PR #25300 完整报告

sgl-project/sglang

feat: optional caller-supplied mm_hashes on GenerateReqInput

合并时间：2026-06-02 02:04

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25300

执行摘要

- 一句话：可选调用方提供 mm_hashes 以实现确定性 pad_value
- 推荐动作：值得精读。该 PR 展示了如何在复杂系统中添加可选调用方集成接口：清晰的文档、优雅的错误处理和完备的单元测试。设计上对十六进制字符串的选择是有远见的。

功能与动机

外部 KV 路由器（如 Dynamo）需要应用层哈希与 sglang 的 pad_value 对齐，以使路由决策与 RadixAttention 前缀缓存一致。当前 sglang 始终在 set_pad_value 内部调用 hash_feature，导致调用方提供的哈希无法生效，路由侧前缀缓存命中成为偶然。

实现拆解

实现分为以下步骤：

1. 数据模型扩展（python/sglang/srt/managers/io_struct.py）：在 GenerateReqInput 数据类中添加 mm_hashes: Optional[Union[List[str], List[List[str]]]] 字段，默认值为 None，并附上详细文档。
2. 请求解析注入（python/sglang/srt/managers/tokenizer_manager.py）：在 _tokenize_one_request 方法中，从 obj.mm_hashes 中解析十六进制字符串，按顺序匹配 mm_inputs 中的 MultimodalDataItem，设置其 hash 属性。处理长度不匹配（警告并忽略）、非 MultimodalDataItem 跳过（兼容性）和解析异常（记录警告，让该项在后续 set_pad_value 中自动回退）。该注入发生在原有的 set_pad_value 循环之前。
3. 引擎 API 透传（python/sglang/srt/entrypoints/engine.py）：在 generate 和 async_generate 方法中添加 mm_hashes 参数，并将其传递给 GenerateReqInput 构造函数，实现完整链路。
4. 向后兼容与回退：所有路径保持默认 None，对于任何异常（长度不匹配、解析失败），日志警告并忽略调用方哈希，沿用原始 hash_feature 逻辑，确保不会因错误输入阻塞请求。
5. 测试覆盖（test/registered/unit/managers/test_mm_hashes.py）：新增单元测试文件，验证 GenerateReqInput.mm_hashes 接受和默认值、set_pad_value 在预置哈希时跳过 hash_feature、相同哈希产生相同 pad_value、不同哈希产生不同 pad_value。

关键文件：

- python/sglang/srt/managers/io_struct.py（模块 数据模型；类别 source；类型 core-logic）：数据模型变更，添加 mm_hashes 字段，是整个功能的入口。
- python/sglang/srt/managers/tokenizer_manager.py（模块 请求处理；类别 source；类型 core-logic）：核心逻辑变更，在 _tokenize_one_request 中解析 mm_hashes 并设置到 MultimodalDataItem.hash。
- python/sglang/srt/entrypoints/engine.py（模块 引擎层；类别 source；类型 core-logic）：引擎 API 层透传 mm_hashes 参数。
- test/registered/unit/managers/test_mm_hashes.py（模块 测试套件；类别 test；类型 test-coverage；符号 TestMmHashesContract, test_generate_req_input_accepts_mm_hashes, test_generate_req_input_defaults_mm_hashes_to_none, test_set_pad_value_honors_preset_hash）：新增完整单元测试，验证 mm_hashes 选项和 set_pad_value 的确定性行为。

