PR #32734 完整报告

sgl-project/sglang

[metrics] Split tokenizer request metrics by stream

合并时间：2026-08-05 03:53

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32734

执行摘要

- 一句话：按流式与非流式拆分 tokenizer 五项核心指标
- 推荐动作：值得精读，但重点不在代码量而在两个设计决策：一是可观测性维度如何与下游存储 schema 对齐（thrift 保留字导致标签改名），二是 straggler 判定为何固定 stream=true。合并后建议补充一条针对 labelnames 与上报值一致性的单测，并核对现有 TTFT 告警是否仍符合预期。若要精读，聚焦 TokenizerMetricsCollector 的 observe_one_finished_request 与 check_time_to_first_token_straggler 两个函数即可。

功能与动机

PR body 明确指出：流式与非流式请求的延迟分布本质不同（流式请求按设计常驻），但 tokenizer 的按请求指标缺少 stream 维度，导致请求计数、e2e 延迟与 token 吞吐无法按流量类别分段，e2e 延迟直方图混叠两种分布形态，影响容量评估与 SLO 判读。PR 同时明确排除 num_aborted_requests_total（调用点只有 rid，abort_all 场景 flag 有歧义）与 per-batch 调度器指标（混合批次无法归类），避免在语义不清晰处引入伪维度。

实现拆解

1. 指标定义扩展（python/sglang/srt/observability/metrics_collector.py）：TokenizerMetricsCollector.__init__将 prompt_tokens_total、generation_tokens_total、num_requests_total、histogram_time_to_first_token、histogram_e2e_request_latency 的 labelnames 统一扩展为 list(labels.keys()) + ["is_streaming"]；spec_verify_calls_total、cached_tokens_total、inter-token latency 等其余指标保持原标签集，控制 series 膨胀面。
2. 请求维度透传（python/sglang/srt/managers/tokenizer_manager.py）：TokenizerManager.collect_metrics 在请求结束上报处新增 is_streaming=getattr(state.obj, "stream", False)，用 getattr 兜底保证兼容未设置该字段的请求对象。
3. 上报与判定逻辑调整（metrics_collector.py）：observe_one_finished_request 新增 is_streaming: bool = False 参数，函数内构造 stream_labels 字典，用于 5 个目标指标的 inc / observe；observe_time_to_first_token 的 label 从 stream 改为 is_streaming；check_time_to_first_token_straggler 固定以 is_streaming="true" 的直方图作为 p99 基准，因为流式请求才是首 token 尾延迟（straggler）的观察对象。
4. 标签命名对齐下游存储：中途 commit（23ccd17）将 label 从 stream 改名 is_streaming。原因是下游指标存储把该维度固化为列名，而 stream 是 thrift 保留关键字无法作为列名；

若 label 名与存储不匹配，依赖 alias 解析不可靠，会导致同一 task 内各 stream 计数器系列交错，windowed max 聚合会低估计数。该约束以注释形式固化在 histogram_time_to_first_token 定义处，防止未来误改。

5. 测试与验证配套：本 PR 未附带自动化测试；作者说明已在内部部署验证 5 个指标带新标签导出且可抓取，既有基于 base 标签的 dashboard 在新维度上仍可聚合。

关键文件：

- python/sglang/srt/observability/metrics_collector.py（模块 指标采集；类别 source；类型 core-logic；符号 TokenizerMetricsCollector, observe_one_finished_request, observe_time_to_first_token, check_time_to_first_token_straggler）：核心改动文件：5 个指标新增 is_streaming 维度、上报方法与 straggler 判定逻辑调整，标签命名（stream -> is_streaming）的约束也固化在此处的代码注释中。
- python/sglang/srt/managers/tokenizer_manager.py（模块 请求管理；类别 source；类型 core-logic；符号 collect_metrics）：指标数据入口：请求完成时把请求对象上的 stream 标志透传给采集器，是 is_streaming 维度的数据来源。

关键符号：TokenizerMetricsCollector.init, TokenizerMetricsCollector.observe_one_finished_request, TokenizerMetricsCollector.observe_time_to_first_token, TokenizerMetricsCollector.check_time_to_first_token_straggler, TokenizerManager.collect_metrics

关键源码片段

python/sglang/srt/observability/metrics_collector.py

核心改动文件：5 个指标新增 is_streaming 维度、上报方法与 straggler 判定逻辑调整，标签命名（stream -> is_streaming）的约束也固化在此处的代码注释中。

