PR #34316 完整报告

sgl-project/sglang

[metrics] Fix prefill FLOPs estimate to count prefix and per-request causal pairs

合并时间：2026-08-18 06:18

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34316

执行摘要

- 一句话：修复 prefill FLOPs 估算：按请求计入 prefix 因果对与 KV 读取
- 推荐动作：值得精读。本 PR 是 observability 指标 " 从数学正确性到物理真实性 " 的典型样本：数学修复本身直白，但 read_bytes 的演进（sum(c*p) → sum(prefix_lens)）展示了如何用 HBM 峰值带宽做合理性检验来校正建模假设。测试技巧也可借鉴——把所有系数清零、把 attention 系数置 1，让估计值归约成可断言的整数。建议关注 hnyls2002 对 FA 内核流式读取行为的分析，以及作者对 prefix_lens 索引对齐假设的明确声明。

功能与动机

关联 issue #34298 指出 _estimate_prefill_perf 只用 sum(extend_lens) 计算 attention 对数，产生两个独立错误：一是丢弃 chunk 与缓存 prefix 之间的 c * P attention 对，二是把 batch 内多个请求合并成一条序列而虚构跨请求 attention 对（例如两个 100 token 的 prefill 实际是 10,100 对，旧实现却按 200 token 序列估为 20,100 对）。PR body 强调："the reported est. prefill TFLOPS/s then becomes proportional to 1 / gap_latency rather than to any real measure of throughput"——即固定 --chunked-prefill-size 下每 chunk 的估算 FLOPs 完全相同，指标退化为逆延迟曲线，与 docs 中推荐该指标做趋势分析的初衷背道而驰。

实现拆解

变更共分四步，涉及 python/sglang/srt/managers/scheduler_components/metrics_reporter.py（源码）与 test/registered/unit/observability/test_forward_pass_metrics.py（测试）两个文件。

1. 抽取共享 pair 计数 helper：在 metrics_reporter.py 新增 @staticmethod _prefill_attention_pairs(batch)，用 zip 同时遍历 batch.extend_lens 与 batch.prefix_lens，按请求累计两项：c * p（chunk 对缓存 prefix 的 attention 对）与 c * (c + 1) / 2.0（chunk 内部因果对），并返回总和。同步更新 _prefill_sol_suffix 的 docstring，将其契约从 " 子类自行计算 pair 数 " 改为 " 调用 _prefill_attention_pairs 获取精确因果对数 "，便于模型架构子类复用。
2. 改写 _estimate_prefill_perf 的 FLOPs 项：context_product 由原来的 tokens * (tokens + 1) / 2.0 替换为 self._prefill_attention_pairs(batch)。这一处替换同时修复两个数学错误：补上被丢弃的 prefix 因果对，并避免多请求被合并成单序列导致约 2 倍的虚高。

3. read_bytes 增加 prefix KV 读取项并收敛语义：第二个提交（Lenoplus42）按 sum(c * p) * _kv_cache_bytes_per_token（每条 query-key 对计一次 KV 读取）实现；review 中 hnyls2002 指出该模型假设 FA 类 kernel 无 KV 复用，在 c = 8192 / p = 128K 时会算出远超 HBM 峰值带宽的读取量。最终第三个提交（hnyls2002 直接提交）改为 sum(batch.prefix_lens) * _kv_cache_bytes_per_token——每个 chunk 的 query 共享一次对前缀 KV 的流式读取，作为逻辑下限，与文档中 "modeled traffic" 措辞一致，也与 decode 路径按 total_context 计费的姿势对齐。
4. 单元测试锁定语义：在 test_forward_pass_metrics.py 新增 TestEstimatedPrefillPerf 类，通过 setUp 把线性、权重、激活、KV 等系数全部清零、attention 系数置 1，使返回的 FLOPs 恰好等于因果 pair 数，便于精确断言；覆盖缓存 prefix 计费、prefix KV 每 chunk 只读一次、同 batch 多请求互不 attend、mixed prefill+decode 行使用各自上下文四个场景。测试继承 CustomTestCase 基类并注册到 base-a-test-cpu 套件。

配套说明：PR 无文档改动（body 未勾选 "Update documentation"）；CI 仅 CPU 路径；/rerun-test test_forward_pass_metrics.py 定向跑测通过。

