

PR #56061 完整报告

vllm-project/vllm

[4/N] Expose HiSparse cache metrics via KV connector stats

合并时间: 2026-09-13 01:47

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56061>

执行摘要

- 一句话: HiSparse 缓存指标接入 KVConnectorStats 管线
- 推荐动作: 值得精读, 尤其适合关注 KV connector 扩展机制与 GPU 侧遥测的同学。建议重点看三处: (1) HiSparseConnectorWorker.get_kv_connector_stats 的两阶段采样——快照读取与采样触发在同一个 hook 里交织, _metrics_pending / _metrics_event / _metrics_calls 三个状态位的配合是理解其正确性的关键; (2) connector.py 只加三个 hook 就完成接入, 是「零核心改动」设计主张的直接证据, 可与 #53782 的 finish_step 方案对照阅读; (3) 提交历史中的两次 CI follow-up, 展示了 vLLM 对「指标断言该放哪个测试套件」的分层偏好。若要在生产读取这些指标, 请按代码而非 PR 描述确认指标名。

功能与动机

PR body 明确说明动机: 这是 #53782 的替代实现, 「reporting the same HiSparse hot-buffer stats through the existing KVConnectorStats pipeline instead of adding a new finish_step connector hook」。原方案需要在 KVConnectorBase 上新增 hook, 并连带改 scheduler、outputs、aggregator、loggers 和 stats.py, 共 16 个文件; 而 NIXL、offloading connector、mooncake 已经证明「transfer/cache telemetry goes through KVConnectorStats and the connector's build_prom_metrics, with zero core plumbing」。HiSparse 的热缓冲驻留计数器本质就是连接器遥测, 因此可以复用 offloading connector 使用的同一槽位, 把 diff 从 16 个文件压缩到 9 个, 并顺带获得 CLI 日志 (body 示例: INFO .. Cumulative HiSparse stats: hits=12 lookups=4 l3_lookups=64)。

实现拆解

1. 内核侧计数 (csrc/libtorch_stable/hisparsed_kernels.cu、ops.h、torch_bindings.cpp)
: `hisparsed_resolve_residency_kernel` 新增 `unsigned long long* stats` 参数 ([2], hits/misses), 在 `tid == 0` 分支用 `atomicAdd` 累计本 batch 的 `total_hits` / `total_misses`; `hisparsed_resolve_residency` 主函数新增参数校验 (必须是 contiguous、dim==1、UInt64、numel==2 的 CUDA 张量)。ops.h 与 torch_bindings.cpp 同步签名, 绑定 schema 的 alias 注解从 `b!` 起整体顺延 (`stats` 占 `b!`, `attention_indices` 起顺延到 `c!~h!`), 因此这是 positional 参数插入, 任何按位置调用的地方都必须重新编译。
2. 运行时共享状态 (vllm/v1/hisparsed/runtime.py): `HiSparseIndexGroup` 新增 `swap_stats` (device 端 `uint64[2]` 计数器)、`swap_stats_host` (pinned CPU 缓冲, 用于 D2H) 与 `stats_row_bytes`; `HiSparseRuntime.__init__` 里 `index_group.stats_row_bytes`

`+= row_bytes`, 把 leader/follower 的行字节累加到 group; `reset_hot_state` 额外执行 `group.swap_stats.zero()`, 避免重置热状态后残留计数; `_resolve_residency` 把 `group.swap_stats` 传给 `torch.ops._C_cache_ops.hispase_resolve_residency`。

