PR #37222 完整报告

sgl-project/sglang

[Rust] Keep sampling and scheduler wire schemas in sync

合并时间：2026-09-01 03:31

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/37222

执行摘要

- 一句话：Rust 采样与线上结构对齐 Python，新增双语言 lockstep 测试
- 推荐动作：值得精读。核心值得学习的是 lockstep 测试模式：用最少的成本把一个位置型协议约束在 Python 单元测试里，适合推广到其他 Rust/Python 双实现结构。需要注意该方案依赖源码文本与相对路径，长期建议改由构建期生成 schema 或共享常量来消除脆弱性。

功能与动机

PR body 明确指出：Close Rust/Python sampling parity gaps and make positional wire-schema drift fail in Python unit tests。这是 stack #37223 中的第 3 个 PR。Rust 服务器正在逐步复刻 Python 侧 SamplingParams 的 post_init→ normalize → verify 语义；若两边在空 grammar、structural_tag、custom_params 类型或字段顺序上不一致，同一个 HTTP 请求在 Rust 与 Python 服务器下会产生不同的 400 响应或不同的调度行为。

实现拆解

实现按 5 步展开，核心都在 rust/sglang-server/src/message/sampling.rs，配套测试落在两个 Python 单元测试文件：

1. 空 grammar 归一化：在 SamplingParams::post_init() 末尾新增循环，把 json_schema、regex、ebnf、structural_tag 四个字段的空字符串全部置为 None，与 Python __post_init__ 行为一致，避免空字符串被下游 is_not_none 误判为已设置的约束，从而触发多余的互斥校验或走入错误分支。
2. verify() 校验顺序与互斥规则对齐：把 beam_width < 1 的检查从函数尾部提前到温度校验之前，与 Python verify 的错误报告优先级一致；同时把 structural_tag 纳入 grammar 互斥计数，错误消息更新为 Only one of json_schema, regex, ebnf, or structural_tag can be set。
3. custom_params 类型收紧：字段类型从 Option<serde_json::Value> 改为 Option<BTreeMap<String, CustomParamValue>>，并新增 CustomParamValue 与 JsonScalar 两个 #[serde(untagged)] 枚举，只允许标量、标量列表、字符串键对象，拒绝嵌套对象与嵌套列表，匹配 Python 侧 CustomParamValue 的形状。
4. Rust 回归测试：在 sampling.rs 的测试模块新增 3 个用例，分别覆盖空 grammar 归一化、structural_tag 互斥、custom_params 接受 / 拒绝边界。

5. Python lockstep 测试：test_sampling_params.py 新增 test_rust_sampling_schema_stays_in_lockstep，用正则从 Rust 源码提取 pub struct SamplingParams 的字段列表，与 Python msgspec 的 __struct_fields__ 做全等比较；test_io_struct.py 新增 test_rust_tokenized_generate_schema_stays_in_lockstep，提取 TokenizedGenerateReqInput<'a> 字段并断言其必须是 Python 字段的前缀，且 Python 侧被 Rust 省略的字段都带默认值。两个文件同步补充了 re 与 pathlib.Path 导入。

关键文件：

- rust/sglang-server/src/message/sampling.rs（模块 采样参数；类别 source；类型 core-logic；符号 empty_grammar_constraints_are_unset, structural_tag_is_mutually_exclusive_with_other_grammars, custom_params_matches_python_shape）：核心源码文件：归一化空 grammar、把 structural_tag 纳入互斥校验、提前 beam_width 校验，并将 custom_params 从任意 JSON 收紧为与 Python 形状一致的有界类型，同时新增 3 个 Rust 回归测试。
- test/registered/unit/managers/test_io_struct.py（模块 线上结构；类别 test；类型 test-coverage；符号 test_rust_tokenized_generate_schema_stays_in_lockstep）：新增 TokenizedGenerateReqInput 的跨语言 lockstep 测试：解析 Rust io_struct.rs 中的字段，断言 Rust 字段是 Python msgspec 结构的前缀，且被省略的 Python 尾部字段都必须带默认值。
- test/registered/unit/sampling/test_sampling_params.py（模块 采样参数；类别 test；类型 test-coverage；符号 test_rust_sampling_schema_stays_in_lockstep）：新增 SamplingParams 的跨语言 lockstep 测试：从 Rust 源码提取 pub struct SamplingParams 的全部字段，与 Python msgspec 的 __struct_fields__做全等比较，确保字段集合与顺序完全一致。

