PR #32388 完整报告

sgl-project/sglang

Observability enhancement for HiCache

合并时间：2026-08-06 05:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32388

执行摘要

- 一句话：HiCache 新增分层命中、备份回载与丢弃指标
- 推荐动作：值得精读。重点关注三类设计： 1) HiCacheAck 同时携带按 pool 的 token 数与总字节数，让带宽计算和分池监控只依赖 ack 快照； 2) make_timing_event_pair() 与设备端事件的配合，使 backup/load-back 耗时统计避开调度器同步噪声； 3) 新旧缓存路径共用 _log_write_ack_metrics 的接入方式。建议结合 PR #33580（统一树核心接口）一起阅读，理解 UnifiedRadixCache 对旧缓存路径的替代进程中观测性如何保持对齐。若团队在用 write-back 策略，可基于新指标建立丢弃率与 L2 带宽告警。

功能与动机

PR body 明确说明动机：'This PR introduce several new metrics for HiCache observability including: separated cache hit counter from tiers, backuped token, load back token and dropped token (for write-back policy) counter, more accurate timing and memory footprint counter for live bandwidth computation.' 即现有 HiCache 只有整体 evict/load-back 计数，缺少按 device/host/storage 层的命中拆分，缺少 write-back 场景下 D->H 备份与 H->D 回载的真实流量和耗时，无法计算 L2 实际带宽，也无法观测内存压力下 KV 被丢弃的损失，因此需要这套更细粒度的指标。

实现拆解

实现按 5 个层次推进：

1. 指标注册层：metrics_collector.py 的 StorageMetricsCollector 新增 sglang:prefill_effective_tokens_total、sglang:hicache_backup_tokens_total、sglang:hicache_backup_bytes_total、sglang:hicache_backup_duration_seconds、sglang:load_back_bytes_total、sglang:hicache_dropped_tokens_total 共 6 个 Prometheus 序列，并预置 input/device_hit/host_hit/storage_hit 四档 mode 与各 pool 的 0 值序列；同时把 eviction 直方图默认桶上限从 1 秒扩展到 60 秒，因为 write-back 的阻塞式合并备份可能持续数秒。配套新增 increment_effective_prefill_tokens、increment_backup_num_tokens、increment_backup_num_bytes、increment_load_back_num_bytes、observe_backup_duration、increment_dropped_tokens 等方法。
2. 传输量定义：managers/cache_controller.py 的 HiCacheAck 增加 num_tokens_by_pool 与 num_bytes 字段；基础版 _transfer_num_bytes 只计算 kv + draft 池字节；

hybrid_cache_controller.py 提供增强版 _num_tokens_by_pool（按 pool 统计且排除复用其他 pool 索引的 sidecar 传输）与 _transfer_num_bytes（计入 draft piggyback 与 sidecar 字节）。start_writing 与 start_loading 改用 make_timing_event_pair() 生成 ack 专用事件对，使耗时统计只覆盖真正的 D->H/H->D 拷贝。

3. 消费端上报：hiradix_cache.py、unified_radix_cache.py、hi_mamba_radix_cache.py 的 writing_check/loading_check 在 ack 完成时按 pool 累加 backup/load-back token 与字节，并观测耗时直方图；unified 侧新增 _record_dropped_tokens（用 drop 前后 tracker 差值计算各池丢弃量），旧hiradix侧在_drop_subtree_no_host中直接上报释放的token。
4. 分层命中统计：schedule_batch.py 抽出纯函数 split_cached_prefix_by_tier；schedule_policy.py 的 PrefillAdder 在 _update_prefill_budget 中按 device/host/storage 拆分非 retracted 请求的命中 prefix 并累加到新增字段；metrics_reporter.py 的 PrefillStats 读取这些字段并在 report_prefill_stats 中调用 increment_effective_prefill_tokens。
5. 测试配套：test_hicache_staged_write_back_dispatch.py 为 fake host pool 补齐 anchor_entry/entry_map/size_per_token，并在 setUp/tearDown 中清理 _timing_events_supported 缓存；test_hicache_load_back_timing.py 与 test_prefill_adder.py 补充新字段断言。