3. Worker 两阶段采样 (`vllm/distributed/kv_transfer/kv_connector/v1/hispase/worker.py`) : 新增模块级 `_METRICS_INTERVAL = 2000` 与实例字段 `_metrics_calls / _metrics_event / _metrics_pending`。`get_kv_connector_stats()` 先读取上一轮已完成 D2H 的快照 (`_metrics_pending and _metrics_event.query()`) , 对每个 leader runtime 取 `swap_stats_host.tolist()`, 非零才 `record_snapshot(hits, misses, misses * stats_row_bytes)`; 随后 `_metrics_calls += 1`, 到达间隔且无 pending、非 CUDA graph capture 时, 用 `compute_stream.wait_stream(copy_stream) / copy_stream.wait_stream(compute_stream)` 做双向同步, `copy_(non_blocking=True)` 后立即 `zero()`, 最后 `event.record()` 标记 pending。body 特别指出采样计数在无 forward 步也推进, 避免长 prefill 拉长采样节奏。
4. 统计容器与 Prometheus 导出 (新文件 `hispase/stats.py`) : `HiSparseKVConnectorStats` 继承 `KVConnectorStats`, `data` 为 `dict[str, list[int]]` (每个 list 元素是一次快照增量), 实现 `reset / record_snapshot / aggregate` (按 key extend) / `reduce` (求和, 用于 CLI 日志) / `is_empty`; `HiSparsePromMetrics` 继承 `KVConnectorPromMetrics`, 按 `_HISPASE_COUNTERS` 声明式创建 `vllm:hispase_cache_hits`、`vllm:hispase_cache_misses`、`vllm:hispase_host_to_device_bytes` 三个计数器, `observe` 对 list 中每个增量逐值 `inc`。
5. 连接器接入标准管线 (`hispase/connector.py`) : 新增 `get_kv_connector_stats()` (转发 worker, worker 为空返回 None)、`build_kv_connector_stats(data)` (`data is None` 时返回空容器, 否则 `HiSparseKVConnectorStats(data=data)`) 与 `build_prom_metrics()` (返回 `HiSparsePromMetrics`)。这三个 hook 由既有 scheduler / stats 聚合管线自动调用, 因此核心文件零改动。
6. 测试配套: 新增 `tests/v1/kv_connector/unit/test_hispase_stats.py`, 覆盖 `is_empty`、`record_snapshot+reduce` 求和、`aggregate` 扩展与跳过空 stats、`to_dict` → `build_kv_connector_stats` 往返; `tests/v1/worker/test_utils.py` 更新 `_make_hispase_worker` fixture, 新增 `test_hispase_worker_get_kv_connector_stats_reads_completed_snapshot`, 断言首次调用返回 None 但触发 `wait_stream` 双向调用、`event.record()` 与 `swap_stats` 清零, 第二次调用读到 `[12, 4] / 64` 字节。body 说明模型评测不适用 (仅遥测、不影响输出)。

关键文件:

- `vllm/distributed/kv_transfer/kv_connector/v1/hispase/stats.py` (模块 指标统计; 类别 source; 类型 core-logic; 符号 `HiSparseKVConnectorStats`, `post_init`, `reset`, `record_snapshot`) : 本 PR 的核心新增文件: 定义 `HiSparse` 的统计容器与 Prometheus 导出, 是整个「复用 `KVConnectorStats` 管线」方案的承载点。
- `vllm/distributed/kv_transfer/kv_connector/v1/hispase/worker.py` (模块 指标采样; 类别 source; 类型 core-logic; 符号 `get_kv_connector_stats`, `_METRICS_INTERVAL`) : 新增 `get_kv_connector_stats` 的两阶段采样逻辑: 每 2000 次调用触发一次设备计数器 D2H 快照, 并用 CUDA event 保证读取顺序, 是整条指标链路最复杂的一段。

- `vllm/distributed/kv_transfer/kv_connector/v1/hisparse/connector.py` (模块 连接器接入; 类别 `source`; 类型 `core-logic`; 符号 `get_kv_connector_stats`, `build_kv_connector_stats`, `build_prom_metrics`): 连接器侧只加三个 `hook` 就完成接入, 是「不新增 `finish_step`、零核心管线改动」这一设计主张的直接证据。
- `vllm/v1/hisparse/runtime.py` (模块 运行时状态; 类别 `source`; 类型 `core-logic`; 符号 `HiSparseIndexGroup.init`, `HiSparseRuntime.init`, `reset_hot_state`, `_resolve_residency`): `HiSparseIndexGroup` 新增 `device/pinned` 双份计数器与 `stats_row_bytes`, 并把 `swap_stats` 注入 `hisparse_resolve_residency`, 是计数器的数据来源。
- `csrc/libtorch_stable/hisparse_kernels.cu` (模块 内核计数; 类别 `source`; 类型 `core-logic`; 符号 `hisparse_resolve_residency_kernel`, `hisparse_resolve_residency`): 内核侧真正产生计数的位置: 由 `tid==0` 线程 `atomicAdd total_hits/total_misses`, 并新增 `stats` 张量的类型 / 形状校验。
- `csrc/libtorch_stable/torch_bindings.cpp` (模块 算子绑定; 类别 `source`; 类型 `data-contract`; 符号 `hisparse_resolve_residency`): `stable ABI` 的 `schema` 变更: `stats` 插入为 `b!`, 其余 `mutable alias` 注解整体顺延到 `c!~h!`, 是跨语言签名的关键同步点。
- `csrc/libtorch_stable/ops.h` (模块 内核签名; 类别 `source`; 类型 `data-contract`; 符号 `hisparse_resolve_residency`): C++ 侧函数声明的同步改动, 与 `kernel` 与绑定 `schema` 三处必须严格一致。
- `tests/v1/kv_connector/unit/test_hisparse_stats.py` (模块 单元测试; 类别 `test`; 类型 `test-coverage`; 符号 `test_is_empty_on_fresh_stats`, `test_record_snapshot_and_reduce`, `test_aggregate_extends_snapshot_deltas`, `test_aggregate_skips_empty_stats`): 为新统计容器提供完整的单元覆盖, 包括 `is_empty`、`reduce` 求和、`aggregate` 合并与 `build_kv_connector_stats` 往返。
- `tests/v1/worker/test_utils.py` (模块 采样测试; 类别 `test`; 类型 `test-coverage`; 符号 `test_hisparse_worker_get_kv_connector_stats_reads_completed_snapshot`): 新增 `worker` 采样生命周期的回归测试, 并更新 `HiSparse worker` 的测试 `fixture` 以补齐三个采样状态位。

