

PR #57045 完整报告

vllm-project/vllm

[UX] Add thinking support to `vllm chat`

合并时间: 2026-09-16 16:54

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/57045>

执行摘要

- 一句话: vllm chat 流式展示 reasoning, 思考阶段不再静默
- 推荐动作: 值得精读, 但只需 5 分钟: 重点看 `_print_chat_stream` 的状态机与 TTFT 打点两处设计决策, 而不是 19 行的体量。
- 想理解 vLLM 前端如何处理 reasoning 字段差异的读者, 可以关注 `getattr(delta, "reasoning", ...)` or `getattr(delta, "reasoning_content", ...)` 这种 "双字段兼容 + 安全降级" 的写法, 这在对接多家 OpenAI 兼容后端时是低成本高收益的模式。
- 关注 CLI/ 前端体验的工程师应注意到 "thinking 不进返回值" 这一取舍: 它保证多轮历史不被思考链污染, 但也意味着需要 reasoning 回传的 interleaved thinking 场景不能靠这里兜底。
- 如果你维护下游脚本消费 vllm chat 输出, 需要评估新增 `<think>` / `</think>` 标签对解析逻辑的冲击。

功能与动机

PR body 明确说明动机: "When using `vllm chat` there was no output while the model was thinking when using a reasoning parser, so this PR starts reading from `reasoning` or `reasoning_content` fields to show those deltas". 作者贴出的 Before 截图注释为 "very long wait while thinking before response", After 为 "immediate response streaming after `<think>` tag", 可见痛点是首字等待体感极差, 而非功能缺失。

实现拆解

1. 入口与作用域: 唯一改动文件是 `vllm/entrypoints/cli/openai.py`, 核心函数 `_print_chat_stream` 是 `vllm chat` 交互式会话中消费 OpenAI 流式响应的唯一渲染与统计入口, 其返回值会被用于拼接该轮助手回复。
2. 新增 reasoning 解析与双字段兼容: 在循环内对 `chunk.choices[0].delta` 增加 `reasoning = getattr(delta, "reasoning", None) or getattr(delta, "reasoning_content", None)`。
vLLM 的 reasoning parser 通常输出 `reasoning_content`, 部分 OpenAI 兼容后端输出 `reasoning`; 用 `getattr` 而非属性直取, 可在旧版 SDK 或未开 reasoning 的场景下安全降级为 `None`。
3. 思考内容的状态机打印: 新增局部变量 `in_reasoning: bool`。首个 reasoning 增量到达时打印 `<think>` 并置位, 随后逐增量透传; 正文 `delta.content` 到达时先补 `\n</think>` 再复

位，然后照旧累加并打印正文。流结束时若仍处于 `in_reasoning`（例如 `max_tokens` 在思考中途耗尽），再补一次 `</think>`，避免终端留下悬空标签。

4. TTFT 语义调整：原来 TTFT 只在首个 `delta.content` 处打点，现在改为 `if ttft is None and (reasoning or delta.content)`，即首个可见字符（思考或正文）即计时，使 `--stats` 的 TTFT 更贴近用户真实感知。
5. 返回值保持不变：`output` 仅累加 `delta.content`，`thinking` 文本不进入返回值，因此多轮 `vllm chat` 历史里不会把思考链回灌给模型。
6. 测试与文档配套：本次没有任何测试文件、文档或配置改动；同文件的 `_print_completion_stream`（`/v1/completions` 的 `chunk.choices[0].text` 路径）也未同步支持 `reasoning`。

关键文件：

- `vllm/entrypoints/cli/openai.py`（模块 命令行；类别 `source`；类型 `core-logic`；符号 `_print_chat_stream`, `_print_metrics`）：唯一的改动文件，也是 `vllm chat` 交互式会话的流式渲染与统计入口；`_print_chat_stream` 的新状态机决定了思考内容何时、以何种标签展示，并顺带调整了 TTFT 打点位置。