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 increment_effective_prefill_tokens, increment_load_back_num_tokens, increment_backup_num_tokens, increment_backup_num_bytes）：全部新指标的注册地与上报方法所在，是本次观测性增强的核心数据面；新增 prefill 分层命中、backup/load-back 按池 token 与字节计数、耗时直方图及 dropped tokens 计数，并扩展 eviction 直方图桶。
- python/sglang/srt/mem_cache/hybrid_cache/hybrid_cache_controller.py（模块 混合缓存；类别 source；类型 entrypoint；符号 _num_tokens_by_pool, _transfer_num_bytes）：为 merged transfer op 计算按 pool 的 token 数与总字节数的核心定义点；start_writing/start_loading 用 timing event pair 精确测时并把结果写入 ack，是带宽指标的数据来源。
- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 统一缓存；类别 source；类型 core-logic；符号 _record_dropped_tokens, _log_write_ack_metrics）：新缓存路径的指标接入点，包含 _record_dropped_tokens 与 _log_write_ack_metrics，以及 evict 指标口径从全部层改为 full 层的调整。
- python/sglang/srt/managers/cache_controller.py（模块 缓存控制；类别 source；类型 entrypoint；符号 _transfer_num_bytes）：HiCacheAck 数据结构新增 num_tokens_by_pool/num_bytes 字段并落地基础版 _transfer_num_bytes，write/load 两条 ack 队列都携带新字段。
- python/sglang/srt/mem_cache/hiradix_cache.py（模块 缓存核心；类别 source；类型 core-logic；符号 _log_write_ack_metrics）：旧 HiRadixCache 路径同步接入 backup/load-back 指标与 dropped 计数，保证与统一缓存口径一致。

- python/sglang/srt/managers/schedule_batch.py（模块 批量调度；类别 source；类型 core-logic；符号 split_cached_prefix_by_tier）：抽出 split_cached_prefix_by_tier 纯函数供调度两侧复用，避免 breakdown 计算逻辑漂移。
- python/sglang/srt/managers/schedule_policy.py（模块 调度策略；类别 source；类型 dependency-wiring；符号 _update_prefill_budget）：PrefillAdder 记录 device/host/storage 分层命中 token，是 prefill 分层命中率指标的来源。
- python/sglang/srt/managers/scheduler_components/metrics_reporter.py（模块 指标上报；类别 source；类型 dependency-wiring；符号 PrefillStats）：PrefillStats 扩展分层命中字段并在 report_prefill_stats 中调用 increment_effective_prefill_tokens，打通调度层到 Prometheus 的上报链路。
- python/sglang/srt/mem_cache/hi_mamba_radix_cache.py（模块 Mamba 缓存；类别 source；类型 core-logic；符号 loading_check）：Mamba 专用缓存路径的 loading_check 同步按 pool 上报 load-back 与字节指标。
- test/registered/unit/mem_cache/test_hicache_staged_write_back_dispatch.py（模块 单元测试；类别 test；类型 test-coverage；符号 setUp, tearDown）：更新 fake event/host pool 以兼容新 ack 字段，并在 setUp/tearDown 清理 timing 探测缓存，防止跨测试污染。
- test/registered/unit/mem_cache/test_hicache_load_back_timing.py（模块 单元测试；类别 test；类型 test-coverage）：覆盖 load-back 计时与按 pool 上报的新逻辑。
- test/registered/unit/managers/test_prefill_adder.py（模块 单元测试；类别 test；类型 test-coverage）：扩展 PrefillAdder 分层命中统计的断言。

关键符号：increment_effective_prefill_tokens, increment_backup_num_tokens, increment_backup_num_bytes, increment_load_back_num_tokens, increment_load_back_num_bytes, observe_backup_duration, increment_dropped_tokens, split_cached_prefix_by_tier, _transfer_num_bytes, _num_tokens_by_pool, _log_write_ack_metrics, _record_dropped_tokens, writing_check, loading_check, _update_prefill_budget, start_writing, start_loading