关键符号: `HiSparseConnectorWorker.get_kv_connector_stats`,
`HiSparseConnector.get_kv_connector_stats`, `HiSparseConnector.build_kv_connector_stats`, `HiSparseConnector.build_prom_metrics`, `HiSparseKVConnectorStats.record_snapshot`,
`HiSparseKVConnectorStats.aggregate`, `HiSparseKVConnectorStats.reduce`,
`HiSparseKVConnectorStats.is_empty`, `HiSparsePromMetrics.observe`,
`HiSparseRuntime.reset_hot_state`, `hisparse_resolve_residency`,
`hisparse_resolve_residency_kernel`

关键源码片段

`vllm/distributed/kv_transfer/kv_connector/v1/hisparse/worker.py`

新增 `get_kv_connector_stats` 的两阶段采样逻辑: 每 2000 次调用触发一次设备计数器 D2H 快照, 并用 `CUDA event` 保证读取顺序, 是整条指标链路最复杂的一段。

`_METRICS_INTERVAL = 2000` # 每 2000 次 `get_kv_connector_stats` 调用触发一次设备计数器快照

```

class HiSparseConnectorWorker:
    # ... 初始化阶段新增三个采样状态位
    # self._metrics_calls = 0
    # self._metrics_event = torch.Event()
    # self._metrics_pending = False

    def get_kv_connector_stats(self) -> HiSparseKVConnectorStats | None:
        # 第一阶段：读取上一轮已经完成 D2H 的快照。
        stats = None
        if self._metrics_pending and self._metrics_event.query():
            stats = HiSparseKVConnectorStats()
            for runtime in self.leader_runtimes:
                group = runtime.index_group
                hits, misses = group.swap_stats_host.tolist()
                # 只有产生过活动的 runtime 才记录，避免写出无意义的 0 增量。
                if hits or misses:
                    # 字节数是估算值：miss 行数 × 该 index group 的累计行字节数。
                    stats.record_snapshot(hits, misses, misses * group.stats_row_bytes)
            self._metrics_pending = False
            if stats.is_empty():
                stats = None

        # 第二阶段：累计调用次数，达到间隔后发起新一次设备→主机拷贝。
        # 计数在无 forward 步也推进，避免长 prefill 把采样节奏拉长。
        self._metrics_calls += 1
        if (
            self._metrics_calls % _METRICS_INTERVAL == 0
            and not self._metrics_pending
            and not torch.cuda.is_current_stream_capturing()
        ):
            compute_stream = current_stream()
            for runtime in self.leader_runtimes:
                group = runtime.index_group
                # D2H 走 copy_stream，并通过双向 wait_stream 建立顺序：
                # 先等计算流把计数器写完，再让计算流等拷贝流完成清零。
                compute_stream.wait_stream(group.copy_stream)
                group.swap_stats_host.copy_(group.swap_stats, non_blocking=True)
                group.swap_stats.zero_()
                group.copy_stream.wait_stream(compute_stream)
            # event 标记本轮 D2H 完成，供下次调用判断快照是否可读。
            self._metrics_event.record()
            self._metrics_pending = True
        return stats

```

[vllm/distributed/kv_transfer/kv_connector/v1/hisparsed/connector.py](#)

连接器侧只加三个 hook 就完成接入，是「不新增 finish_step、零核心管线改动」这一设计主张的直接证据。

```

class HiSparseConnector(KVConnectorBase_V1, SupportsHMA):

