PR #24416 完整报告

sgl-project/sglang

[PD] Fix KV transfer metrics

合并时间：2026-05-06 18:44

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24416

执行摘要

- 一句话：修复 KV 传输指标并统一接口
- 推荐动作：值得精读，尤其是 KVTransferMetric 的抽象设计和 duration_between 的安全模式。开发者可借此学习如何在不破坏现有接口的前提下统一异构后端的指标报告。此外，review 中关于预计算长度和避免 assert 的建议也值得在日常开发中参考。

功能与动机

根据 PR 描述，当前 KV 传输指标存在三个主要问题：NIXL 的传输时间计算不正确；KV 传输大小计算错误；请求时间统计中的持续时间可能为负（特别是中止请求）。需要统一修复并重构接口，确保指标准确性并提高代码可维护性。

实现拆解

1. 定义统一指标数据类：在 python/sglang/srt/disaggregation/base/conn.py 中新增 KVTransferMetric dataclass，包含 transfer_latency_s 和 transfer_total_bytes 字段（可选），并在 BaseKVSender 中声明 get_transfer_metric 抽象方法。
2. 实现后端指标跟踪：
   - CommonKVSender（common/conn.py）新增 _record_transfer_indices 方法，在每次 send 调用时累加索引数；并在 get_transfer_metric 中根据预计算的 kv_item_lens_sum 和 state_item_lens_sum 计算总字节数。
   - NixlKVSender（nixl/conn.py）在首次发送数据时记录时间戳，轮询完成时计算传输延迟并存入 _transfer_metric。
   - MooncakeKVSender（mooncake/conn.py）调用 _record_transfer_indices。
   - FakeKVSender（fake/conn.py）提供默认实现。
3. 重构请求时间统计：req_time_stats.py 中的 compute_and_observe_kv_transfer_metrics 方法改为接受 KVTransferMetric 对象，优先使用后端报告的延迟，否则回退到基于队列时间计算。同时引入 duration_between 辅助函数，在 convert_to_duration 中替换直接减法，防止负持续时间。
4. 更新调用入口：prefill.py 中的 process_disagg_prefill_inflight_queue 不再自行计算页面大小和总字节，而是直接调用 req.disagg_kv_sender.get_transfer_metric() 获取指标，并增加了对 FAKE_BOOTSTRAP_HOST 和 dummy cp rank 的跳过逻辑。

关键文件：

- python/sglang/srt/observability/req_time_stats.py（模块 请求时间统计；类别 source；类型 core-logic；符号 duration_between, compute_and_observe_kv_transfer_metrics, convert_to_duration）：核心改动，重构指标计算方法并引入安全时间差计算
- python/sglang/srt/disaggregation/base/conn.py（模块 传输基础；类别 source；类型 data-contract；符号 KVTransferMetric, get_transfer_metric）：定义统一的指标数据类 KVTransferMetric 和抽象方法 get_transfer_metric，是本次重构的接口层。
- python/sglang/srt/disaggregation/common/conn.py（模块 通用连接；类别 source；类型 core-logic；符号 get_transfer_metric, _record_transfer_indices）：实现了 CommonKVSender 中的指标跟踪逻辑，是核心实现。
- python/sglang/srt/disaggregation/prefill.py（模块 预处理调度；类别 source；类型 core-logic）：调用入口更新，使用新的指标接口。
- python/sglang/srt/disaggregation/nixl/conn.py（模块 NIXL 连接；类别 source；类型 core-logic）：增加 NIXL 后端的传输时间记录，修复指标不准确问题。

关键符号：duration_between, get_transfer_metric, _record_transfer_indices, compute_and_observe_kv_transfer_metrics, convert_to_duration

关键源码片段

python/sglang/srt/observability/req_time_stats.py

核心改动，重构指标计算方法并引入安全时间差计算

```
def duration_between(start: float, end: float) -> float:
    # 安全计算时间差，防止负持续时间，返回 0 避免误导
    return max(0.0, end - start)

def compute_and_observe_kv_transfer_metrics(
    self, transfer_metric: KVTransferMetric
) -> Optional[dict]:
    # 没有传输大小信息时直接返回
    if transfer_metric.transfer_total_bytes is None:
        return None

    # 优先使用后端报告的延迟，否则回退到基于队列时间的估算
    if transfer_metric.transfer_latency_s is not None:
        transfer_latency_s = transfer_metric.transfer_latency_s
    else:
        if self.prefill_transfer_queue_entry_time <= 0 or self.completion_time <= 0:
            return None
        transfer_latency_s = (
            self.completion_time - self.prefill_transfer_queue_entry_time
        )

    if transfer_latency_s > 0:
        latency_ms = transfer_latency_s * 1000
        total_bytes = transfer_metric.transfer_total_bytes
        total_mb = total_bytes / (1024 * 1024)
```

```
            speed_gb_s = (total_mb / 1024) / transfer_latency_s
            self.transfer_total_mb = total_mb
            self.transfer_speed_gb_s = speed_gb_s
            if self.enable_metrics:
                self.metrics_collector.observe_kv_transfer_metrics(
                    latency_ms=latency_ms, total_mb=total_mb, speed_gb_s=speed_gb_s
                )
            return {"latency_ms": latency_ms, "total_mb": total_mb, "speed_gb_s": speed_gb_s}
        return None
```

