PR #48048 完整报告

vllm-project/vllm

feat(frontend): session id plumbing into requests

合并时间：2026-08-04 06:18

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48048

执行摘要

- 一句话：新增一级 session_id，双栈贯穿请求链路至引擎核心
- 推荐动作：值得精读，尤其是 Rust 侧 resolve_session_id 的优先级设计与 EngineCoreRequest 作为 array-like msgpack 结构演进时必须 " 追加末尾 " 的 wire 约束。对后续 session-aware 调度 / KV 策略 PR 有前置理解价值。建议合入后人工核对 Python 与 Rust 两侧 EngineCoreRequest 字段顺序是否一致，并关注是否补跨语言端到端测试。

功能与动机

PR body 自述这是 #48049 系列的第 1 步："Add a first-class request-level session_id to vLLM so related requests in the same conversation, agent run, or session can carry stable identity without overloading request_id or storing session identity in SamplingParams.extra_args"。现状下会话身份没有结构化承载位置，只能塞进 vllm_xargs（最终落入 SamplingParams.extra_args），既破坏采样参数语义，引擎内部也无法统一读取；而 P/D 分离、KV connector、会话级 cache 等场景需要请求携带稳定 session 身份才能做会话级策略。因此本 PR 先把身份贯通到引擎内部，明确不做 session 感知调度或路由策略。

实现拆解

1. 协议层（Python 请求体字段）：vllm/entrypoints/openai/chat_completion/protocol.py、completion/protocol.py、responses/protocol.py 三处为请求模型新增可选字段 session_id: str | None，属于向前兼容的 API 扩展，旧客户端不受影响。
2. 入口解析层（Python 身份来源合并）：vllm/entrypoints/generate/base/serving.py 新增静态方法 _get_session_id_from_headers（读取 X-Session-ID 头，raw_request 为 None 时安全返回）与 _get_session_id（优先级：请求体 session_id > X-Session-ID 头 > vllm_xargs["session_id"] 兼容 fallback，空字符串与非字符串值一律忽略）；SESSION_ID_HEADER 常量统一头名。review 后补齐 tokens-in 路径 serve_tokens 到 engine.generate 的透传。
3. Rust 前端（请求上下文与转换）：rust/src/server/src/utils.rs 给 ResolvedRequestContext 增加 session_id，resolve_request_context 从 X-Session-ID 头解析并过滤空值；新增 resolve_session_id(ctx, request_session_id, xargs) 按 请求体 > 头 > vllm_xargs 优先级合并，vllm_xargs 取值强制 Value::as_str。completions/convert.rs 与 chat_completions/convert.rs 在构造 TextRequest / ChatRequest 前计算 session_id 并写入字段，模式与既有 data_parallel_rank 透传一致；

rust/src/text、rust/src/llm、rust/src/chat 的请求结构同步加字段。

4. 引擎核心与内部请求：vllm/v1/engine/__init__.py 中 EngineCoreRequest 增加 session_id: str | None = None；vllm/v1/engine/async_llm.py 的 add_request / _add_streaming_input_request / generate 增加同名参数并逐层透传；vllm/v1/request.py 的 Request 增加字段并在 from_engine_core_request 复制。并行采样展开子请求时通过复制父请求继承身份。Rust engine-core-client 的 EngineCoreRequest 以 serde_tuple 镜像 Python 侧，测试把数组长度断言从 20 改为 21 并验证 session_id 落位 array[20]。
5. gRPC 与测试配套：按 njhill 建议向 rust/proto/inference.proto 增加 session_id 字段完成 gRPC 面透传；新增 tests/entrypoints/openai/test_session_id.py 覆盖三类请求 × 三通道优先级及非法值过滤；另在 tokens-in 流（test_generate_stream.py）、并行采样（test_parallel_sampling.py）、内部请求拷贝（test_request.py）处补测试。