以下是 TokenizerMetricsCollector.observe_one_finished_request 的完整实现（head 版本整理），核心是 stream_labels 的构造与使用：

```
# TokenizerMetricsCollector 类方法：请求完成时上报按请求维度的指标
def observe_one_finished_request(
    self,
    labels: Dict[str, str],
    prompt_tokens: int,
    generation_tokens: int,
    cached_tokens: int,
    e2e_latency: float,
    has_grammar: bool,
    cached_tokens_details: Optional[Dict[str, Any]] = None,
    spec_verify_ct: int = 0,
    # 新增参数：标记该请求是否为流式请求，默认 False 以兼容旧调用方
    is_streaming: bool = False,
):
    # 在基础标签之上叠加 is_streaming 维度，
    # 值为 "true"/"false" 字符串，符合 Prometheus label value 约定
    stream_labels = {
```

```
    **labels,
    "is_streaming": "true" if is_streaming else "false",
}
self.prompt_tokens_total.labels(**stream_labels).inc(prompt_tokens)
self.generation_tokens_total.labels(**stream_labels).inc(generation_tokens)
if spec_verify_ct > 0:
    self.spec_verify_calls_total.labels(**labels).inc(spec_verify_ct)

# 缓存命中 token 按来源拆分上报，与 stream 维度无关，
# 因此沿用原始 labels，避免无谓的 series 增长
if cached_tokens > 0:
    if cached_tokens_details:
        def report_cache_source(source: str, value: int):
            if value > 0:
                source_labels = {**labels, "cache_source": source}
                self.cached_tokens_total.labels(**source_labels).inc(value)

        report_cache_source("device", cached_tokens_details.get("device", 0))
        report_cache_source("host", cached_tokens_details.get("host", 0))

        # storage 字段仅在启用 L3 存储后端时出现
        if "storage" in cached_tokens_details:
            storage_tokens = cached_tokens_details.get("storage", 0)
            if storage_tokens > 0:
                backend = (
                    cached_tokens_details.get("storage_backend") or "unknown"
                )
                report_cache_source(f"storage_{backend}", storage_tokens)
    else:
        # 向后兼容兜底路径：无明细时统一记到 cache_source = "total"
        labels_total = {**labels, "cache_source": "total"}
        self.cached_tokens_total.labels(**labels_total).inc(cached_tokens)

# 请求计数与 e2e 延迟直方图按流量类别拆分，
# 解决流式与非流式分布形态不同导致的直方图混叠问题
self.num_requests_total.labels(**stream_labels).inc(1)
if has_grammar:
    self.num_so_requests_total.labels(**labels).inc(1)
self.histogram_e2e_request_latency.labels(**stream_labels).observe(
    float(e2e_latency)
)
self.prompt_tokens_histogram.labels(**labels).observe(float(prompt_tokens))
self.uncached_prompt_tokens_histogram.labels(**labels).observe(
    float(prompt_tokens - cached_tokens)
)
self.generation_tokens_histogram.labels(**labels).observe(
    float(generation_tokens)
)
```

TTFT 上报与 straggler 判定是另一处关键语义——判定基准固定为流式请求：

```
# TTFT 观察与首 token straggler 判定：基准只统计流式请求
def observe_time_to_first_token(
    self, labels: Dict[str, str], value: float, *, stream: bool
):
    # TTFT 直方图上报，is_streaming 维度与 e2e 延迟直方图保持一致
    self.histogram_time_to_first_token.labels(
        **labels, is_streaming="true" if stream else "false"
    ).observe(value)


def check_time_to_first_token_straggler(self, value: float) -> bool:
    # 仅基于流式请求的 TTFT 分布做尾延迟判定：
    # 非流式请求一次性返回全部输出，不构成首 token straggler 观察对象。
    # 注意此处直接读取 prometheus_client 的 bucket 内部字段，
    # 在 multiprocess 模式下只反映当前进程的本地视图
    his = self.histogram_time_to_first_token.labels(
        **self.labels, is_streaming="true"
    )
    total_observations = sum(bucket._value for bucket in his._buckets)
    if total_observations < 100:
        return False
    p99_threshold = total_observations * 0.99
    cumulative_count = 0
    for i, bucket in enumerate(his._buckets):
        cumulative_count += bucket._value
        if cumulative_count > p99_threshold:
            return value >= his._upper_bounds[i]
    return False
```

