PR #46137 完整报告

vllm-project/vllm

[Rust Frontend] Support thinking_token_budget for chat and completions

合并时间：2026-06-22 16:00

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46137

执行摘要

本 PR 在 Rust 前端的三个 API 入口（chat completions、completions、inference generate）中新增了 thinking_token_budget 参数的支持，并通过一个统一的规范化函数对齐 Python 前端的验证逻辑。主要变更集中在 Rust crate 的 text 模块（降层）和 server 模块（入口转换），附带测试和错误映射。这是 Rust 前端请求兼容性追踪项 #44280 的一部分。

功能与动机

目的

填补 Rust 前端与 Python 前端在 thinking_token_budget 参数支持上的差距。此前 Rust chat 端点显式拒绝该参数（返回"not supported"），completions 端点甚至未解析该参数。引擎核心早已支持该参数（#20859），因此该缺失纯粹是前端 gap。

引用

PR body 明确说明："Add support for the thinking_token_budget request parameter in the Rust frontend, for both /v1/chat/completions and /v1/completions, reaching parity with the Python frontend"。

实现拆解

1. 协议层扩展（rust/src/engine-core-client/src/protocol/mod.rs）：在 EngineCoreSamplingParams struct 中添加 thinking_token_budget: Option<u64> 字段，作为引擎核心消费的 DTO。字段注释说明 None 表示无限制。
2. 核心规范化函数（rust/src/text/src/lower.rs）：
   - 新增 normalize_thinking_token_budget，将用户侧 Option<i64> 转换为 Option<u64>，遵守与 Python 完全相同的规则：None/-1 → None；非负 → Some(值)；其他负 → 错误。
   - 在 lower_sampling_params 中解构 thinking_token_budget，调用规范化函数，并将结果赋值给 engine params。
3. 三个入口点透传（rust/src/server/src/routes/openai/chat_completions/convert.rs、completions/convert.rs、inference/generate/convert.rs）：每个 prepare 函数中，将请求中的 thinking_token_budget 直接填入 SamplingParams 对应字段，不做验证。——验证延迟到降层统一处理。

4. 错误映射（rust/src/server/src/error.rs）：将 InvalidThinkingTokenBudget 加入 is_request_validation_error 匹配分支，确保返回 400 和 invalid_request_error 类型。
5. 清理旧拒绝逻辑（rust/src/server/src/routes/openai/chat_completions/validate.rs）：移除之前显式返回"thinking_token_budget is not supported"的代码。
6. 测试覆盖：每个转换层新增独立测试验证字段透传；lower.rs 新增规范化测试覆盖所有分支（正常值、0、-1、None、超大值、非法负值）；error.rs 新增错误映射测试；其他现有测试补充默认字段值以保持编译。

rust/src/text/src/lower.rs

核心降层逻辑，新增 normalize_thinking_token_budget 函数，在 lower_sampling_params 中进行参数规范化，是 PR 的枢纽模块。

```
/// 将用户侧的 thinking_token_budget（Option<i64>）规范化为引擎侧 Option<u64>。
/// 与 Python validate_thinking_token_budget 语义一致：
/// - None 或 -1 → None（无限制）
/// - 非负值 → Some(value)（直接传递）
/// - 其他负值 → 错误 InvalidThinkingTokenBudget
fn normalize_thinking_token_budget(value: Option<i64>) -> Result<Option<u64>> {
    match value {
        None | Some(-1) => Ok(None), // 视作无限制
        Some(budget) if budget >= 0 => Ok(Some(budget as u64)), // 有效预算
        Some(_) => Err(Error::InvalidThinkingTokenBudget), // 非法负值
    }
}

// 在 lower_sampling_params 中使用：
let thinking_token_budget = normalize_thinking_token_budget(thinking_token_budget)?;
// 然后赋值给 EngineCoreSamplingParams 的对应字段
```

评论区精华

> Codex 机器人："引擎端可能拒绝请求返回 500" 作者："引擎端 ThinkingBudgetStateHolder 在 reasoning_config 为 None 时静默忽略，不会 500。" BugenZhao："managed-engine 模式下 reasoning parser 选择未传递，导致 budget 被静默忽略，后续 PR 我将一并修复。" 作者："感谢！"

核心讨论围绕 managed-engine 模式的已知限制，PR 本身正确实现了基础功能，限制将在后续跟进解决。

风险与影响

- 主要风险：Rust managed-engine 模式下，若 --reasoning-parser 未在两边同步配置，thinking_token_budget 会被静默忽略。用户可能预期生效但实际未生效。当前无运行时告警。
- 兼容性风险：低，字段新增且遵循已有降层模式。
- 影响范围：对使用 Rust 前端进行推理预算控制的用户有直接影响（现在可以发送参数）。影响程度中等。

关联脉络

- 跟踪 issue #44280（请求兼容性和验证）是本 PR 的上层背景，该 issue 涵盖多个参数的 Rust 前端对等工作。
- V1 引擎自 #20859 开始支持 thinking_token_budget，Python 前端早已暴露。
- 后续将有针对 reasoning parser 传递的跟进 PR，以确保 managed-engine 模式下 budget 完全生效。