关键文件：

- rust/src/server/src/utils.rs（模块 请求解析；类别 source；类型 core-logic；符号 resolve_session_id, resolve_request_context）：Rust 侧 session_id 解析核心：ResolvedRequestContext 增加 session_id 头解析，新增 resolve_session_id 统一三通道优先级，是 completions 与 chat_completions 两个入口共同依赖的汇合点。
- vllm/entrypoints/generate/base/serving.py（模块 入口服务；类别 source；类型 core-logic；符号 _get_session_id_from_headers, _get_session_id）：Python 侧入口解析核心：_get_session_id 定义了 body > header > vllm_xargs 的行为契约，也是 review 中 tokens-in 路径漏传风险的所在模块。
- tests/entrypoints/openai/test_session_id.py（模块 会话测试；类别 test；类型 test-coverage；符号 _raw_request, test_get_session_id_accepts_body_field, test_get_session_id_accepts_session_header, test_get_session_id_ignores_correlation_header）：新增的 97 行测试文件把 Python 侧 session_id 三通道优先级固化为行为契约，覆盖 chat / completion / responses 三类请求，并明确排除 X-Correlation-ID 与非法值。
- vllm/v1/engine/__init__.py（模块 引擎协议；类别 source；类型 data-contract；符号 EngineCoreRequest）：EngineCoreRequest 是跨语言 wire format 的契约点，session_id 的插入位置直接影响 Rust engine-core-client 的 serde tuple 解码；review 中 claude[bot] 的兼容性评论正指向此处。
- rust/src/server/src/routes/openai/completions/convert.rs（模块 请求转换；类别 source；类型 entrypoint；符号 prepare_completion_request_threads_body_session_id, prepare_completion_request_uses_vllm_xargs_session_id_fallback, prepare_completion_request_ignores_empty_and_non_string_session_id_values）：completion 入口接入 resolve_session_id 并写入 TextRequest.session_id，附 3 个针对性单元测试覆盖 body 优先、vllm_xargs fallback 与非法值忽略。
- rust/src/server/src/routes/openai/chat_completions/convert.rs（模块 请求转换；类别 source；类型 entrypoint；符号 prepare_chat_request_threads_header_session_id, prepare_chat_request_ignores_correlation_header, prepare_chat_request_ignores_empty_session_header_without_fallback）：chat 入口

同款接入，测试明确忽略 X-Correlation-ID 与空 X-Session-ID 头，是会话身份语义澄清的关键样本。

- rust/src/engine-core-client/src/protocol/request.rs（模块 客户端协议；类别 source；类型 data-contract；符号 EngineCoreRequest）：Rust 侧 EngineCoreRequest 以 serde_tuple 镜像 Python 协议；测试把数组长度断言从 20 改为 21 并把 session_id 放在 array[20]（末尾），是验证 wire format 的关键锚点。
- vllm/v1/engine/async_llm.py（模块 引擎 API；类别 source；类型 core-logic；符号 add_request, generate, _add_streaming_input_request）：async_llm 的 add_request / _add_streaming_input_request / generate 链新增 session_id 参数并逐层透传，是 Python 引擎 API 消费身份的入口。
- tests/v1/engine/test_parallel_sampling.py（模块 并行采样；类别 test；类型 test-coverage；符号 test_parallel_sampling_child_requests_preserve_session_id）：验证并行采样展开子请求时通过复制保留 session_id，保证 n>1 采样不丢身份。
- tests/entrypoints/scale_out/token_in_token_out/test_generate_stream.py（模块 生成流；类别 test；类型 test-coverage；符号 test_serve_tokens_threads_session_id_header_to_engine）：对应 review 中 bongwoobak 指出的 tokens-in 漏传问题，验证 X-Session-ID 头经 serve_tokens 透传到 engine.generate。

关键符号：_get_session_id, _get_session_id_from_headers, resolve_session_id, resolve_request_context, prepare_completion_request, prepare_chat_request, add_request, generate

关键源码片段

rust/src/server/src/utils.rs

Rust 侧 session_id 解析核心：ResolvedRequestContext 增加 session_id 头解析，新增 resolve_session_id 统一三通道优先级，是 completions 与 chat_completions 两个入口共同依赖的汇合点。

```
// rust/src/server/src/utils.rs
// 统一解析 session_id 的优先级：请求体字段 > HTTP 头（已解析进 ctx.session_id）> vllm_
xargs["session_id"]。
// 与 Python 侧 _get_session_id 保持同构，保证双栈行为一致。
pub fn resolve_session_id(
    ctx: &ResolvedRequestContext,
    request_session_id: Option<&str>,
    xargs: Option<&HashMap<String, Value>>,
) -> Option<String> {
    request_session_id
        .filter(|value| !value.is_empty())
        .map(str::to_owned)
        .or_else(|| ctx.session_id.clone())
        .or_else(|| {
            // vllm_xargs 里的值必须是字符串才采用，其他 JSON 类型（如 int）直接忽略
            xargs
                .and_then(|map| map.get("session_id"))
```

```
                .and_then(Value::as_str)
                .filter(|value| !value.is_empty())
                .map(str::to_owned)
        })
}

// resolve_request_context 现在会顺带从 X-Session-ID 头提取会话身份，
// 与 X-Request-Id、X-data-parallel-rank 一并放入 ResolvedRequestContext，
// 供各 route 的 convert 逻辑统一消费。
pub fn resolve_request_context(
    headers: &HeaderMap,
    request_id: Option<&str>,
) -> ResolvedRequestContext {
    // 此处省略 request_id / data_parallel_rank 的既有解析逻辑
    let session_id = headers
        .get("X-Session-ID")
        .and_then(|value| value.to_str().ok())
        .filter(|value| !value.is_empty())
        .map(str::to_owned);

    ResolvedRequestContext {
        request_id,
        data_parallel_rank,
        session_id,
    }
}
```