关键符号：`_print_chat_stream`, `_print_metrics`

关键源码片段

`vllm/entrypoints/cli/openai.py`

唯一的改动文件，也是 `vllm chat` 交互式会话的流式渲染与统计入口；`_print_chat_stream` 的新状态机决定了思考内容何时、以何种标签展示，并顺带调整了 TTFT 打点位置。

```
# vllm/entrypoints/cli/openai.py
# `vllm chat` 的核心流式渲染函数：本次改动集中在这里，负责把
# OpenAI 兼容的流式 delta 逐块打印到终端，同时统计 TTFT 与 TPS。
def _print_chat_stream(stream, stats: bool = False) -> str:
    # output 只累加正文 content，thinking 文本不会进入返回值，
    # 因此多轮会话历史里不会把模型的思考链再回灌给模型。
    output: str = ""
    # in_reasoning 用于记录当前是否处于 <think> 段落内，
    # 决定何时补上开闭标签，避免标签重复或悬空。
    in_reasoning: bool = False
    start: float = time.perf_counter()
    ttft: float | None = None
    completion_tokens: int = 0
    for chunk in stream:
        # usage 通常只出现在最后一个（或 allow_usage 开启时的每个）chunk。
        if chunk.usage is not None:
            completion_tokens = chunk.usage.completion_tokens
        if not chunk.choices:
            continue
        delta = chunk.choices[0].delta
        # 兼容两种 reasoning 字段命名：vLLM reasoning parser 一般给
        # reasoning_content，部分 OpenAI 兼容后端给 reasoning；
```

```

# 用 getattr 兜底可以避免旧版 SDK 缺少该字段时抛 AttributeError。
reasoning = getattr(delta, "reasoning", None) or getattr(
    delta, "reasoning_content", None
)
# TTFT 打点从 " 首个正文 token" 提前到 " 首个可见 token",
# 让 --stats 输出的 TTFT 与用户实际看到首字的时刻对齐。
if ttft is None and (reasoning or delta.content):
    ttft = time.perf_counter() - start
if reasoning:
    if not in_reasoning:
        # 只在思考段开始时打印一次开标签。
        print("<think>", flush=True)
        in_reasoning = True
    print(reasoning, end="", flush=True)
if delta.content:
    if in_reasoning:
        # 正文出现即代表思考结束，补一个闭合标签并复位状态。
        print("\n</think>", flush=True)
        in_reasoning = False
    output += delta.content
    print(delta.content, end="", flush=True)
if in_reasoning:
    # 流在思考中途结束（例如 max_tokens 耗尽）时仍要闭合标签，
    # 否则终端会留下未关闭的 <think> 段落。
    print("\n</think>", end="", flush=True)
print()
if stats:
    _print_metrics(start, ttft, completion_tokens)
return output

```

注意：同文件的 `_print_completion_stream` 本次未同步改动，
`/v1/completions` 路径（`chunk.choices[0].text`）仍不支持 reasoning 流。

评论区精华

讨论非常轻量，只有两条人工审批与一条 bot 提示：

- DarkLight1337 (APPROVED) : "Could you attach sample output in the PR description for illustration?" —— 唯一实质性反馈，属于可验证性 / 文档性要求，作者随后补上了 Before/After 截图。
- hmellor (APPROVED) : 空评论直接放行并合并，说明 maintainer 认为改动范围足够安全。
- claude[bot] (COMMENTED) : fork PR 不跑自动 review，需 maintainer 手动 @claude review 触发，本 PR 实际未执行自动审查。
- Issue 评论区只有 mgoin 的 `/ci run` 与 github-actions 触发的 Buildkite CI #89136，属于常规 CI 触发，无技术讨论。