关键源码片段

python/sglang/srt/observability/metrics_collector.py

全部新指标的注册地与上报方法所在，是本次观测性增强的核心数据面；新增prefill分层命中、backup/load-back 按池 token 与字节计数、耗时直方图及 dropped tokens 计数，并扩展 eviction 直方图桶。

```
# ======== HiCache L2 (host DRAM) 备份 / 回载与丢弃指标 ========
# D->H 备份在 --hicache-write-policy write_back 下是阻塞式合并操作，
# 可能持续数秒，因此默认直方图桶比 load-back 更宽
bucket_backup_duration = [
    0.001, 0.002, 0.005, 0.01, 0.02, 0.05,
    0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0, 30.0, 60.0,
]

# D->H 备份的按池 token 计数；覆盖所有 write policy，
```

```
        # 与 L3 层的 sglang:backuped_tokens_total 相区分
        self.backup_num_tokens = Counter(
            name='sglang:hicache_backup_tokens_total',
            documentation='The number of tokens backed up from GPU to local '
            'host DRAM (L2), by host pool (kv, swa, mamba, ...).',
            labelnames=list(labels.keys()) + ['pool'],
        )

        # write-back 在主机内存压力下未备份即销毁的 token 计数,
        # 按 reason 与 pool 两个维度区分
        self.hicache_dropped_tokens = Counter(
            name='sglang:hicache_dropped_tokens_total',
            documentation='The number of device KV tokens destroyed without a '
            'host backup, by pool (kv, swa, ...) and reason.',
            labelnames=list(labels.keys()) + ['reason', 'pool'],
        )

    def increment_backup_num_tokens(self, num_tokens: int, pool: str) -> None:
        # 每次 ack 完成时按池累加，配合耗时直方图可算 D->H 带宽
        self.backup_num_tokens.labels(**self.labels, pool=pool).inc(num_tokens)

    def increment_dropped_tokens(self, num_tokens: int, reason: str, pool: str) -> None:
        self.hicache_dropped_tokens.labels(**self.labels, reason=reason, pool=pool).inc(
            num_tokens
        )
```

python/sglang/srt/mem_cache/hybrid_cache/hybrid_cache_controller.py

为 merged transfer op 计算按 pool 的 token 数与总字节数的核心定义点；start_writing/start_loading 用 timing event pair 精确测时并把结果写入 ack，是带宽指标的数据来源。

```
    def _num_tokens_by_pool(self, op: CacheOperation) -> dict[str, int]:
        # 按 host pool 统计一次合并传输的 token 数（anchor + extra pools），
        # D->H write 与 H->D load 的 ack 共用；复用其他 pool 索引的
        # sidecar 传输不计入，避免重复计数
        counts = {self.mem_pool_host.anchor_entry.name.value: len(op.device_indices)}
        for transfer in op.pool_transfers or []:
            if transfer.indices_from_pool is not None or transfer.host_indices is None:
                continue
            name = transfer.name.value
            counts[name] = counts.get(name, 0) + len(transfer.host_indices)
        return counts

    def _transfer_num_bytes(self, op: CacheOperation) -> int:
        # 全 pool 合计字节数，包含 draft piggyback 与 sidecar 传输；
        # 与耗时直方图 sum 相除即可得 D->H / H->D 的实际带宽
        kv_tokens = len(op.device_indices)
        num_bytes = kv_tokens * self.mem_pool_host.anchor_entry.host_pool.size_per_token
        if self.has_draft:
```

```
            num_bytes += kv_tokens * self.mem_pool_host_draft.size_per_token
        # 统计 sidecar 可复用的源 pool 槽位数
        source_len = {self.mem_pool_host.anchor_entry.name: kv_tokens}
        for t in op.pool_transfers or []:
            if t.indices_from_pool is None and t.host_indices is not None:
                source_len[t.name] = len(t.host_indices)
        for t in op.pool_transfers or []:
            entry = self.mem_pool_host.entry_map.get(t.name)
            if entry is None:
                continue
            if t.indices_from_pool is not None:
                num_slots = source_len.get(t.indices_from_pool, 0)
            else:
                num_slots = len(t.host_indices) if t.host_indices is not None else 0
            num_bytes += num_slots * entry.host_pool.size_per_token
        return num_bytes
```

python/sglang/srt/mem_cache/unified_radix_cache.py

新缓存路径的指标接入点，包含 _record_dropped_tokens 与 _log_write_ack_metrics，以及 evict 指标口径从全部层改为 full 层的调整。

```
# 组件类型到 host pool 标签名的映射；与 hicache_backup_tokens_total
# 以及 host 占用 gauge 使用的 pool 名保持一致
_COMPONENT_POOL_LABEL = {
    ComponentType.FULL: PoolName.KV.value,
    ComponentType.SWA: PoolName.SWA.value,
    ComponentType.MAMBA: PoolName.MAMBA.value,
}

def _record_dropped_tokens(
    self,
    tracker: dict[ComponentType, int],
    freed_before_drop: dict[ComponentType, int],
) -> None:
    """记录主机内存压力下未备份即被丢弃的按池 token 数。

    调用方在 drop 前用 freed_before_drop = dict(tracker) 保存快照，
    drop 完成后 tracker 中各组件的新增释放量即为丢弃量。
    """
    if self.metrics_collector is None:
        return
    for ct, freed in tracker.items():
        dropped = freed - freed_before_drop[ct]
        if dropped > 0:
            self.metrics_collector.increment_dropped_tokens(
                num_tokens=dropped,
                reason='host_pressure',
                pool=_COMPONENT_POOL_LABEL[ct],
            )
```