vllm/entrypoints/generate/base/serving.py

Python 侧入口解析核心：_get_session_id 定义了 body > header > vllm_xargs 的行为契约，也是 review 中 tokens-in 路径漏传风险的所在模块。

```
# vllm/entrypoints/generate/base/serving.py
# session_id 的三通道来源与优先级（body > header > vllm_xargs 兼容 fallback）。
# 空字符串与非字符串值一律忽略，避免脏身份进入引擎。
@staticmethod
def _get_session_id_from_headers(raw_request: Request | None) -> str | None:
    # raw_request 为 None 时（例如非 HTTP 调用方）安全返回 None
    if raw_request is None:
        return None
    # 只认 X-Session-ID 头，不把 X-Correlation-ID 等旁路头当作会话身份
    if value := raw_request.headers.get(SESSION_ID_HEADER):
        return value
    return None

@staticmethod
def _get_session_id(
    request: ChatCompletionRequest | CompletionRequest | ResponsesRequest,
    raw_request: Request | None,
) -> str | None:
```

```
    # 1. 请求体里显式声明优先，且非空字符串才生效
    if request.session_id:
        return request.session_id
    # 2. 其次读取 HTTP 头 X-Session-ID
    if value := GenerateBaseServing._get_session_id_from_headers(raw_request):
        return value
    # 3. 最后兜底 vllm_xargs["session_id"]：仅接受非空字符串，
    # 兼容历史用法同时避免 int / bool 等类型污染
    if request.vllm_xargs:
        session_id = request.vllm_xargs.get("session_id")
        if isinstance(session_id, str) and session_id:
            return session_id
    return None
```

tests/entrypoints/openai/test_session_id.py

新增的 97 行测试文件把 Python 侧 session_id 三通道优先级固化为行为契约，覆盖 chat / completion / responses 三类请求，并明确排除 X-Correlation-ID 与非法值。

```
# tests/entrypoints/openai/test_session_id.py
# 该测试文件把 Python 侧 session_id 的三通道优先级固化为行为契约。
def _raw_request(headers: dict[str, str]) -> Request:
    # 手工构造 Starlette Request，模拟真实 HTTP 入口的原始请求
    return Request(
        {
            "type": "http",
            "method": "POST",
            "path": "/v1/chat/completions",
            "headers": [
                (key.lower().encode("latin-1"), value.encode("latin-1"))
                for key, value in headers.items()
            ],
        }
    )


# 请求体字段优先级最高：即使带有 X-Session-ID 头，也应取 body 里的值
@pytest.mark.parametrize(
    "openai_request",
    [
        ChatCompletionRequest(model="test-model", messages=[{"role": "user", "content": "hi"}]),
        CompletionRequest(model="test-model", prompt="hi"),
        ResponsesRequest(model="test-model", input="hi"),
    ],
)
def test_get_session_id_accepts_body_field(openai_request):
    openai_request.session_id = "body-session"
    session_id = GenerateBaseServing._get_session_id(
        openai_request,
        _raw_request({"X-Session-ID": "header-session"}),
    )
```

```
    assert session_id == "body-session"

# vllm_xargs 只是兼容 fallback：X-Correlation-ID 一类的旁路头不应被当作会话身份
def test_get_session_id_ignores_correlation_header():
    request = CompletionRequest(
        model="test-model",
        prompt="hi",
        vllm_xargs={"session_id": "xargs-session"},
    )
    session_id = GenerateBaseServing._get_session_id(
        request,
        _raw_request({"X-Correlation-ID": "correlation-session"}),
    )
    assert session_id == "xargs-session"

# 空字符串与非字符串值都要过滤，保证 None 语义干净
def test_get_session_id_ignores_empty_and_non_string_values():
    request = CompletionRequest(
        model="test-model",
        prompt="hi",
        session_id="",
        vllm_xargs={"session_id": 7},
    )
    session_id = GenerateBaseServing._get_session_id(request, None)
    assert session_id is None
```