python/sglang/srt/managers/tokenizer_manager.py

指标数据入口：请求完成时把请求对象上的 stream 标志透传给采集器，是 is_streaming 维度的数据来源。

```
# TokenizerManager.collect_metrics：请求完成分支，透传 stream 标志
if state.finished:
    # 请求对象上的 stream 字段标记是否为流式请求；
    # 使用 getattr 兜底，兼容未设置该字段的请求对象
    self.metrics_collector.observe_one_finished_request(
        labels,
        recv_obj.prompt_tokens[i],
        completion_tokens,
        recv_obj.cached_tokens[i],
        state.time_stats.get_e2e_latency(),
        self._request_has_grammar(state.obj),
        cached_tokens_details,
        spec_verify_ct=spec_verify_ct,
        is_streaming=getattr(state.obj, "stream", False),
```

)

评论区精华

唯一有效 review 评论来自合并者 merrymercy，要求把 tokenizer_manager.py 调用处的关键字参数从 stream 重命名为 is_streaming（与指标 label 命名统一），最终在最后一次 commit 4c84034 中落地。更有价值的设计讨论体现在 commit 23ccd17 的提交说明里：标签名 stream 与下游存储冲突——stream 是 thrift 保留关键字，无法成为 schema 列名；label 名不匹配时依赖 alias 解析不可靠，会导致按任务维度的 windowed max 聚合低估计数器。此外 PR body 对拆分范围做了明确取舍：不拆 num_aborted_requests_total（调用点只有 rid，abort_all 时无法确定 stream 属性）与 per-batch 调度器指标（混合批次无法归类）。

- 调用处关键字参数重命名 stream -> is_streaming (style): 已采纳，最后一次 commit 将调用改为 is_streaming=getattr(state.obj, "stream", False)。
- 标签名与下游存储对齐（thrift 保留字问题） (design): 采用 is_streaming 作为 5 个指标的统一维度名，并在代码注释中固化该约束。
- 拆分范围取舍：abort 计数与 per-batch 指标不拆 (design): 维持原样，仅拆 5 个按请求维度清晰的指标。

风险与影响

- 风险：
  1. 指标基数翻倍：5 个指标各新增一个 bool 枚举标签，series 数量翻倍；在指标量大的集群可观测性开销有所上升，但影响可控。
  2. straggler 判定语义：check_time_to_first_token_straggler 的 p99 基准固定为 is_streaming="true" 的流式请求分布，非流式请求的 TTFT 不再参与判定；依赖全量 TTFT 尾延迟告警的团队需复核预期。
  3. 缺少测试覆盖：没有新增单测或 e2e 测试验证 labelnames 与上报值一致；若后续调用方漏传 is_streaming，Prometheus client 会在 .labels() 处抛 MissingLabelName 异常，回归风险集中在采集链路。
  4. 消费端兼容性：新增 label 改变 series 身份，依赖旧 label 集合的告警表达式需复查；PR 声称基于 base 标签的 dashboard 聚合不受影响。
  5. 即时生效：无 server_args 开关，合并后所有部署立即启用新维度。 - 影响：影响面集中在 tokenizer 指标链路：5 个按请求维度的指标从单一 series 拆分为流式 / 非流式两类，运维与容量规划可按流量类别独立观察请求数、TTFT、e2e 延迟与 token 吞吐，SLO 与告警更加精准。对系统性能影响可忽略（每次请求完成仅多构造一个含一个键的 dict 与一次 getattr）。对团队而言，本次敲定了指标 label 与下游存储列名对齐的约束（thrift 保留字问题），后续新增维度时应沿用 is_streaming 这类存储安全命名；同时 TTFT straggler 检查的语义被收紧到流式流量。 - 风险标记：指标基数翻倍，无测试覆盖，straggler 判定语义变化，label 命名影响消费端

关联脉络

- PR #33375 [Observability] Add startup, memory, and hybrid SWA diagnostics: 直接改动同一文件 python/sglang/srt/observability/metrics_collector.py，是同一观测体系内的持续扩展。
- PR #33562 fix(metrics): clear forward occupancy on idle: 同一指标监控链路的正确性修复，说明该模块处于活跃维护期，指标语义精确化是持续演进方向。