PR #33558 完整报告

sgl-project/sglang

[Laguna] fix YaRN mscale double-application in rope config

合并时间：2026-08-08 07:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33558

执行摘要

- 一句话：修复 Laguna YaRN mscale 双重应用导致的精度退化
- 推荐动作：建议精读。该 PR 是“框架语义适配”的微型范本：核心设计决策包括（1）用除法把 HF 的“最终值”语义还原为 SGLang 的“乘数”语义；（2）通过 pop 掉 mscale/mscale_all_dim 让 yarn.py 走简单默认 base，消除除法基准不确定性；（3）契约测试基于真实 get_rope 而非手抄公式，并专门构造 (32.0, 1.1) 反例以区分错误实现。对维护多模型配置适配层的团队尤其有参考价值。

功能与动机

PR body 明确指出：HF 的 attention_factor 是最终 YaRN mscale，但 SGLang 的 YaRN 把 attn_factor 当作对自身 yarn_get_mscale_simple(factor) 的乘数，直接复制会平方 scale（factor 128: 1.4852 -> 2.2058，+48.5%），破坏全局注意力层。作者报告修复后 GSM8K 提升约 20 分，与其他框架对齐。

实现拆解

1. 确认语义差异：SGLang 的 YaRNScalingRotaryEmbedding（python/sglang/srt/layers/rotary_embedding/yarn.py）把配置的 attn_factor 乘到 yarn_get_mscale_simple(factor) 上得到最终 mscale；HF 的 attention_factor 本身已经是最终值，直接透传等于把最终值再乘一次默认值，形成平方。
2. 修改配置转换函数：python/sglang/srt/configs/laguna.py 的 _to_sglang_rope_scaling 在 attention_factor 分支中新增三步处理：先 pop 掉 mscale 与 mscale_all_dim，避免 yarn.py 用两者比值作为 base 导致除法基准不一致；再将 factor 转 float 并兼容空值；最后计算 attn_factor = attention_factor / yarn_get_mscale_simple(factor)。yarn 工具函数在函数内部 import，规避模块级循环依赖。
3. 新增契约测试：test/registered/unit/configs/test_laguna_config.py 的 TestLagunaRopeScaling 不再手抄 yarn.py 公式，而是通过 get_rope 构造真实 rotary embedding，断言其 mscale 等于 HF attention_factor；用例覆盖 Laguna S（factor 128）、XS（factor 32）以及 (32.0, 1.1) 非默认反例，并验证 rope_type 为 default 或空 dict 时返回 None。
4. CI 兼容处理：RotaryEmbedding.__init__在非 CUDA 且非 CPU 引擎的主机会硬导入 vllm，测试用 mock.patch 将 rotary_embedding.base._is_cpu 置 True 伪装 CPU 引擎，使测试可在无 GPU CI 运行；通过 register_cpu_ci(est_time=5, suite="base-a-test-cpu") 注册

进 CPU 套件常驻执行。

5. 验证：作者报告 GSM8K 提升约 20 分，与 vLLM 等框架对齐；改动仅发生在配置转换期，无速度回归报告。

关键文件：

- python/sglang/srt/configs/laguna.py（模块 模型配置；类别 source；类型 core-logic；符号 _to_sglang_rope_scaling）：修复核心所在：_to_sglang_rope_scaling 将 HF 的 attention_factor（最终 mscale）转换为 SGLang 的 attn_factor（乘数），并抑制 mscale/mscale_all_dim 透传，避免 YaRN scale 被平方。
- test/registered/unit/configs/test_laguna_config.py（模块 模型配置；类别 test；类型 test-coverage；符号 TestLagunaRopeScaling, setUpClass, _composed_mscale, test_s_mscale_matches_hf）：新增契约测试：通过 get_rope 构造真实 embedding 断言 mscale 与 HF attention_factor 对齐，并用 (32.0, 1.1) 反例挡住错误实现；注册 CPU CI 常驻运行。

