PR #33269 完整报告

sgl-project/sglang

[rust-server] Reland: fix TCP-layer TTFT stalls (#33026)

合并时间：2026-08-03 11:52

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33269

[rust-server] Reland: 修复 TCP 层 TTFT 停顿（#33269）

执行摘要

本 PR 重放被回滚的 #33026，修复 Rust server 两个 TCP 层 TTFT 停顿：大请求体因内核默认 rcvbuf（约 87 KB）不足导致约 200 ms 停顿，以及 keep-alive 连接上 Nagle + delayed-ACK 带来约 13 ms 额外延迟。与上一版的关键差异是改用 socket2 构建监听 socket，在 bind 前设置 16 MiB SO_RCVBUF，同时保持 bind + listen 全同步，不再依赖 tokio reactor，从而消除 runtime::tests 的启动竞态失败。实测各模型 TTFT 下降 10%～20%，keep-alive 场景与 Python TM 完全对齐（26.4 ms），test_cargo_workspace CI 恢复通过。

功能与动机

PR body 给出了两个基于实测的根因：

1. 大请求体停顿约 200 ms：一次 headers + body 突发超过内核初始 rcvbuf（tcp_rmem[1] = 87380 B）时，尾部数据会在 hyper 轮询 body 之前被内核丢弃，发生在每个连接的第一个大请求上。
2. keep-alive 连接额外约 13 ms：hyper/axum 不设置 TCP_NODELAY（Python 的 asyncio 默认设置），Nagle 算法与 delayed-ACK 互相等待。

原始修复 #33026 因为 tokio::net::TcpSocket::listen() 需要 reactor，被迫把 listen() 从同步的 runtime::start() 移到 async api runtime 上，破坏了「start() 返回后端口立即可连接」的测试语义，导致 CI 与本地 runtime::tests 失败（ECONNREFUSED），被 #33257 回滚。本次重放的目标是在不牺牲同步监听语义的前提下保留两个性能修复。

实现拆解

1. 监听 socket 重建（rust/sglang-server/src/runtime.rs 的 Runtime::start）：用 socket2 替换 std::net::TcpListener::bind 一步式流程，改为 Socket::new → set_reuse_address(true) → set_recv_buffer_size(16 MiB) → bind → listen(1024) → 转回 std listener → set_nonblocking(true)。这样 SO_RCVBUF 能在 bind 前生效，且 bind、listen 全部在 spawn 任何线程之前同步完成。listen backlog 从 std 默认的 128 提高到 1024。
2. 连接级 TCP_NODELAY（rust/sglang-server/src/api_server.rs 的 serve）：在 axum::serve 之前通过 axum::serve::ListenerExt::tap_io 对每个已 accept 的 socket 设置 set_nodelay(true)，失败仅记 debug 日志，不阻断启动。tap_io 在连接进入 tokio 事件

循环前完成设置，避免「先写后改」导致的 TCP 语义差异。

3. 依赖配套（rust/Cargo.toml、rust/sglang-server/Cargo.toml、rust/Cargo.lock）：workspace 统一声明 socket2 = "0.6"，server crate 以 workspace 引用，锁文件同步。没有新增测试文件，验证依赖现有 cargo test --workspace（含此前失败的两个运行时测试）+ cargo fmt + cargo clippy，并附完整 benchmark 对照。

rust/sglang-server/src/runtime.rs

核心改动：用 socket2 重写监听 socket 构建流程，在 spawn 线程前同步完成 bind + listen，并设置 16 MiB SO_RCVBUF 与 1024 backlog。这是解决大请求体 TTFT 停顿的关键，同时也是避免重蹈 #33026 竞态失败的核心设计。

```
/// 启动整个前端。线程 spawn 完成后立即返回（非阻塞），
/// 让 Python 调用方立刻拿回 GIL。配置错误（如缺 tokenizer）返回 Err。
pub fn start(cfg: RuntimeConfig) -> Result<Runtime, String> {
    // 绑定 API server 端口必须早于任何线程 spawn：端口占用（EADDRINUSE）
    // 应当成为硬性启动错误。这里用 socket2 而非 std::net::TcpListener，
    // 因为只有 socket2 能在 listen() 之前设置 SO_RCVBUF。
    let addr = cfg.rust_server_args.http_addr;
    let socket = socket2::Socket::new(
        socket2::Domain::for_address(addr),
        socket2::Type::STREAM,
        Some(socket2::Protocol::TCP),
    )
    .map_err(|e| format!("socket for {addr} failed: {e}"))?;

    // 允许快速重启复用 TIME_WAIT 端口，保持与旧 std TcpListener 行为一致
    socket
        .set_reuse_address(true)
        .map_err(|e| format!("set_reuseaddr failed: {e}"))?;

    // 16 MiB 接收缓冲：内核默认 rcvbuf（tcp_rmem[1] ≈ 87 KB）会在 hyper
    // 轮询请求体之前丢弃大请求的尾部，造成每个连接首个大请求约 200 ms 停顿。
    // 设置在 listener 上会被 accept 出的连接继承；内核按 net.core.rmem_max 钳制，
    // 且内存是惰性分配的，实际开销只在连接真正收发数据时产生。
    if let Err(e) = socket.set_recv_buffer_size(16 * 1024 * 1024) {
        eprintln!("warning: set_recv_buffer_size on listener failed: {e}");
    }

    // bind + listen 同步完成，确保 start() 返回时端口已可接受连接。
    // 这是 runtime::tests 依赖的语义：上一版 #33026 用 tokio TcpSocket::listen
    // 需要 reactor，被迫把 listen 挪进 async runtime，导致测试在 start() 返回后
    // 立即连接时竞态失败（ECONNREFUSED），PR 被回滚。socket2 让调优与同步语义共存。
    socket
        .bind(&addr.into())
        .map_err(|e| format!("bind {addr} failed: {e}"))?;
    socket
        .listen(1024)
        .map_err(|e| format!("listen on {addr} failed: {e}"))?;
```

```
    let listener: std::net::TcpListener = socket.into();
    listener
        .set_nonblocking(true)
        .map_err(|e| format!("listener set_nonblocking failed: {e}"))?;
    // ... 后续创建 ring、channel、spawn 线程 ...
}
```