关键符号：post_init, verify, empty_grammar_constraints_are_unset, structural_tag_is_mutually_exclusive_with_other_grammars, custom_params_matches_python_shape, test_rust_sampling_schema_stays_in_lockstep, test_rust_tokenized_generate_schema_stays_in_lockstep

关键源码片段

rust/sglang-server/src/message/sampling.rs

核心源码文件：归一化空 grammar、把 structural_tag 纳入互斥校验、提前 beam_width 校验，并将 custom_params 从任意 JSON 收紧为与 Python 形状一致的有界类型，同时新增 3 个 Rust 回归测试。

```
// ============================================================
// CustomParamValue：与 Python srt/sampling/sampling_params.py 中的
// `CustomParamValue` 一一对应，只接受标量、标量列表、字符串键对象，
// 拒绝嵌套对象与嵌套列表，保证两边对同一请求给出相同解析结果。
// ============================================================
#[derive(Debug, Clone, PartialEq, Serialize, Deserialize)]
#[serde(untagged)]
pub enum CustomParamValue {
    Null(()),
```

```
    Bool(bool),
    Signed(i64),
    Unsigned(u64),
    Float(f64),
    String(String),
    List(Vec<JsonScalar>),
    Object(BTreeMap<String, JsonScalar>),
}

/// `List` 与 `Object` 变体内部使用的标量类型。
#[derive(Debug, Clone, PartialEq, Serialize, Deserialize)]
#[serde(untagged)]
pub enum JsonScalar {
    Null(()),
    Bool(bool),
    Signed(i64),
    Unsigned(u64),
    Float(f64),
    String(String),
}
```

```
    fn post_init(&mut self) {
        // Python `__post_init__` 的 guard：已 normalized 的请求直接返回，
        // 否则第二次 normalize 会把 stop 别名清掉，导致停止词全部失效。
        if self.is_normalized {
            return;
        }
        // …… stop / stop_token_ids / temperature / top_k 的处理与 base 一致 ……

        // 与 Python 对齐：空字符串形式的 grammar 等价于“未设置”。
        // 若保留空字符串，下游 `is_not_none` 会误判约束存在，从而
        // 触发多余的互斥校验或在调度器里走错分支。
        for constraint in [
            &mut self.json_schema,
            &mut self.regex,
            &mut self.ebnf,
            &mut self.structural_tag,
        ] {
            if constraint.as_deref() == Some("") {
                *constraint = None;
            }
        }
    }

    fn verify(&self, vocab_size: u64) -> Result<(), Error> {
        // beam_width 校验前移到开头：Python `verify` 先报 beam_width 再
        // 报 temperature，Rust 侧保持同序，错误消息优先级才能一致。
        if let Some(beam_width) = self.beam_width
            && beam_width < 1
```

```
        {
            return Err(bad(format!(
                "beam_width must be at least 1, got {beam_width}."
            )));
        }
        // …… temperature / top_p / min_p / top_k / penalty 等校验与 base 一致 ……

        // Grammars 互斥：structural_tag 也参与计数，与 Python 一致；
        // 任何两个 grammar 同时设置都会得到 400。
        let grammars = [
            &self.json_schema,
            &self.regex,
            &self.ebnf,
            &self.structural_tag,
        ]
        .iter()
        .filter(|g| g.is_some())
        .count();
        if grammars > 1 {
            return Err(bad(
                "Only one of json_schema, regex, ebnf, or structural_tag can be set".into(),
            ));
        }
    }
```

test/registered/unit/managers/test_io_struct.py

新增 TokenizedGenerateReqInput 的跨语言 lockstep 测试：解析 Rust io_struct.rs 中的字段，断言 Rust 字段是 Python msgspec 结构的前缀，且被省略的 Python 尾部字段都必须带默认值。

```
    def test_rust_tokenized_generate_schema_stays_in_lockstep(self):
        """Rust 与 Python 的调度器输入结构必须保持字段前缀一致。"""
        rust_path = (
            Path(__file__).resolve().parents[4]
            / "rust/sglang-server/src/message/io_struct.rs"
        )
        source = rust_path.read_text()
        start = source.index("pub(super) TokenizedGenerateReqInput<'a> {")
        end = source.index("\n    }\n}", start)
        # 前两个字段是 Rust 侧固定前缀，后面由正则从源码中提取。
        rust_fields = (
            "rid",
            "http_worker_ipc",
            *re.findall(r"^\s*([a-z][a-z0-9_]*):", source[start:end], re.MULTILINE),
        )
        python_fields = TokenizedGenerateReqInput.__struct_fields__

        # Rust 字段必须等于 Python 字段的前缀（Rust 可省略 Python 尾部
        # 带默认值的字段），这样调度器按位置读取时不会错位。
```

```
        self.assertEqual(python_fields[: len(rust_fields)], rust_fields)
        self.assertTrue(
            all(
                default is not msgspec.NODEFAULT
                for default in TokenizedGenerateReqInput.__struct_defaults__[
                    len(rust_fields) :
                ]
            ),
            "Rust may omit only a defaulted suffix of the Python wire schema",
        )
```