关键符号：_to_sglang_rope_scaling, TestLagunaRopeScaling.setUpClass, TestLagunaRopeScaling._composed_mscale, TestLagunaRopeScaling.test_s_mscale_matches_hf, TestLagunaRopeScaling.test_xs_mscale_matches_hf, TestLagunaRopeScaling.test_non_default_attention_factor, TestLagunaRopeScaling.test_default_and_empty_rope_stay_plain

关键源码片段

python/sglang/srt/configs/laguna.py

修复核心所在：_to_sglang_rope_scaling 将 HF 的 attention_factor（最终 mscale）转换为 SGLang 的 attn_factor（乘数），并抑制 mscale/mscale_all_dim 透传，避免 YaRN scale 被平方。

```
def _to_sglang_rope_scaling(rope_params: Dict[str, Any]) -> Optional[Dict[str, Any]]:
    """把 HF 的 per-layer rope dict 转成 SGLang get_rope 的 rope_scaling，None 表示普通
    RoPE。"""
    if not rope_params:
        return None
    rope_type = rope_params.get("rope_type") or rope_params.get("type")
    if rope_type in (None, "default"):
        return None

    out: Dict[str, Any] = {"rope_type": rope_type}
    pass_through = (
        "factor",
        "original_max_position_embeddings",
        "beta_fast",
        "beta_slow",
        "extrapolation_factor",
        "truncate",
        "low_freq_factor",
        "high_freq_factor",
```

```
            "mscale",
            "mscale_all_dim",
            "short_factor",
            "long_factor",
            "short_mscale",
            "long_mscale",
        )
        for key in pass_through:
            if key in rope_params:
                out[key] = rope_params[key]

        if "attention_factor" in rope_params:
            # HF 的 attention_factor 是最终 YaRN mscale，而 SGLang 的 YaRN 会把
            # attn_factor 乘到自己的 yarn_get_mscale_simple(factor) 上。直接透传
            # 会把 scale 平方（factor 128 时 1.4852 -> 2.2058），所以这里除回默认值。
            # 同时需要删掉 mscale/mscale_all_dim：yarn.py 在两者都存在时会用它们的
            # 比值作为 base，只有去掉才能保证以简单默认值为基数做除法。
            from sglang.srt.layers.rotary_embedding.yarn import yarn_get_mscale_simple

            out.pop("mscale", None)
            out.pop("mscale_all_dim", None)
            factor = float(rope_params.get("factor", 1.0) or 1.0)
            out["attn_factor"] = rope_params["attention_factor"] / yarn_get_mscale_simple(
                factor
            )
        return out
```

test/registered/unit/configs/test_laguna_config.py

新增契约测试：通过 get_rope 构造真实 embedding 断言 mscale 与 HF attention_factor 对齐，并用 (32.0, 1.1) 反例挡住错误实现；注册 CPU CI 常驻运行。

```
class TestLagunaRopeScaling(CustomTestCase):
    @classmethod
    def setUpClass(cls):
        # RotaryEmbedding.__init__ 会读取 get_server_args()，先发布测试用配置。
        publish(ServerArgs(model_path="dummy"), role="test")

    def _composed_mscale(self, factor, attention_factor):
        """服务端实际生效的 mscale：走完整配置转换并构造真实 embedding。"""
        rope_scaling = _to_sglang_rope_scaling(
            {
                "rope_type": "yarn",
                "factor": factor,
                "original_max_position_embeddings": 8192,
                "attention_factor": attention_factor,
            }
        )
        # 伪装 CPU 引擎：RotaryEmbedding.__init__ 在无 GPU 且非 CPU 引擎的主机会
        # 硬导入 vllm，CPU CI runner 上会 ModuleNotFoundError；mscale 在选核前
```

```
        # 就已算出，不影响本测试的断言目标。
        with mock.patch("sglang.srt.layers.rotary_embedding.base._is_cpu", True):
            emb = get_rope(
                128,
                rotary_dim=128,
                max_position=262144,
                base=500000,
                rope_scaling=rope_scaling,
                partial_rotary_factor=0.5,
                dtype=torch.float32,
            )
        return float(emb.mscale)

    def test_non_default_attention_factor(self):
        # 反例：attention_factor 既不是 1.0 也不是 yarn 默认值，能同时拦住
        # “硬编码 attn_factor=1.0”与“直接透传导致平方”两类错误实现。
        factor, af = _NON_DEFAULT # (32.0, 1.1)
        self.assertAlmostEqual(self._composed_mscale(factor, af), af, places=6)
```

