PR #19524 完整报告

sgl-project/sglang

feature: upstream cancel

合并时间：2026-05-18 16:20

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/19524

执行摘要

本次 PR 为 sgl-model-gateway 添加了客户端断连时自动取消上游 streaming 请求的能力。核心是通过新增 BreakerTrackedStream 适配器来安全地终止流并更新断路器状态，同时在路由器中使用 tokio::select! 检测客户端断开。该变更消除了 GPU 计算浪费，且经过 A/B 测试确认无性能回归。建议仔细阅读其断路器逻辑和 BreakerTrackedStream 的设计。

功能与动机

PR 正文指出："When a client disconnects mid-stream, the router now detects the closed connection via tx.closed() in a tokio::select! loop and drops the upstream HTTP connection immediately. This signals the engine to abort generation, freeing GPU compute that would be wasted otherwise." 此前在客户端断开后，网关不会主动取消上游请求，导致引擎继续生成 token，浪费 GPU 资源。该 PR 解决了这一痛点，提升集群资源利用率。

实现拆解

1. 核心适配器：新增 BreakerTrackedStream（streaming_utils.rs），包装上游 stream，维护 Terminal 三态，在 Drop 时根据终止状态调用 record_success 或 record_failure，从而分离 HTTP 响应状态与实际流传输结果。
2. 路由器修改：在 HTTP、PD、OpenAI 四个路由器的 streaming 转发路径中，使用 tokio::select! 良性竞争 stream.next() 与 tx.closed()，客户端断开时立即跳出循环并丢弃上游 body，reqwest 连接随之关闭，引擎收到断开信号。
3. 断路器记录时机调整：对于 streaming 请求，不在初始 200 响应时记录成功，而是由 BreakerTrackedStream 在流结束时决定；同时修复了请求发送失败时忘记记录失败的问题。
4. 测试与基准：扩展 mock_worker.rs 支持慢速流和错误注入；新增 2700+ 行集成测试覆盖取消、完成、错误等场景；新增微基准测试量化开销。
5. 配置与工具链：更新 Cargo.toml 依赖，添加 pre-commit rustfmt 钩子。

sgl-model-gateway/src/routers/streaming_utils.rs

核心新增文件，定义 BreakerTrackedStream 结构，封装 upstream stream 的终止状态，并在 Drop 时安全地决定断路器状态。是所有路由器客户端断开取消的基础。

```
/// 三态状态机，表示流终止的方式。
enum Terminal {
    Active, // 仍在飞行或丢弃前未终止
    Completed, // 流正常结束或调用方标记完成
```

```
    Errored, // 流产生错误
}

/// 包装一个 `Stream<Item = Result<Bytes, E>>`，使得断路器只在 Drop 时更新。
pub struct BreakerTrackedStream<E = reqwest::Error> {
    inner: Pin<Box<dyn Stream<Item = Result<Bytes, E>> + Send + 'static>>,
    worker: Arc<dyn Worker>,
    log_url: String,
    terminal: Terminal,
}

impl<E> BreakerTrackedStream<E> {
    pub fn new<S>(inner: S, worker: Arc<dyn Worker>, log_url: String) -> Self {
        Self {
            inner: Box::pin(inner),
            worker,
            log_url,
            terminal: Terminal::Active,
        }
    }

    /// 标记流为正常完成。用于路由器通过带内哨兵（如 `data: [DONE]`）提前发现流结束。
    pub fn mark_completed(&mut self) {
        if self.terminal == Terminal::Active {
            self.terminal = Terminal::Completed;
        }
    }

    /// 预标记流为错误。用于包装已包含失败结果的内容（如非 2xx 响应体）。
    pub fn mark_errored(&mut self) {
        self.terminal = Terminal::Errored;
    }
}

impl<E: Display> Stream for BreakerTrackedStream<E> {
    type Item = Result<Bytes, E>;
    fn poll_next(mut self: Pin<&mut Self>, cx: &mut Context<'_>) -> Poll<Option<Self::Item>> {
        match self.inner.as_mut().poll_next(cx) {
            Poll::Ready(Some(Ok(b))) => Poll::Ready(Some(Ok(b))),
            Poll::Ready(Some(Err(e))) => {
                tracing::error!("Upstream stream error from worker {}: {}", self.log_url, e);
                self.terminal = Terminal::Errored;
                Poll::Ready(Some(Err(e)))
            }
            Poll::Ready(None) => {
                if self.terminal == Terminal::Active {
                    self.terminal = Terminal::Completed;
                }
                Poll::Ready(None)
```

```
            }
            Poll::Pending => Poll::Pending,
        }
    }
}

impl<E> Drop for BreakerTrackedStream<E> {
    fn drop(&mut self) {
        match self.terminal {
            Terminal::Completed => self.worker.record_success(),
            Terminal::Errored => self.worker.record_failure(),
            Terminal::Active => { /* client disconnected, do nothing */ }
        }
    }
}
```

评论区精华

> " 每个 SSE 块通过 format! 拼接 `

会额外分配String，建议优化。" — alexnails 提出的性能问题已通过sse_block_to_bytes` 解决。

> "double accounting here?" — alexnails 对断路器计数提出的质疑，通过调整记录时机已解决。
>
> "There is extra task here we do not need, clean up?" — 关于额外 spawn 开销的讨论，当前保留但未来可优化。

风险与影响

- 风险：断路器记录逻辑调整可能影响非 streaming 路径，但测试充分且无回归；BreakerTrackedStream 增加极微开销（bench 证实）；并发安全通过单任务控制保证。
- 影响：所有 streaming 请求受益于客户端断连时的及时资源回收；团队需理解新抽象，但模块设计良好；无需改动 API 或用户配置。

关联脉络

暂无直接关联的上游 PR，但该 PR 为 model-gateway 的流处理层增加了通用的 BreakerTrackedStream，为未来统一所有路由器的流转发路径奠定基础。