；类别 source；类型 core-logic；符号 forward）：解码阶段：forward 将 batch.usage 透传到 OutputBatch，完成 usage 从 Req 到最终响应的关键一跳。

关键符号：_extract_srt_usage, _merge_srt_usage, _merge_srt_usages, generate_prior_tokens, generate_prior_tokens_batch, run_grouped_requests, GlmImageAR.forward, add_common_data_to_response, _build_image_response_kwargs, _req_to_output_batch, _merge_expanded_singletons

关键源码片段

python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/glm_image.py

核心实现文件：新增 _extract_srt_usage / _merge_srt_usage / _merge_srt_usages，扩展 generate_prior_tokens 与 generate_prior_tokens_batch 返回值，在 run_grouped_requests 和 forward 中完成 usage 聚合与按输出拆分，是 AR/DiT 分离部署下 usage 传递的源头。

```
# 从外部 SRT AR 服务的响应 meta_info 中提取 token 用量。
# 外部 SRT 是独立的 LLM 服务，AR 阶段的 token 在这里产生；
# 本地模式（直接调用 vision_language_encoder.generate）没有 meta_info，返回 None。
def _extract_srt_usage(meta_info: dict[str, Any] | None) -> dict[str, int] | None:
    if not isinstance(meta_info, dict):
        return None

    usage = {
        "prompt_tokens": int(meta_info.get("prompt_tokens", 0) or 0),
        "completion_tokens": int(meta_info.get("completion_tokens", 0) or 0),
        "reasoning_tokens": int(meta_info.get("reasoning_tokens", 0) or 0),
        "cached_tokens": int(meta_info.get("cached_tokens", 0) or 0),
    }
    # total_tokens 由计算得出，不直接依赖上游字段，保证一致性
```

```
    usage["total_tokens"] = usage["prompt_tokens"] + usage["completion_tokens"]
    return usage


# 把单次 AR 输出的 usage 合并进累计值。cached_tokens / reasoning_tokens
# 都按整数累加，最终由响应层决定是否对外暴露。
def _merge_srt_usage(
    total_usage: dict[str, Any] | None, usage: dict[str, int] | None
) -> dict[str, Any] | None:
    if usage is None:
        return total_usage
    if total_usage is None:
        total_usage = {}
    for key, value in usage.items():
        total_usage[key] = int(total_usage.get(key, 0)) + int(value)
    return total_usage


# 多个输出的 usage 依次合并，供 num_outputs_per_prompt > 1 的场景使用
# （例如一次请求生成多张图时，把每张图对应 AR 调用的 token 数加总）。
def _merge_srt_usages(
    usages: list[dict[str, int] | None],
) -> dict[str, Any] | None:
    total_usage = None
    for usage in usages:
        total_usage = _merge_srt_usage(total_usage, usage)
    return total_usage
```

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

测试配套：新增 4 个用例覆盖 meta_info 提取、多输出 usage 聚合、image_count 字段、cache-report 开关行为，并调整既有用例适配新的三元组返回值。

```
# 验证从外部 SRT 响应的 meta_info 中正确提取 usage：
# prompt/completion/reasoning/cached 全部按 int 读取，total 由计算得出。
def test_srt_ar_extracts_usage_from_meta_info(self, mock_post, _mock_device):
    set_global_server_args(self._server_args())
    mock_post.return_value = _FakeResponse(
        list(range(1025)),
        meta_info={"prompt_tokens": 13, "completion_tokens": 25,
                   "reasoning_tokens": 0, "cached_tokens": 5},
    )
    stage = GlmImageAR(processor=_FakeProcessor(), vision_language_encoder=None)

    _, _, usage = stage.generate_prior_tokens(
        prompt="A simple product sketch",
        height=1024,
        width=1024,
        server_args=self._server_args(),
    )
```

```
        self.assertEqual(
            usage,
            {"prompt_tokens": 13, "completion_tokens": 25,
             "reasoning_tokens": 0, "cached_tokens": 5, "total_tokens": 38},
        )


    # 验证多输出聚合：同一请求生成多张图时，usage 按整数累加
    # （两个输出各 13 prompt + 25 completion，聚合后为 26 + 50）。
    def test_srt_ar_forward_aggregates_usage(self, mock_post, _mock_device):
        set_global_server_args(self._server_args())
        mock_post.side_effect = [
            _FakeResponse(list(range(1025)),
                          meta_info={"prompt_tokens": 13, "completion_tokens": 25}),
            _FakeResponse(list(range(1025)),
                          meta_info={"prompt_tokens": 13, "completion_tokens": 25}),
        ]
        stage = GlmImageAR(processor=_FakeProcessor(), vision_language_encoder=None)
        batch = SimpleNamespace(prompt="A simple product sketch", height=1025,
                                width=1001, image_path=None,
                                num_outputs_per_prompt=2, seed=None)

        batch = stage.forward(batch, self._server_args())

        self.assertEqual(batch.usage["prompt_tokens"], 26)
        self.assertEqual(batch.usage["completion_tokens"], 50)
        self.assertEqual(batch.usage["total_tokens"], 76)
```