值得注意的是：TTFT 口径变化、`<think>` 标签硬编码写入 stdout、completions 路径未同步这三点，review 中均无人提及。

- 要求在 PR 描述中补充示例输出 (documentation): 作者在 PR 描述中补充了 Before/After 两张终端截图 (Before 标注 "very long wait while thinking before response")，并获 hmellor 批准合并。
- TTFT 统计口径变更未被讨论 (correctness): 变更被默认接受并随 PR 合并；属于隐式行为变更，建议后续在 `--stats` 文档或 changelog 中说明。
- fork PR 未执行自动 review (other): 仅有人工轻量审批 (DarkLight1337、hmellor)，无人审查 `_print_completion_stream` 未同步、`<think>` 标签硬编码等边界问题。

风险与影响

- 风险:
 1. CLI 输出格式变更 (低 - 中) : `<think>` / `</think>` 直接 print 到 stdout 且无开关。把 `vllm chat` 输出重定向到文件或管道、或用脚本做快照对比的用户，会看到额外标签；没有 `--no-thinking` 之类的回退选项，只能用 `--stats` 之外的方式绕过。
 1. TTFT 统计口径变化 (中) : `ttft` 打点提前到首个 reasoning token，所有启用 `--stats` 的用户在开启 reasoning parser 后 TTFT 数值语义改变，历史数据不可直接比较。对不使用 reasoning parser 的模型无影响。
 2. 状态机边界 (低 - 中) : 若服务端在单次响应内多次交错返回 reasoning 与 content，会反复打印成对的 `<think>` / `</think>`；流被截断时的闭合分支已处理，但没有测试覆盖这些分支。
 3. 缺少测试覆盖 (中) : 仓库没有针对 `_print_chat_stream` 的流式打印与 TTFT 语义的回归测试，本次也未新增，状态机与 `getattr` 兼容逻辑属于 "改坏不易被发现" 的类型。
 4. 路径不一致 (低) : `_print_completion_stream` 仍只读 `text`，走 completions 端点的 reasoning 流照旧静默，两个渲染函数的用户体验出现分层。
 5. 类型注解强化无风险: `output: str`、`start: float`、`completion_tokens: int` 等显式注解是纯风格改动，不改变运行行为。 - 影响:
 - 用户影响: 仅影响 `vllm chat` 命令行交互体验，且是正向提升 —— 使用 DeepSeek / Qwen / Kimi 等带 reasoning parser 的模型时，首字体感延迟从数十秒降到首个思考 token 到达，作者截图已直观展示。
 - 系统影响: 不触碰推理引擎、调度器、KV cache、API server，对吞吐、延迟、显存零影响。
 - 团队与生态影响: 改动位于 `vllm/entrypoints/cli`，与 Rust 前端、Responses API 等前端能力补齐方向同属一条线；`<think>` 标签格式虽未写入文档，但可能被用户视为既定输出格式，后续调整需注意兼容。
 - 后续工作: completions 路径同步、reasoning 展示开关、CLI 流式输出测试是三个自然的 follow-up。
 - 风险标记: CLI 输出格式变更，TTFT 统计口径变化，缺少测试覆盖，completions 路径未同步，无关闭开关

关联脉络

- PR #55912 [Doc] Show how to get Responses prompt token IDs: 同属 frontend 模块下 " 用户可见输出 / 渲染行为 " 的改动, 该 PR 选择补文档说明如何获取 prompt token ID; 对照可见本 PR 改变了 vllm chat 的可见输出格式却未更新任何文档。
- PR #56931 [Rust Frontend] Support HF config overrides --hf-overrides: 同为前端 CLI/ 配置能力的补齐, 反映 vLLM 前端 (Python CLI 与 Rust 前端) 并行扩张命令与选项的演进脉络。
- PR #57033 [Rust Frontend][Metrics] Add per-request preemption histogram: 同属前端指标体系改进方向, 与本 PR 调整 TTFT 打点以贴近真实用户体验的思路一致, 可作为后续 reasoning 相关指标 (如首思考 token 延迟) 的参考。