评论区精华

评审中有三条关键讨论：

- bongwoobak 指出 ServingTokens（tokens-in 路径）继承 GenerateBaseServing 但 *generate* 调用未传 session_id，而该路径正是 router 与 P/D 部署调用的入口、router 是会话身份的主要生产者，漏传即 " 静默丢标签 "；作者 karen-sy 回复已补齐并新增单测，问题关闭。
- claude[bot] 指出 EngineCoreRequest 的 session_id 字段被插在 resumable 与 external_req_id 之间，而 external_req_id 几乎总是非默认值，会导致其槽位及后续字段偏移，版本错配的 peer（如 Rust engine-core-client 对旧 Python engine core）会静默错解；建议把字段追加到 struct 末尾。作者未公开回复此条，且合并后 Rust 侧测试显示 session_id 在数组末尾（array[20]）而 Python 侧摘要显示其在 resumable 之后，两侧顺序是否最终一致存疑。
- njhill 建议把 session_id 也加到 rust gRPC 接口 rust/proto/inference.proto，作者在最后一个 commit "add wiring into grpc proto" 中补齐，njhill 随后 APPROVED。
- tokens-in 路径漏传 session_id 的静默丢标签风险 (correctness): karen-sy 确认并补齐：serve_tokens 透传 X-Session-ID 头到 engine.generate，并新增 test_serve_tokens_threads_session_id_header_to_engine 单测。

- EngineCoreRequest 字段插入位置与跨语言 wire compatibility (correctness): claude[bot] 建议移动字段位置；作者未公开回复该条。当前材料显示 Python 侧插入位置与 Rust 侧末尾位置可能不一致，需人工确认，否则存在跨语言解码错位风险。
- gRPC 接口补充 session_id (design): karen-sy 在最终 commit "add wiring into grpc proto" 中补齐 gRPC 侧透传，njhill 随后 APPROVED。

风险与影响

- 风险：
  1. 跨语言字段顺序（高风险）：EngineCoreRequest 是 array-like + omit_defaults 的 msgspec Struct，Rust 侧以 serde_tuple 镜像。Python 侧摘要显示 session_id 插入在 resumable 之后，而 Rust 侧测试断言 session_id 位于数组末尾（array[20]）；若两侧最终顺序不一致，Rust engine-core-client 与 Python engine core 之间的 msgpack 数组会错位，且由于 omit_defaults / serde(default) 的存在通常是静默错解而非报错。
  2. 升级耦合：EngineCoreRequest 数组长度 +1（20 → 21），Python 与 Rust、新老版本必须同步升级，否则 wire 错位；这属于引擎核心协议变更。
  3. 契约面扩大：三通道优先级（body > header > vllm_xargs）是新 API 契约，本 PR 未同步文档；且 vllm_xargs 中的 session_id 仍会残留在 extra_args 中，未来引擎若读取该键可能看到两份身份。
  4. 测试盲区：session_id 的跨语言验证仅靠 Rust 侧单测断言数组长度，缺少 Rust engine-core-client 到 Python engine core 的端到端 wire 测试。 - 影响：用户 / API：OpenAI 兼容的 chat、completion、responses 接口新增可选 session_id body 字段，并支持 X-Session-ID 头；不传的请求完全无感知（透传 None），传了会在引擎内部 Request 上可用。系统：V1 引擎核心协议（EngineCoreRequest msgpack 数组）长度变化，要求 Python / Rust 双栈同步升级，存在版本兼容约束。团队：为 #48049 系列的后续会话级策略（调度、KV / cache、connector）铺路；新增的 test_session_id.py 成为 session 语义的行为契约，后续消费方开发时可直接参照。 - 风险标记：跨语言协议字段顺序风险，引擎核心协议数组长度变更，多入口透传易遗漏，缺少跨语言端到端测试

关联脉络

- PR #50746 [Bugfix][Frontend] Reject empty gRPC stop strings: 同属 rust 请求解码链路（rust/src/text、rust gRPC 层）的前端加固，与 session_id 透传共享同一请求解析路径，反映该层正在系统化收紧契约。