```

```
# ... 其余 hook 保持不变，仅新增以下三个 telemetry 入口

def get_kv_connector_stats(self) -> KVConnectorStats | None:
    # 调度侧每步通过这个 hook 拉取 worker 已完成的快照。
    if self.connector_worker is None:
        return None
    return self.connector_worker.get_kv_connector_stats()

@classmethod
def build_kv_connector_stats(
    cls, data: dict[str, Any] | None = None
) -> KVConnectorStats | None:
    # 反序列化入口：日志或聚合路径用 data 重建；
    # data 为 None 时返回空容器占位，保持与其它 connector 一致的语义。
    return (
        HiSparseKVConnectorStats(data=data)
        if data is not None
        else HiSparseKVConnectorStats()
    )

@classmethod
def build_prom_metrics(
    cls,
    vllm_config: VllmConfig,
    metric_types: dict[type[PromMetric], type[PromMetricT]],
    labelnames: list[str],
    per_engine_labelvalues: dict[int, list[object]],
) -> KVConnectorPromMetrics:
    # 与 NIXL / offloading connector 相同的契约，因此核心 stats.py 无需改动。
    return HiSparsePromMetrics(
        vllm_config, metric_types, labelnames, per_engine_labelvalues
    )
```

评论区精华

评审讨论量不大，但几条信息对理解合入顺序与测试策略很关键：

LucasWilkinson (APPROVED) : "LGTM, thanks for doing this! good to land once <https://github.com/vllm-project/vllm/pull/53781> lands"

这明确了依赖顺序——HiSparse 的 host-resident 稀疏 MLA 层 (#53781) 要先落地，本 PR 才具备可测的运行时基础。

mgoin (APPROVED, 无文字)

claude[bot]: "This pull request is from a fork — automated review is disabled." —— fork 来源导致自动 review 未运行，本 PR 的静态检查主要依赖人工 review 与 CI，没有任何机器人审查记录。

mergify[bot]: "This pull request has merge conflicts that must be resolved before it can be merged." LucasWilkinson: "Retargeted to `main` after the previous base branch was merged and deleted. ... since the previous base was squash-merged."

分支流程上经历了一次 `retarget + rebase`: 原 `base` 分支被 `squash-merge` 后删除, 重新指向 `main`, 随后三次 `/ci run` (Buildkite #87897 / #88476 / #88527) 验证通过。

提交历史还揭示了一次测试归属的返工: 作者提交了主体实现, 随后 LucasWilkinson 用两个 follow-up commit 收敛测试——先修好 HiSparse metrics 回归测试的 `swap` 生命周期 (改用 `cache handle` 重放查找, 让断言真正走到 `hit/miss` 计数器), 再把冗余的指标断言从 GPU attention 套件移除, 理由是覆盖已由专用 `connector / worker` 测试提供。

- 用 `KVConnectorStats` 替代自建 `finish_step` hook (design): 采纳 `KVConnectorStats` 方案, diff 从 16 个文件降到 9 个、核心管线零改动; mgoin 与 LucasWilkinson 均 APPROVED。
- 合入依赖 #53781 先落地 (other): #53781 已在同仓库历史中合入 ([3/N] HiSparse: `host-resident sparse-MLA decode hot-buffering`), 本 PR 随后 merge。
- 指标回归测试的归属与 `swap` 生命周期 (testing): 指标管线覆盖收敛到 `tests/v1/kv_connector/unit/test_hisparsed_stats.py` 与 `tests/v1/worker/test_utils.py`, 不再扩展 GPU attention 测试套件。
- fork 来源导致自动 review 未执行 (other): 未触发自动 review; 需要人工留意 mypy / Ruff 之外的静态问题 (作者称本地 `pre_commit` 全通过)。

风险与影响

- 风险:

1. 内核与绑定签名变更 (`csrc/libtorch_stable`): `stats` 参数被插入到 `hisparsed_resolve_residency` 的 `positional` 参数中间, `ops.h`、`torch_bindings.cpp`、`kernel` 三处必须严格同步。虽然仅 C++ 侧受影响, 但 `torch_bindings.cpp` 的 `alias` 注解整体重排 (`b!~h!`), 任何依赖旧位置参数或旧 `schema` 的编译产物与新 Python 代码混用时会直接运行期报错, 必须从源码重编译。
1. 指标语义为估算值: worker 侧 `host_to_device_bytes = misses * group.stats_row_bytes`, 而 `stats_row_bytes` 是在每个 `HiSparseRuntime.__init__` 中累加到 `index group` 的总行字节数, `swap_stats` 又是 `group` 级别共享的 (`leader` 与 `follower` 的 `kernel` 调用共同 `atomicAdd` 到同一个 [2] 张量)。这意味着字节数是「总 miss 数 × 总行宽」的乘积近似, 只有在各层 `miss` 分布与行宽成比例时才等于真实 D2H 传输量; 用于容量规划时需注意这一语义 (此结论来自代码推导, 建议以实际部署观测为准)。
2. 采样与 CUDA graph 的交互: 采样路径依赖 `torch.cuda.is_current_stream_capturing()` 判断来跳过 `capture` 阶段, 并用 `_metrics_event` + 双向 `wait_stream` 保证 D2H 与清零的顺序。若 `capture` 判断失效或 `event` 复用不当, 可能在 `graph replay` 期间插入隐式同步, 测试里通过 `monkeypatch is_current_stream_capturing` 和 `current_stream` 来绕过, 说明这条路径的真实行为依赖 GPU 运行时环境。
3. 指标命名的文档漂移: PR body 声称「Same three Prometheus counters: `vllm:hisparsed_hits`, `vllm:hisparsed_lookups`, `vllm:hisparsed_l3_lookups`」, CLI 示例也写

作 `hits=12 lookups=4 l3_lookups=64`；但最终代码中的 `_HISPARSE_COUNTERS` 是 `cache_hits / cache_misses / host_to_device_bytes`，`reduce()` 的日志键为 `HiSparse hot-buffer hits/misses` 与 `HiSparse host-to-device bytes`。描述沿用了原 PR 的措辞，落地时改名，使用方应以代码为准，仪表盘 / 告警若照抄 body 会配置到不存在的指标名。

4. 测试覆盖缺口：`HiSparsePromMetrics.observe` 没有直接单测，GPU 套件中的指标断言已被 follow-up commit 移除，因此「计数器最终是否真的进入 Prometheus / CLI 日志」缺少端到端自动化验证，只有 stats 容器层与 worker 采样层的单元测试。
- 影响：面向运维与模型服务团队：HiSparse 部署现在可以抓取 `vllm:hisparse_cache_hits / cache_misses / host_to_device_bytes`，并在日志中看到 `Cumulative HiSparse stats` 的汇总行，热缓冲命中率与 `host→device` 回填压力首次可观测，直接影响 HiSparse 的调参与容量规划。
 - 面向系统：由于走 `KVConnectorStats` 契约，调度器、输出结构、loggers 与核心 `stats.py` 完全没有改动，回归面被限制在 HiSparse 自身路径与三处 C++ 签名；采样每 2000 次调用才触发一次 `pinned copy + event`，D2H 开销可忽略，且计数器在无 forward 步也推进，采样节奏与请求形态解耦。
 - 面向团队：本 PR 是 `KVConnectorStats` + `build_prom_metrics` 契约的第三个完整参考实现（前有 NIXL、offloading connector），为后续连接器补充遥测提供了可复制的模板，也把「telemetry 属于连接器职责、不进入核心管线」这一设计取向固化下来。
 - 风险标记：CUDA 内核签名变更，指标语义为估算值，采样依赖 CUDA graph capture 判断，缺少 Prometheus 端到端测试，描述与实现的指标命名漂移

关联脉络

- PR #53781 [3/N] HiSparse: host-resident sparse-MLA decode hot-buffering: 本 PR 的直接前置依赖（同属 [4/N] 系列），提供 host-resident 稀疏 MLA KV 层与 HiSparse runtime；LucasWilkinson 明确要求它先合入，本 PR 的 `swap_stats` 与 `leader_runtimes` 均依赖其数据结构。
- PR #53782 HiSparse metrics 的 `finish_step` hook 实现（PR body 中的被替代方案）：PR body 说明本 PR 是它的替代实现：原方案新增 `KVConnectorBase.finish_step` 并改 `scheduler/outputs/loggers/stats.py`，本 PR 用 `KVConnectorStats` 达到同样效果；两者是同一功能的两种架构取舍，值得对照阅读。
- PR #54689 [Bugfix][NIXL] Don't evict a remote engine a transfer is still reading from: PR body 把 NIXL 列为本 PR 遵循的 `KVConnectorStats` 契约参照实现之一；两者都改动 `vllm/distributed/kv_transfer/kv_connector/v1` 下的连接器代码，属于同一连接器遥测 / 生命周期演进脉络。