PR #27742 完整报告

sgl-project/sglang

test(sgl-router): cover sticky scale-up no-redistribution e2e

合并时间：2026-06-10 23:54

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27742

执行摘要

该 PR 为 sgl-router 的 sticky-session 路由策略添加了一个端到端集成测试，验证运行时新增 worker 不会导致已固定的路由键重新分配。这是 sticky 策略区别于一致性哈希的核心属性。所有变更仅涉及测试代码，生产代码无任何修改。

功能与动机

Sticky-session 策略的 "true-sticky" 属性要求在添加 worker 时，已固定的路由键保持不变（与一致性哈希不同，后者会重新分布部分键）。这一属性在单元测试层面已有验证（sticky.rs 中的 adding_a_worker_does_not_redistribute_existing_key），但缺少端到端的集成测试。已有的集成测试覆盖了同键固定、移除时重映射、无键回退和动态头名称，但未覆盖运行时 scale-up。如果 registry-add → candidate-set → policy 路径中出现回归，单元测试无法捕获。该 PR 填补了这一空白。

实现拆解

1. 新增测试函数：在 experimental/sgl-router/tests/proxy/sticky_routing.rs 中新增 adding_a_worker_does_not_redistribute_existing_key 测试。
2. 测试步骤：
   - 启动两个 MockWorker，构建 sticky 路由上下文和路由器。
   - 固定键 "alice" 到初始选中的 worker，记录该 worker URL。
   - 通过 WorkerRegistry::add 动态添加第三个 MockWorker，并断言该 worker 成为健康候选者（healthy_workers_for 返回 3）。
   - 连续发送 5 次相同键的请求，验证所有请求均路由到最初固定的 worker。
   - 断言 sticky 指标：assigned=1，remap=0，hit=5。
3. 确定性保证：测试利用 MockWorker 和同步操作，无需睡眠或定时器，保证稳定性和可重复性。

experimental/sgl-router/tests/proxy/sticky_routing.rs

这是唯一的变更文件，新增了一个集成测试 adding_a_worker_does_not_redistribute_existing_key，覆盖 sticky-session 策略在运行时 worker 扩容时不重新分配已有路由键的核心属性。

```
/// True-sticky scale-up: a worker that joins the registry at runtime must
/// NOT redistribute an already-pinned key — the defining difference from
/// consistent hashing, where adding a node remaps a fraction of keys. The
```

```
/// policy unit tests assert this over a bare worker slice; this drives it
/// end-to-end through the HTTP stack and a live `WorkerRegistry::add`, so
/// the freshly-added worker is a genuine healthy candidate the policy could
/// pick — and provably doesn't.
#[tokio::test]
async fn adding_a_worker_does_not_redistribute_existing_key() {
    // Arrange: start 2 workers with sticky policy
    let w0 = MockWorker::start(vec![]).await;
    let w1 = MockWorker::start(vec![]).await;
    let ctx = build_sticky_ctx("x-sgl-routing-key", &[w0.url.clone(), w1.url.clone()]);
    let app = build_router(ctx.clone());

    // Act: pin key "alice" to whichever worker is initially selected
    let res = app
        .clone()
        .oneshot(chat_request(Some(("x-sgl-routing-key", "alice"))))
        .await
        .unwrap();
    assert_eq!(res.status(), StatusCode::OK);
    let pinned_url = success_counts(&ctx.metrics.render())
        .into_iter()
        .find(|(_, c)| *c > 0)
        .map(|(url, _)| url)
        .expect("first request should have been served by some worker");

    // Scale up: add a 3rd worker at runtime via WorkerRegistry::add (full HTTP stack)
    let w2 = MockWorker::start(vec![]).await;
    ctx.registry
        .add(WorkerSpec {
            id: WorkerId("w2".into()),
            url: w2.url.clone(),
            mode: WorkerMode::Plain,
            model_ids: vec![ModelId("tiny".into())],
            bootstrap_port: None,
        })
        .unwrap();

    // Guard: ensure w2 is actually eligible (premise check)
    assert_eq!(
        ctx.registry
            .healthy_workers_for(&ModelId("tiny".into()))
            .len(),
        3,
        "added worker must be an eligible candidate"
    );

    // Same-key requests after scale-up; all must stay on original pin
    const N: usize = 5;
    for _ in 0..N {
```

```
        let res = app
            .clone()
            .oneshot(chat_request(Some(("x-sgl-routing-key", "alice"))))
            .await
            .unwrap();
        assert_eq!(res.status(), StatusCode::OK);
    }

    // Assert metrics: one assignment, zero remaps, N hits
    let metrics = ctx.metrics.render();
    assert_eq!(sticky_count(&metrics, "assigned"), 1, "{metrics}");
    assert_eq!(sticky_count(&metrics, "remap"), 0, "{metrics}");
    assert_eq!(sticky_count(&metrics, "hit"), N as u64, "{metrics}");

    // Assert all requests landed on the original pin worker
    let counts = success_counts(&metrics);
    assert_eq!(
        counts.get(&pinned_url).copied().unwrap_or(0),
        (N + 1) as u64,
        "all same-key requests must stay on the original pin: {counts:?}"
    );
}
```

评论区精华

无 review 评论。

风险与影响

风险：无，仅添加测试，无生产代码变更。

影响：

- 对用户：无直接影响。
- 对系统：CI 中新增一个端到端测试，增强对 sticky-session 策略核心属性的回归检测。
- 对团队：测试前提守卫（如 healthy_workers_for 数量断言）增加了测试的有效性，避免因未来变更导致测试流于形式。

关联脉络

该 PR 与单元测试 sticky.rs 中的 adding_a_worker_does_not_redistribute_existing_key 形成互补，将相同属性的验证从纯策略层面扩展到完整的 HTTP 路由栈。与近期其他 sticky 路由测试 PR（如 PR#27695、27617）同属路由可靠性增强系列。