评论区精华

该 PR 没有人工 review 评论（唯一的 issue 评论是 Gemini Code Assist 的停止服务提示），因此没有可引用的审阅争论。值得注意的设计取舍都写在代码注释里：一是 eviction_duration_seconds 桶范围扩大到 60 秒，注释说明 write-back 下 eviction 包含阻塞式 D->H 备份，可能持续数秒；二是 _num_tokens_by_pool 明确排除 sidecar 传输（sidecar transfers reusing another pool's indices are excluded），而 _transfer_num_bytes 则相反地包含它们，两个口径互补，避免重复计数或漏计。

- 暂无高价值评论线程

风险与影响

- 风险：
  - 共享结构扩展：HiCacheAck 是 NamedTuple，新增字段有默认值，向后兼容；但所有构造点需要显式填充 num_tokens_by_pool/num_bytes，遗漏只会导致指标缺失而非崩溃，不易察觉。
  - 签名变更：increment_load_back_num_tokens 由 (num_tokens) 变为 (num_tokens, pool)，三个调用点已同步修改；仓库外或旧分支的自定义调用会直接失败。
  - 统计口径调整：unified_radix_cache.py 的 evict 从 sum(tracker.values()) 改为 tracker[BASE_COMPONENT_TYPE]，sglang:evicted_tokens_total 将不再包含 SWA/MAMBA 层，依赖旧口径的告警或面板可能出现跳变。
  - 指标膨胀：新增多个带 pool/reason/mode 标签的 Counter 与预置 0 序列，PROMETHEUS_MULTIPROC_DIR 下序列数随 host pool 数量线性增加。
  - 性能：ack 处理循环增加了字典遍历与 Prometheus 累加；直方图桶扩展对内存影响很小，风险低。
- 影响：
  - 运维 / 用户：新增 sglang:prefill_effective_tokens_total{device_hit,host_hit,storage_hit}、hicache_backup_*、load_back_*、hicache_dropped_tokens_total、hicache_backup_duration_seconds 等指标，可直接绘制分层命中率与 L2 实时带宽，量化 write-back 在内存压力下的丢弃损失。
  - 系统：启用 HiCache 的实例会多上报若干时间序列；未启用 HiCache 的部署仅在 collector 初始化时多注册空序列，开销可忽略。
  - 团队：新老两条缓存路径（UnifiedRadixCache/HiRadixCache）使用同一套 _log_write_ack_metrics 口径，降低后续迁移与对比成本；调度侧 split_cached_prefix_by_tier 成为命中拆分的单一事实来源。
  - 风险标记：HiCacheAck 扩展影响所有调用点，evict 指标统计口径调整，新增标签序列导致指标膨胀，新旧缓存路径需同步维护

关联脉络

- PR #33580 [Unified Radix Cache] Complete the tree-core interface boundary: 与本 PR 同时触及 unified_radix_cache.py：33580 完成统一树核心接口边界，本 PR 在其上加

装 L2 指标上报与 dropped 计数，两者共同推动 UnifiedRadixCache 逐步替代旧 HiRadixCache。

- PR #33595 Measure prefill busy time between launches: 同属调度器可观测性增强，且同样修改 schedule_batch.py 与 metrics_reporter 相关路径；本 PR 的 prefill_effective_tokens_total 与 prefill busy time 互为补充，构成更完整的负载画像。
- PR #33348 [DCP] Match the replicated draft KV pool's page granularity to its allocator: 同属 HiCache L2 数据面：33348 修复 draft KV pool 页粒度越界写，本 PR 的字节统计也专门处理 draft piggyback 传输，两者都是 HiCache 传输正确性的组成部分。
- PR #33556 Add Ling-3.0-flash cookbook: 同属 hicache 功能线：33556 提供 HiCache 部署与调优文档入口，本 PR 补充的带宽与丢弃指标为 cookbook 中的容量规划提供量化依据。