test/registered/unit/sampling/test_sampling_params.py

新增 SamplingParams 的跨语言 lockstep 测试：从 Rust 源码提取 pub struct SamplingParams 的全部字段，与 Python msgspec 的 __struct_fields__做全等比较，确保字段集合与顺序完全一致。

```
    def test_rust_sampling_schema_stays_in_lockstep(self):
        """Rust 字段必须与 Python msgspec 的 SamplingParams 字段完全一致。"""
        rust_path = (
            Path(__file__).resolve().parents[4]
            / "rust/sglang-server/src/message/sampling.rs"
        )
        source = rust_path.read_text()
        start = source.index("pub struct SamplingParams {")
        end = source.index("\n}

/// The `/generate`", start)
        rust_fields = tuple(
            re.findall(
                r"^\s*pub ([a-z][a-z0-9_]*):",
                source[start:end],
                re.MULTILINE,
            )
        )
        # 位置型 wire schema：字段顺序本身就是协议的一部分。
        # 任何一侧增删字段或调整顺序都会让该断言直接失败。
        self.assertEqual(SamplingParams.__struct_fields__, rust_fields)
```

评论区精华

本 PR 没有实质性的 review 讨论：merrymercy 两次在 review 中评论 approve，rainj-me 最终 APPROVED。Issue 评论主要是 CI 重跑记录，最初多组目标测试失败，经过三轮 /rerun-test 后全部通过。github-actions[bot] 的典型记录为：Results for /rerun-test test/registered/rust/test_run_rust_tests.py test/registered/core/test_srt_endpoint.py test/registered/vlm/test_rust_native_mm_e2e.py test/registered/vlm/test_rust_native_mm_mmmu.py: 前两轮 ❓，最终 ❓。未发现与本次 parity 改动直接相关的回归。

- CI rerun 与测试稳定性 (testing): 最终所有目标测试全绿，未发现与本次 parity 改动直接相关的回归；失败原因在上下文中未标注。
- 合并前自审与 approval (other): 无实质性争议，直接合并。

风险与影响

- 风险：
  1. custom_params 类型收紧：Rust 侧不再接受任意 JSON，嵌套 list 与嵌套 object 会解析失败并返回 400；若此前有调用方依赖旧行为发送嵌套结构，Rust server 会新增拒绝路径，但该拒绝与 Python 类型边界一致。
  2. 空 grammar 归一化语义变化：空字符串从已设置约束变为未设置，改变请求语义，需确保与 Python 完全一致后再合并。
  3. verify 校验顺序变化：beam_width 与 temperature 同时非法时返回的错误消息改变，可能影响依赖错误文本的客户端。
  4. lockstep 测试脆弱性：测试依赖源码字符串标记（如 pub struct SamplingParams {）与相对路径 Path(__file__).resolve().parents[4]，Rust 大重构或测试文件移动会让测试直接失败或解析到错误位置。
  5. 影响面相对有限：改动只落在新增的 Rust server 路径，Python 主路径未被触碰。 - 影响：影响范围集中在 rust/sglang-server 的采样参数与调度器线上输结构。对用户而言，Rust server 处理空 grammar、嵌套 custom_params、structural_tag 与其他 grammar 并存时，行为会与 Python 对齐。对团队而言，lockstep 测试成为维护双实现的第一道防线，字段位置漂移会在 Python 单元测试阶段立即失败，而不是到端到端测试才暴露。CI 影响：两个测试文件均已注册到 CPU/GPU/XPU 网格，新增测试为纯文件解析与字符串比较，没有额外模型加载成本。 - 风险标记：custom_params 类型收紧可能拒绝旧请求，空 grammar 语义归一化行为变化，错误消息优先级随校验顺序改变，lockstep 测试依赖 Rust 源码文本与相对路径，影响面集中在 Rust 服务器路径

关联脉络

- PR #37226 [Rust] Simplify request defaults and document batch header ABI: 同一 rust-server-cleanup stack 的第 4/4 个 PR，继续精简 Rust 请求结构与文档化 batch header ABI，与本 PR 共享 rust/sglang-server 的线上结构代码路径。