关键符号：_tokenize_one_request, generate, async_generate, TestMmHashesContract

关键源码片段

python/sglang/srt/managers/io_struct.py

数据模型变更，添加 mm_hashes 字段，是整个功能的入口。

```
# python/sglang/srt/managers/io_struct.py
# 在 GenerateReqInput 数据类中新增字段 mm_hashes
@dataclass
class GenerateReqInput(BaseReq):
    # ... 其他字段 ...
    # Optional per-image hashes the caller has already computed (hex strings,
    # one per image in `image_data`). When supplied, each MultimodalDataItem's
    # `hash` is initialised from this list and `set_pad_value` skips the
    # internal `hash_feature()` recompute, so the resulting `pad_value` is
    # deterministic from the caller's hash. Intended for external KV routers
    # that compute their own per-image hash for routing decisions and need
    # sglang's prefix-cache key to align. When unset, behavior is unchanged
    # (sglang hashes the processor feature tensor).
    mm_hashes: Optional[Union[List[str], List[List[str]]]] = None
```

python/sglang/srt/managers/tokenizer_manager.py

核心逻辑变更，在 _tokenize_one_request 中解析 mm_hashes 并设置到 MultimodalDataItem.hash。

```
# python/sglang/srt/managers/tokenizer_manager.py
# 在 _tokenize_one_request 方法中，处理完 mm_inputs 后插入
async def _tokenize_one_request(self, obj, ...):
    # ... 之前的代码处理 mm_inputs ...

    # 尝试使用调用方提供的 per-image hashes
    caller_mm_hashes = getattr(obj, "mm_hashes", None)
    if caller_mm_hashes and mm_inputs and mm_inputs.mm_items:
```

```
        if len(caller_mm_hashes) != len(mm_inputs.mm_items):
            # 长度不匹配警告忽略
            logger.warning(
                "mm_hashes length (%d) != mm_items length (%d); "
                "ignoring caller hashes for this request.",
                len(caller_mm_hashes),
                len(mm_inputs.mm_items),
            )
        else:
            for item, hex_hash in zip(mm_inputs.mm_items, caller_mm_hashes):
                if not isinstance(item, MultimodalDataItem):
                    continue # 跳过非标准项
                try:
                    # 将十六进制字符串转换为整数并设置到 hash
                    item.hash = int(hex_hash, 16)
                except (TypeError, ValueError):
                    # 解析失败时记录警告，后续 set_pad_value 会回退
                    logger.warning(
                        "Ignoring malformed mm_hashes entry %r; "
                        "this item will fall back to hash_feature().",
                        hex_hash,
                    )

    # 原有的 set_pad_value 循环，当 hash 已设置时跳过特征计算
    if envs.SGLANG_MM_PRECOMPUTE_HASH.get() and mm_inputs and mm_inputs.mm_items:
        for item in mm_inputs.mm_items:
            if isinstance(item, MultimodalDataItem):
                item.set_pad_value()
    # ... 后续代码 ...
```

评论区精华

该 PR 的 Review 讨论较少。主要的交互是维护者 ishandhanani 触发了 CI 重新运行，并在 CI 通过后确认 'All relevant CI has passed'。设计上选择十六进制字符串而非整数，已在 PR 描述中解释：JSON 兼容性、便于未来扩展更宽哈希。

- CI 重运行与状态确认 (other): CI 通过后合并。

风险与影响

- 风险：风险较低：
  - mm_hashes 为空时无行为变化，完全向后兼容。
  - 解析异常被捕获并回退，不会导致请求失败。
  - 可能的风险：调用方未正确使用 mm_hashes（如顺序错误）会导致路由预期与实际前缀缓存不一致，但这是使用错误而非 Bug。
  - 测试覆盖了关键路径，但缺失端到端（集成）测试。由于涉及外部路由器的协作，端到端验证有限。

- 影响：影响范围中等：
  - 用户：对多模态请求，调用方可选择提供 mm_hashes 以获得确定性的前缀缓存行为，优化跨请求的缓存命中。无需变更现有调用。
  - 系统：在 tokenizer_manager 中添加了解析逻辑，性能开销极小；解析仅发生在提供 mm_hashes 时。
  - 团队：提供了与外部路由器集成的正式协议，促进 sglang 在更广泛系统中的可组合性。
  - 风险标记：核心路径变更，可选字段，缺少端到端测试

关联脉络

- 暂无明显关联 PR