python/sglang/multimodal_gen/runtime/entrypoints/openai/protocol.py

协议模型层：新增 ImagePromptTokensDetails 与 ImageUsage，ImageResponse 挂载可选 usage，定义了对外的 OpenAI 兼容契约。

```
    # OpenAI 兼容的图像生成 usage 协议模型。
    # prompt_tokens_details 只在 --enable-cache-report 时由响应层填充，
    # 协议层保持可选，避免破坏既有客户端。
    class ImagePromptTokensDetails(BaseModel):
        cached_tokens: int = 0


    class ImageUsage(BaseModel):
        prompt_tokens: Optional[int] = None
        total_tokens: Optional[int] = None
        completion_tokens: Optional[int] = None
        prompt_tokens_details: Optional[ImagePromptTokensDetails] = None
        reasoning_tokens: Optional[int] = 0
        # image_count 是 SGLang 对图像 API 的扩展字段，表示本次实际生成的图片数
        image_count: Optional[int] = None

    class ImageResponse(BaseModel):
```

```
    id: str
    created: int = Field(default_factory=lambda: int(time.time()))
    data: List[ImageResponseData]
    peak_memory_mb: Optional[float] = None
    inference_time_s: Optional[float] = None
    # usage 可选：本地模式或旧版本响应可能不携带该字段
    usage: Optional[ImageUsage] = None
```

评论区精华

review 讨论集中在四个问题上：

- 动机说明：ping1jing2 要求 PR 描述补充为什么需要 token usage 报告；作者给出四点理由（prefix cache 校验、计费统计、与 LLM API 对齐、AR/DiT 分离部署调试），已被接受。
- 新 API 是否冗余：ping1jing2 质疑新增的 flag 是用户可见 API 却未被使用；作者澄清该 flag 正被图像响应路径使用，LLM 侧已有同名参数，二者不冲突。
- 文档更新要求：ping1jing2 在合入前明确要求 "if you add one more API, please update the documentation"，但最终合入的变更集中没有 docs 文件，该要求未在 PR 内落实（reviewer 最终仍 approved）。
- CI 合入门禁：mickqian 质疑为何未通过 Nvidia CI 即合入；ping1jing2 解释失败 CI 与该 PR 无关，且当时所有 GPU CI 均被取消，属于流程执行而非技术问题。
  - PR 描述缺少动机说明 (question): 作者补充四点理由：检查 prefix cache 是否生效、cached tokens 计费规则差异、与 LLM API 对齐、AR/DiT 分离部署调试。
  - 新增用户可见 API 是否未使用 (design): 作者澄清：--enable-cache-report 用于图像响应路径，LLM API 侧 SGLang 已支持同名参数，并非未使用的 API。
  - 新增 API 需同步更新文档 (documentation): 合入的变更集中未见 docs 文件改动，该要求未在 PR 内落实；reviewer 最终仍 approving（pending 遗留）。
  - 未通过 Nvidia CI 即合入 (other): ping1jing2 解释：失败 CI 与该 PR 无关，且 GPU CI 均被取消，属于流程执行问题而非代码缺陷。