rust/sglang-server/src/api_server.rs

入口层补上 TCP_NODELAY：通过 axum ListenerExt::tap_io 对已 accept socket 设置 set_nodelay(true)，消除 keep-alive 连接上约 13 ms 的 Nagle/delayed-ACK 延迟，与 Python asyncio 行为对齐。

```
    // 与 Python 对齐：asyncio 默认给 TCP 连接设置 TCP_NODELAY，
    // 而 hyper/axum 不会自动设置。不设的话，keep-alive 连接上会
    // 出现 Nagle 算法与 delayed-ACK 互相等待，带来约 13 ms 额外延迟。
    // ListenerExt::tap_io 钩住每个被 accept 的 socket，在它进入
    // tokio 事件循环之前立即完成设置，行为与 Python 一致。
    use axum::serve::ListenerExt;
    let listener = listener.tap_io(|io| {
        if let Err(e) = io.set_nodelay(true) {
            tracing::debug!(error = %e, "set_nodelay failed");
        }
    });

    // with_connect_info 把对端地址暴露给 access-log 中间件
    let serve = axum::serve(
        listener,
        app.into_make_service_with_connect_info::<std::net::SocketAddr>(),
    );
```

评论区精华

Reviewer rainj-me 提出两条 nic（非阻塞）意见后 APPROVED：

> nic: add a TODO comments to extract 16M to and 1024 backlog to server args
>
> nic: refactor the tcp socket tuning logic to different file

两条建议都针对 runtime.rs 中内联的 socket 调优代码：一是把 16 MiB 缓冲与 1024 backlog 硬编码参数化到 server args，二是把调优逻辑抽取到独立文件。均未在本次落地，属于「通过但留待后续维护」的改进项，说明 reviewer 认可当前内联实现的功能正确性。

风险与影响

风险：

- 内核配置依赖：16 MiB SO_RCVBUF 受 net.core.rmem_max 钳制，低配环境可能静默降级；高 rmem_max + 大量连接时内存压力需监控（PR body 说明为惰性分配，风险有限）。
- backlog 从 128 提升到 1024，在超过 net.core.somaxconn 的连接突发下行为取决于内核配置。

- 无新增测试：同步 listen 语义依赖现有 runtime::tests 回归，但 SO_RCVBUF 与 TCP_NODELAY 本身没有专项测试，性能防回归依赖 benchmark。
- 启动路径用 eprintln! 输出缓冲设置失败告警，库代码中改用 tracing 更合适（小瑕疵）。

影响：所有嵌入 Rust server 的部署（PyO3 模块 sglang.srt.server._core）都会受益。实测：gpt-oss 16K in 的 TTFT 均值 192 → 157 ms（P99 369 → 161），64K in 1253 → 1047 ms；MiMo 64K in 2197 → 1957 ms，256K in 9716 → 9513 ms；keep-alive 128 连发 40.9 → 26.4 ms，与 Python TM（26.4 ms）完全对齐；裸 socket 探针 208 → 0.4 ms。解码路径不受影响（c=256 下 ITL 与 Python 持平）。

关联脉络

本 PR 是 #33026 的第三次形态：原修复 → #33257 回滚 → 本次 socket2 方案重放，完整呈现了一次「性能修复引入启动竞态 → 回滚 → 换底层 API 绕开约束」的工程迭代。它属于 Rust server 与 Python TM 性能对齐的持续工作线，同一方向还包括 #33125（PD 分离部署）、#33105（DP attention 客户端负载均衡），说明团队正逐步将 Rust server 推向生产可用的性能基线。后续值得关注 reviewer 提议的 socket 参数可配置化，以及是否将本次性能基准纳入常规 CI benchmark 以防回归。