PR #46314 完整报告

vllm-project/vllm

[Frontend] Port seed_oss to the streaming parser engine as a Qwen3 subclass

合并时间：2026-06-25 08:08

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46314

执行摘要

- 一句话：将 seed_oss 解析器重构为 Qwen3Parser 子类，复用流式解析引擎
- 推荐动作：值得精读。PR 展示了如何通过继承和参数化重用 parser 引擎，是清理重复代码的优秀范例。

功能与动机

消除重复代码，利用已有的 Qwen3Parser 和流式解析引擎 (#45413) 提供的参数类型强制转换，减少维护成本。PR body 说明 seed_oss 的 tool-call/reasoning 语法与 Qwen3 完全相同，仅包装 token 不同，因此可以继承 Qwen3Parser 并仅覆写 token。

实现拆解

1. 参数化 qwen3_config()：在 vllm/parser/qwen3.py 中修改 qwen3_config() 函数，使其接受可选的四个包装 token 参数（think_start, think_end, tool_start, tool_end），默认保持 Qwen3 原始值，确保 Qwen3 行为不变。
2. 添加 SeedOssParser 类：在 vllm/parser/seed_oss.py 中创建 SeedOssParser(Qwen3Parser)，仅设置 CONFIG_NAME = "seed_oss" 和四个 seed: 前缀的 token（<seed:tool_call>, </seed:tool_call>, <seed:think>, </seed:think>），所有状态转移和参数转换逻辑继承自 Qwen3Parser。
3. 注册引擎适配器：在 vllm/tool_parsers/seed_oss_engine_tool_parser.py 和 vllm/reasoning/seed_oss_engine_reasoning_parser.py 中通过 make_adapters(SeedOssParser) 创建薄的适配器类。在 vllm/parser/engine/registered_adapters.py 中注册 seed_oss 条目，指向新增适配器。
4. 删除旧代码：删除手写的 vllm/tool_parsers/seed_oss_tool_parser.py（633 行）和 vllm/reasoning/seedoss_reasoning_parser.py（27 行），以及对应的旧测试文件 tests/tool_parsers/test_seed_oss_tool_parser.py（522 行）和 tests/reasoning/test_seedoss_reasoning_parser.py（236 行）。
5. 测试覆盖：新增 tests/parser/engine/test_seed_oss.py（189 行），覆盖 token 覆盖、单次工具调用、流式、reasoning 转工具边界等场景。在 tests/parser/engine/trace_builder.py 中添加 seed_oss 重放构建，使 parser engine 的统一重放测试自动覆盖 SeedOssParser。

关键文件：

- vllm/parser/seed_oss.py（模块 解析器；类别 source；类型 core-logic；符号 SeedOssParser）：新增的 SeedOssParser 定义，继承 Qwen3Parser 并仅覆写四个 seed: 包装 token，是整个重构的核心。
- tests/parser/engine/test_seed_oss.py（模块 测试；类别 test；类型 test-coverage；符号 mock_tokenizer, tool_parser, parser, test_token_overrides_wired）：新增的 engine-based 测试，覆盖 token 覆盖、单次工具调用、malformed header 回归、流式、reasoning 转工具边界等场景，并注册到重放构建器。
- vllm/parser/qwen3.py（模块 解析器；类别 source；类型 core-logic；符号 qwen3_config）：被修改以支持参数化 token，使 Qwen3Parser 可被子类化并复用全部逻辑。

关键符号：SeedOssParser.init, Qwen3Parser.qwen3_config

关键源码片段

vllm/parser/seed_oss.py

新增的 SeedOssParser 定义，继承 Qwen3Parser 并仅覆写四个 seed: 包装 token，是整个重构的核心。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project
"""seed_oss parser for tool calls and reasoning.

seed_oss shares the Qwen3 XML grammar exactly; only the four wrapper
token strings differ::

    <think> -> <seed:think>
    </think> -> </seed:think>
    <tool_call> -> <seed:tool_call>
    </tool_call> -> </seed:tool_call>

``<function=...>`` and ``<parameter=...>`` are byte-identical, so the
entire transition table and ``_qwen3_arg_converter`` are inherited from
:class:`Qwen3Parser` unchanged.
"""

