PR #27386 完整报告

sgl-project/sglang

[router] Apply chat template before cache-aware hashing (fix overlap=0 on chat traffic)

合并时间：2026-06-11 01:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27386

1. 执行摘要

本 PR 为 SGLang 实验性路由器的 cache_aware_zmq 策略添加了聊天模板渲染功能，解决了缓存感知哈希在聊天流量上重叠块始终为零的 bug。通过使用 minijinja 渲染模型聊天模板（或为 DeepSeek-V4 使用内置编码器），路由器在哈希前生成与引擎完全一致的 token 序列，从而正确匹配缓存的 KV 块。这一修复将缓存命中率从 0% 恢复至正常水平，显著降低首 token 延迟和显存占用。变更集中在 experimental/sgl-router 模块的 Rust 源码和测试，不影响核心推理路径。

2. 功能与动机

问题

对于聊天模型（如 DeepSeek-V4-Flash），引擎的 KV 缓存基于应用聊天模板后的 token 序列（例如 [0, 128803, 51453, ...]）。但路由器的缓存感知哈希直接对原始 messages[*].content 进行哈希，导致从第一个 token 开始就与引擎存储的块不一致，sgl_router_overlap_blocks_sum 始终为零，缓存感知路由完全失效。

目标

使路由器在哈希前执行与引擎相同的聊天模板渲染，使查询 token 序列与引擎缓存的块完全一致，从而恢复缓存感知路由在聊天流量上的有效性。

3. 实现拆解

步骤	涉及文件	关键变更
1. 聊天模板渲染引擎	chat_template.rs（新增）	使用 minijinja 实现与 HuggingFace 兼容的模板渲染。from_tokenizer_config 方法从 tokenizer_config.json 解析模板源码和特殊 token，配置 trim_blocks、lstrip_blocks、SemiStrict UndefinedBehavior 和 minijinja_contrib::pycompat 以对齐引擎行为。
2. DeepSeek-V4 内置编码器	dsv4.rs（新增）	DSV4 不提供 Jinja 模板，引擎在 Python 中构建 prompt。此文件复现 encoding_dsv4.encode_messages 核心逻辑，包括插入空 system 消息、合并连续 user 轮次、使用与引擎一致的字面量标记，并经过 /tokenize 的字节级对齐验证。

步骤	涉及文件	关键变更
3. Token 注册表扩展	tokenizer/mod.rs（修改）	新增 ChatEncoder 枚举（Jinja / DeepSeekV4）和 ChatEncoderEntry（含 fallback 日志控制）。load_from_config 中通过 resolve_chat_encoder 选择编码器，并在所有分支记录 INFO/WARN 日志，确保模板启用状态可诊断。
4. tokenizer_config.json 加载	adapter.rs（修改）	新增 load_tokenizer_config，用于加载与 tokenizer 同目录或同一 HF 仓库的 tokenizer_config.json。对远程 404（无模板）与网络 / 权限错误进行区分记录。
5. 路由策略调整	cache_aware_zmq.rs（修改）	在 tokens_for_request 中：若请求含 messages 且模型有编码器，调用 encode_chat 渲染后 tokenize；否则走原始 prompt 提取。渲染失败时回退并记录首次 WARN、后续 DEBUG 日志。同时将 extract_prompt_text 拆分为 byte-slice 版本和 extract_prompt_text_from_value，方便测试。
6. 测试与配置更新	多个测试文件和 Cargo.toml	删除 passthrough_chat_template.jinja，端到端测试使用模型真实模板验证对齐；添加 minijinja、minijinja-contrib 等依赖。