python/sglang/srt/disaggregation/base/conn.py

定义统一的指标数据类 KVTransferMetric 和抽象方法 get_transfer_metric，是本次重构的接口层。

```
@dataclasses.dataclass
class KVTransferMetric:
    # 后端无法分离传输延迟时，transfer_latency_s 可为 None
    transfer_latency_s: Optional[float] = None
    # 传输的总字节数，若为 None 则表示无法计算
    transfer_total_bytes: Optional[int] = None

class BaseKVSender(ABC):
    # ... 其他抽象方法 ...

    @abstractmethod
    def get_transfer_metric(self) -> KVTransferMetric:
        # 各后端必须实现此方法，返回自身当前请求的传输指标
        ...
```

python/sglang/srt/disaggregation/common/conn.py

实现了 CommonKVSender 中的指标跟踪逻辑，是核心实现。

```
def get_transfer_metric(self) -> KVTransferMetric:
    # 根据累计的 KV 索引数和 state 索引数，乘以预计算的每项字节总数，得到总字节量
    total_bytes = self._transfer_num_kv_indices * self.kv_mgr.kv_item_lens_sum
    total_bytes += self._transfer_num_state_indices * self.kv_mgr.state_item_lens_sum
    self._transfer_metric.transfer_total_bytes = total_bytes
    return self._transfer_metric

def _record_transfer_indices(
    self, kv_indices: npt.NDArray[np.int32], state_indices: Optional[List[int]]
):
    # 在每次 send 时记录发送的索引数量，用于后续计算总字节
    self._transfer_num_kv_indices += len(kv_indices)
    if state_indices is not None:
        self._transfer_num_state_indices += len(state_indices)
```

评论区精华

Review 过程中 gemini-code-assist[bot] 提出了三条关键建议：

- 在 compute_and_observe_kv_transfer_metrics 中使用 assert 检查 transfer_total_bytes 可能导致崩溃，建议改为提前返回。该建议已被采纳。
- get_transfer_metric 中每次调用都计算 sum(kv_item_lens) 效率不高，建议在初始化时预计算。已在 CommonKVManager 中缓存。
- 在 prefill.py 中直接访问 req.disagg_kv_sender.kv_mgr 可能因后端实现不同而缺少属性，建议使用 getattr 安全访问。也已修复。 最终审核人 ShangmingCai 批准了该 PR，认为改动对混合注意力场景提供了更精确的指标。
- 取消 assert 改为优雅返回 (correctness)：已在 commit cc048fd 中采纳，改为 if transfer_metric.transfer_total_bytes is None: return result if result else None。
- 预计算 kv_item_lens 和 state_item_lens 的和 (performance)：已在 CommonKVManager 初始化中计算并存储 kv_item_lens_sum 和 state_item_lens_sum，之后直接引用。
- 使用 getattr 安全访问 kv_mgr (correctness)：已采纳，改为 getattr(req.disagg_kv_sender, 'kv_mgr', None)。

风险与影响

- 风险：主要风险包括：
  - 抽象基类接口变更：所有 BaseKVSender 子类必须实现 get_transfer_metric，当前四个后端（nixl、mooncake、fake、mori）均已适配，但未来新增后端若忘记实现会导致运行时错误。
  - 缺少测试覆盖：本次变更未包含相应的单元测试或集成测试，指标计算正确性依赖人工验证和运行时观察。
  - 指标数据模型变化：旧接口（传递 num_tokens, page_size 等）被移除，任何外部代码如果直接调用旧签名的 compute_and_observe_kv_transfer_metrics 将无法编译。
  - 影响：对用户透明，但运维监控数据会更准确，有助于排查性能问题。对系统而言，指标采集职责从调用方转移到后端，降低了耦合，便于未来扩展更多传输特征。团队维护成本降低，代码更清晰。
  - 风险标记：缺少测试覆盖 , 抽象接口变更

关联脉络

- PR #24188 [NIXL][XPU] Use np.uint64 for pointer/length arrays in disaggregation KV transfer: 都涉及 nixl/conn.py 的修改，但侧重点不同，本 PR 修复指标计算。