评论区精华

审阅者 Jiminator 的几轮意见直接塑造了最终实现与测试：

> “yarn.py:96-105 picks the mscale/mscale_all_dim ratio as its base when both are set.” —— 除法基准不一定是实际 base，需要 pop 掉 mscale/mscale_all_dim，最终实现采纳。
>
> “Currently the attention_factor always equals the yarn default so the conversion always produces 1.0 currently in the test... out['attn_factor'] = 1.0 passes, which isnt ideal.” —— 指出测试用例盲区，要求补充 (32.0, 1.1) 非默认值，最终采纳。
>
> “This helper hardcodes its own copy of the yarn.py composition rule instead of exercising yarn.py itself, so it stays green even if the composition changes.” —— 契约测试必须基于真实组件构造而非手抄公式，最终重写为 get_rope + 断言 mscale。

另有两条 nit（删除冗余断言方法、导入提升到模块级）均已处理。

- 除法基准应与 yarn.py 实际 base 一致 (correctness): 通过 pop 掉 mscale 与 mscale_all_dim 保证以简单默认值为 base，最终实现采纳。
- 测试需覆盖 attention_factor 非默认值 (testing): 新增 (32.0, 1.1) 非默认反例，最终采纳。
- 契约测试应基于真实 embedding 而非手抄组合规则 (testing): 重写为 _composed_mscale 通过 get_rope 构造真实 embedding 并断言 mscale，最终采纳。
- 删除冗余断言方法 (style): 冗余方法已移除。
- 导入位置调整 (style): 测试中的模块级 import 已调整。

风险与影响

- 风险：

1. 模型输出数值变更：修复会改变 Laguna S/XS 等模型的实际注意力缩放，属预期修正，但部署方需重新验证线上效果；改动仅作用于携带 attention_factor 的 rope 配置，其他模型不受影响。
2. 配置兼容性约束：pop 掉 mscale/mscale_all_dim 意味着未来若有人同时配置 mscale 与 attention_factor，将得到与注释一致的新语义（以简单默认值为 base），新增模型 config 时需注意这一隐含约束。
3. 测试稳健性：_composed_mscale 依赖 mock _is_cpu 与 get_rope 内部初始化路径，rotary_embedding 未来重构时需要同步更新该测试，否则可能出现误报或漏报。
4. CI 状态：材料显示 PR Test (Extra) 曾有失败记录（Run #31143061375）且无明细，合入前已通过 /tag-and-rerun-ci 重跑，但仍建议关注 extra 套件后续稳定性。 - 影响：影响范围集中在 Laguna 模型系列的 RoPE 配置转换：修复后服务端 mscale 与 HF/vLLM 对齐，纠正全局注意力层的缩放偏差（factor 128 时由 2.2058 回到 1.4852），GSM8K 提升约 20 分；对系统而言这是一次启动期配置计算变更，无运行时开销变化；对团队的价值是新增了可常驻 CPU CI 的契约测试，今后 yarn.py 的组合规则若有变化会立即被测试捕获，避免同类漂移再次发生。 - 风险标记：模型输出数值变更，配置转换核心路径，测试依赖 mock 环境标志，PR test extra 曾有失败

关联脉络

- PR #33925 config: route DCP topology reads through get_parallel(): 同属“配置语义与运行时实际行为对齐”的一致性主题，都强调消除配置解释与运行时行为之间的偏差。
- PR #33417 Fix deterministic inference for Inkling: 同属模型输出数值一致性修复线，且均配套新增针对性回归测试，反映仓库对跨框架语义对齐的持续投入。