experimental/sgl-router/src/tokenizer/chat_template.rs

核心新增文件：实现 HuggingFace 兼容的聊天模板渲染，是解决查询哈希与引擎缓存不匹配的关键。

experimental/sgl-router/src/tokenizer/dsv4.rs

核心新增文件：为 DeepSeek-V4 模型实现内置 prompt 编码器，因为该模型不使用 Jinja 模板，引擎在 Python 中构建 prompt。

关键源码片段

experimental/sgl-router/src/tokenizer/dsv4.rs

核心新增文件：为 DeepSeek-V4 模型实现内置 prompt 编码器，因为该模型不使用 Jinja 模板，引擎在 Python 中构建 prompt。

```
// SPDX-FileCopyrightText: Copyright (c) 2026 The SGLang Authors
// SPDX-License-Identifier: Apache-2.0

//! DeepSeek-V4 prompt 编码器，用于缓存感知路由。
//!
//! DSV4 没有 Jinja 模板；引擎在 Python 中构建 prompt。此模块复现了其核心逻辑，
//! 使路由器哈希的 token 序列与引擎缓存一致。

/// 以下常量与引擎中定义的 marker token 对应的字面量一致。
```

```
const BOS: &str = "<｜begin▁of▁sentence｜>"; // token id 0
const EOS: &str = "<｜end▁of▁sentence｜>"; // token id 1
const USER: &str = "<｜User｜>"; // token id 128803
const ASSISTANT: &str = "<｜Assistant｜>"; // token id 128804
const THINK_END: &str = "</think>"; // token id 128822

/// 将 `messages` 数组渲染为 DSV4 聊天 prompt 字符串。
///
/// 与引擎的 `encoding_dsv4.encode_messages` 对齐（仅聊天模式、纯文本、无 tools/tasks）。
pub fn render_messages(messages: &serde_json::Value) -> String {
    let mut msgs: Vec<(String, String)> = messages
        .as_array()
        .map(|arr| {
            arr.iter()
                .map(|m| {
                    let role = m.get("role")
                        .and_then(|r| r.as_str())
                        .unwrap_or("")
                        .to_string();
                    (role, content_to_string(m.get("content")))
                })
                .collect()
        })
        .unwrap_or_default();

    // 引擎会隐式插入空 system 消息（若非 system 开头）
    if msgs.first().map(|(r, _)| r != "system").unwrap_or(true) {
        msgs.insert(0, ("system".to_string(), String::new()));
    }

    // 合并连续 user 轮次（用

 连接）
    merge_consecutive_user_turns(&mut msgs);

    let mut out = String::from(BOS);
    for i in 0..msgs.len() {
        render_one(i, &msgs, &mut out);
    }
    out
}

/// 合并连续 user 轮次，匹配引擎预处理。
fn merge_consecutive_user_turns(msgs: &mut Vec<(String, String)>) {
    let mut merged: Vec<(String, String)> = Vec::with_capacity(msgs.len());
    for (role, content) in msgs.drain(..) {
        match merged.last_mut() {
            Some((last_role, last_content)) if last_role == "user" && role == "user" => {
                last_content.push_str("
```

```
");
                last_content.push_str(&content);
            }
            _ => merged.push((role, content)),
        }
    }
    *msgs = merged;
}

/// 渲染单个轮次到 out 字符串。
fn render_one(i: usize, msgs: &[(String, String)], out: &mut String) {
    let (role, content) = &msgs[i];
    match role.as_str() {
        "system" => out.push_str(content),
        "user" | "developer" => {
            out.push_str(USER);
            out.push_str(content);
        }
        "assistant" => {
            // 聊天模式无推理块，直接添加标记
            out.push_str(ASSISTANT);
            out.push_str(content);
            out.push_str(EOS);
        }
        _ => {} // 忽略未知 role
    }
}
```

5. 评论区精华

> Review 评论 (gemini-code-assist[bot]): `extract_chat_template` 当前仅处理 String 和 Array 变体。在某些 HuggingFace 配置中，chat_template 以 JSON 对象（map）形式存储，包含多个命名模板（如 {"default": "...", "tool_use": "..."}）。此情况会返回 None，静默禁用聊天模板路由。建议通过查找 "default" 键或回退到第一个模板来支持 Object 变体。

此问题在合并前未被解决，构成一个已知的兼容性限制。后续可通过在 `extract_chat_template` 中增加 Object 分支来补充支持。

6. 风险与影响

风险类型	描述	严重程度
模板兼容性	minijinja 可能不支持所有 Jinja2 特性（如 {%- %} 精确控制、部分 Python 内置方法），导致渲染结果与引擎不同，使缓存感知降级至原始文本哈希。	中

风险类型	描述	严重程度
JSON Object 格式未支持	若模型 tokenizer_config.json 中 chat_template 为 Object，路由模板完全禁用且无显著告警。	中
依赖体积	新增 minijinja 和 minijinja-contrib 依赖，增加二进制体积和编译时间。	低
测试覆盖	仅覆盖了 Qwen3-0.6B 和 DSV4-Flash 模型，其他模型的模板渲染正确性未验证。	中
回退日志	log_fallback 使用 AtomicBool 控制首次 WARN、后续 DEBUG，能有效提示问题但不会淹没日志。	低

影响范围：仅涉及 experimental/sgl-router 模块，不影响核心推理、API 或其他路由策略。对于使用聊天模板的模型，缓存命中率从 0% 恢复正常，显著降低首 token 延迟和显存占用。

7. 关联脉络

该 PR 无直接关联 Issue，但与以下历史开发方向紧密相关：

- DeepSeek-V4 支持（PR #27529、#27380）：这些 PR 在引擎端修复 DSV4 的注意力后端和数据类型问题，本 PR 则在路由器端对齐 DSV4 的 prompt 编码，形成完整的 DSV4 缓存感知路由链路。
- sgl-router 缓存感知基础设施（PR #17260 的 ngram spec v2、#27695 的 KVWriteLoc 重构）：路由器不断在提高缓存感知路由的准确性和性能，本 PR 补上了 chat 模板这一个关键缺口。
- 测试基础设施演化：之前依赖 passthrough 模板来对齐哈希（见 PR #17260 中的相关配置），本 PR 移除了这一 hack，使用真实模板进行端到端验证，提升了测试的真实性。

后续可能的工作包括：支持 Object 格式聊天模板、添加 per-request fallback 计数器 metric、扩展到更多模型类型的模板验证。