风险与影响

- 风险：具体风险点如下：
- 返回类型变更（二元组 → 三元组）：glm_image.py 的 generate_prior_tokens / generate_prior_tokens_batch 返回值结构变化，若仓库内存在未更新的调用方会直接解包崩溃；当前已确认 forward 内两处调用和测试均同步更新，但这是数据契约变更，外部扩展（如自定义 stage）存在遗漏可能。
- usage 合并逻辑的覆盖语义：gpu_worker.py::_merge_expanded_singletons 对非 int 类型的 usage 值直接覆盖而非合并，未来若 usage 中出现嵌套结构（如 prompt_tokens_details）且多输出合并时，非缓存字段可能被覆盖；当前内部 usage 仅含 int 字段，风险有限。
- cached_tokens 剥离的隐式依赖：utils.py 无条件 pop 顶层 cached_tokens 并仅在 --enable-cache-report 时转为 prompt_tokens_details.cached_tokens，如果上游模块已填充 prompt_tokens_details，则不会重复注入但会丢掉顶层字段，语义需要文档明确。

- 同名参数配置分离：multimodal_gen 的 --enable-cache-report 与 SRT LLM 服务的同名参数是两套独立配置对象，用户若只在 AR SRT 服务上开启，图像响应不会生效，存在配置混淆风险。
- 本地模式行为差异：未配置 srt_encoder_url 的本地模式（直接调用 vision_language_encoder.generate）usage 恒为 None，响应中不出现 usage 字段，调用方需兼容有无 usage 两种响应。
- CI 验证不足：合入时 Nvidia GPU CI 被取消，GLM-Image 真实 GPU 回归未完整验证；单元测试只覆盖 mock 路径。
- 影响：影响范围集中在 multimodal_gen 模块的 GLM-Image 路径，但波及 9 个文件、多个子系统：
- 用户侧：GLM-Image 调用方（尤其 AR/DiT 分离部署）首次获得 token 用量反馈，可用于计费、prefix cache 命中率观测和成本核算；默认响应新增 usage 字段为向后兼容扩展，cached_tokens 默认隐藏避免内部成本信息泄漏。
- 系统侧：usage 从 AR stage 的 Req / OutputBatch 一路透传到 HTTP 响应层，涉及 schedule_batch、decoding、gpu_worker、image_api 等多个模块，属于跨模块数据契约变更，影响后续所有 GLM-Image 执行路径。
- 团队侧：为 multimodal_gen 统一 OpenAI 兼容 usage 语义奠定基础，ImageUsage / ImagePromptTokensDetails 协议模型可被其他图像 / 视频模型复用，聚合与拆分逻辑（_merge_srt_usage 系列）也有通用价值。
- 影响程度：中等。默认行为仅是响应新增字段，不破坏既有客户端；但契约变化和文档缺失需要后续跟进。
- 风险标记：跨模块数据契约变更，返回类型变更（二元组→三元组），文档更新未落实，Nvidia GPU CI 未验证即合入，同名配置参数存在混淆风险

关联脉络

- PR #33731 [CI] Fix GLM-Image usage unit tests: 该 PR 修复 GLM-Image usage 单测夹具以匹配本 PR 引入的新契约（如 OutputBatch.usage），是直接由本 PR 触发的后续修复，验证了 usage 字段对测试体系的影响。
- PR #31483 [AMD] ci: run vetted nested multimodal_gen unit tests on AMD: 同为 multimodal_gen 测试域扩展，GLM-Image 相关单测在 AMD CI 上的覆盖与该 PR 的测试配套形成连续脉络。
- PR #33655 [diffusion] Prefer cuDNN SDPA over FA4 for dense attention on sm_100 (B200): 同为 diffusion/multimodal_gen 运行时管线演进，反映该模块持续优化性能与观测能力的整体方向。