from __future__ import annotations

from vllm.parser.qwen3 import Qwen3Parser


class SeedOssParser(Qwen3Parser):
    # 配置名用于注册和路由
    CONFIG_NAME = "seed_oss"
    # 覆盖四个包装 token，其余逻辑全部继承自 Qwen3Parser
    THINK_START = "<seed:think>"
    THINK_END = "</seed:think>"
    TOOL_START = "<seed:tool_call>"
    TOOL_END = "</seed:tool_call>"
```

tests/parser/engine/test_seed_oss.py

新增的 engine-based 测试，覆盖 token 覆盖、单次工具调用、malformed header 回归、流式、reasoning 转工具边界等场景，并注册到重放构建器。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project
"""Tests for the engine-based seed_oss parser.

seed_oss is Qwen3 with four overridden wrapper tokens, so the shared grammar
(arg types, multiline values, parallel calls, streaming mechanics, …) is
already covered by ``test_qwen3.py``/``test_qwen3_reasoning.py``. These tests
cover only what is seed_oss-specific: that the ``seed:`` token overrides are
wired through, the reasoning→tool boundary holds with them, the malformed
header from #46314 no longer drops sibling calls, and the registered adapters
resolve. Seed-specific budget-reflect tags inside reasoning are also covered
here because the old dedicated parser tests exercised them.
"""

import json

import pytest

from tests.parser.engine.conftest import make_mock_tokenizer
from tests.parser.engine.streaming_helpers import (
    collect_function_name,
    collect_tool_arguments,
    simulate_reasoning_streaming,
    simulate_tool_streaming,
)
from vllm.parser.engine.registered_adapters import (
    SeedOssParserReasoningAdapter,
    SeedOssParserToolAdapter,
)
from vllm.parser.seed_oss import SeedOssParser

TOOL_CALL_START = "<seed:tool_call>"
TOOL_CALL_END = "</seed:tool_call>"
THINK_START = "<seed:think>"
THINK_END = "</seed:think>"

_THINK_END_ID = 51
_TOOL_CALL_ID = 60

# mock 词汇表包含 seed_oss 特有的 token
_SEED_OSS_VOCAB = {
    THINK_START: 50,
    THINK_END: _THINK_END_ID,
    TOOL_CALL_START: _TOOL_CALL_ID,
```

```
    TOOL_CALL_END: 61,
}


@pytest.fixture
def mock_tokenizer():
    return make_mock_tokenizer(_SEED_OSS_VOCAB)


@pytest.fixture
def tool_parser(mock_tokenizer):
    # 关闭 thinking 以便专注于 tool 解析
    return SeedOssParser(
        mock_tokenizer, chat_template_kwargs={"enable_thinking": False}
    )


@pytest.fixture
def parser(mock_tokenizer):
    # 允许 thinking 的解析器
    return SeedOssParser(mock_tokenizer)


def test_token_overrides_wired(parser):
    # 验证配置名和 token 覆盖正确连接
    assert parser.parser_engine_config.name == "seed_oss"
    assert parser.reasoning_start_str == THINK_START
    assert parser.reasoning_end_str == THINK_END


def test_single_tool_call(tool_parser, mock_request):
    text = (
        f"{TOOL_CALL_START}\n<function=get_weather>\n"
        "<parameter=city>Tokyo</parameter>\n"
        f"</function>\n{TOOL_CALL_END}"
    )
    result = tool_parser.extract_tool_calls(text, mock_request)
    assert result.tools_called is True
    assert result.tool_calls[0].function.name == "get_weather"
    assert json.loads(result.tool_calls[0].function.arguments) == {"city": "Tokyo"}
```

评论区精华

- sfeng33 提议将 seed_oss 作为 Qwen3Parser 子类移植到流式解析器引擎 (#45413)，EazyReal 同意并执行。
- bbrowning 提醒旧的 seed_oss 解析器有关于 <seed:cot_budget_reflect> 的测试，手动验证新实现通过。EazyReal 随后在测试中加入了该场景。
- 需要 rebase 解决冲突（由 mergify 和 sfeng33 指出），最终 rebase 成功。

- 建议将 seed_oss 移植为 Qwen3Parser 子类 (design): 采纳建议，实施子类化方案。
- 旧测试场景（cot_budget_reflect）的覆盖 (testing): EazyReal 在测试中加入了该场景，确保回归覆盖。
- 需 rebase 解决冲突 (other): EazyReal 成功 rebase 到 main，冲突解决。

风险与影响

- 风险:
  - 核心风险：如果 Qwen3Parser 的行为在未来发生变化，可能隐式影响 seed_oss。但 seed_oss 通过继承共享逻辑，维护成本反而降低。
  - 回归风险：旧 parser 的测试被删除，但 engine-based 测试覆盖了核心场景，且重放测试自动覆盖。reviewer 手动验证了旧测试场景，风险可控。
  - 兼容性风险：对外接口不变（tool-call-parser/reasoning-parser 名称仍然为 "seed_oss"），用户无感知。
- 影响:
  - 用户：无需任何配置更改，seed_oss 模型用户继续使用 --tool-call-parser seed_oss 和 --reasoning-parser seed_oss。
  - 系统：代码量净减 1186 行，减少维护负担。SeedOssParser 自动获得引擎的 bug 修复和功能增强。
  - 团队：其他模型可以参考此模式继承现有 parser，减少重复开发。
  - 风险标记：核心路径变更，缺少测试覆盖（旧测试删除），继承依赖风险

关联脉络

- PR #45413 [Frontend] Streaming parser engine: 本 PR 构建于流式解析器引擎之上，SeedOssParser 继承自 Qwen3Parser，并利用引擎的适配器注册机制。