关键文件：

- python/sglang/srt/managers/scheduler_components/metrics_reporter.py（模块 指标上报；类别 source；类型 core-logic；符号 _prefill_attention_pairs, _estimate_prefill_perf）：源码主路径：新增 _prefill_attention_pairs 共享 helper，改写 _estimate_prefill_perf 的 attention 对数计算与 read_bytes 计费，修复指标退化为 1/latency 的问题。
- test/registered/unit/observability/test_forward_pass_metrics.py（模块 指标测试；类别 test；类型 test-coverage；符号 TestEstimatedPrefillPerf, setUp, _pair_count, test_chunk_is_charged_for_its_cached_prefix）：新增 TestEstimatedPrefillPerf 类，用系数清零技巧把 FLOPs 归约为可断言的 pair 数，覆盖缓存 prefix 计费、prefix KV 按 chunk 只读一次、多请求不互相关注、mixed 批次使用各自上下文四个核心语义。

关键符号：_prefill_attention_pairs, _estimate_prefill_perf

关键源码片段

test/registered/unit/observability/test_forward_pass_metrics.py

新增 TestEstimatedPrefillPerf 类，用系数清零技巧把 FLOPs 归约为可断言的 pair 数，覆盖缓存 prefix 计费、prefix KV 按 chunk 只读一次、多请求不互相关注、mixed 批次使用各自上下文四个核心语义。

```
class TestEstimatedPrefillPerf(CustomTestCase):
    """覆盖 `est. prefill TFLOPS/s` 与 `estimated_flops` 背后的因果 pair 计数语义。"""

    def setUp(self):
        self.scheduler = types.SimpleNamespace()
        self.scheduler.waiting_queue = []
        self.scheduler.disaggregation_mode = DisaggregationMode.NULL
        self.reporter = _make_reporter(self, self.scheduler)
        # 把所有非 attention 系数清零、attention 系数置 1，
        # 这样返回的 FLOPs 就恰好等于因果 pair 数量，便于精确断言
```

```
        self.reporter._linear_flops_per_token = 0.0
        self.reporter._attn_dot_flops_coeff = 1.0
        self.reporter._weight_read_bytes_per_token = 0.0
        self.reporter._qkv_act_bytes_per_token = 0.0
        self.reporter._prefill_attn_act_read_per_token = 0.0
        self.reporter._kv_cache_bytes_per_token = 0.0
        self.reporter._ffn_act_bytes_per_token = 0.0

    def _pair_count(self, extend_lens, prefix_lens):
        batch = types.SimpleNamespace(extend_lens=extend_lens, prefix_lens=prefix_lens)
        flops, _, _ = self.reporter._estimate_prefill_perf(batch)
        return flops

    def test_chunk_is_charged_for_its_cached_prefix(self):
        # 4 个新 token 续接在 3 token 的缓存 prefix 之后：
        # 4 * 3（对前缀的 attention 对）+ 4 * 5 / 2（chunk 内部因果对）
        self.assertEqual(self._pair_count([4], [3]), 4 * 3 + 4 * 5 / 2)

    def test_prefix_kv_is_read_once_per_chunk(self):
        # prefix 的 KV 按 chunk 只流式读取一遍，而不是按 query-key 对计数
        self.reporter._kv_cache_bytes_per_token = 1.0
        batch = types.SimpleNamespace(extend_lens=[4], prefix_lens=[3])
        _, read_bytes, _ = self.reporter._estimate_prefill_perf(batch)
        self.assertEqual(read_bytes, 3)

    def test_requests_in_one_batch_do_not_attend_to_each_other(self):
        # 两个独立 100 token 的 prefill 应为 2 * (100 * 101 / 2)，而不是
        # 200 * 201 / 2（旧实现会把它们合并成一条 200 token 的序列）
        self.assertEqual(self._pair_count([100, 100], [0, 0]), 2 * (100 * 101 / 2))

    def test_mixed_prefill_and_decode_rows_use_their_own_context(self):
        # mix_with_running 会把 running 请求追加为 extend_len = 1、
        # prefix_len = 当前上下文长度，估计器必须使用各自上下文
        self.assertEqual(
            self._pair_count([8, 1, 1], [0, 100, 200]),
            8 * 9 / 2 + (100 + 1) + (200 + 1),
        )
```

评论区精华

review 中 hnyls2002 给出了两条关键评论：

1. 要求收敛变更范围（COMMENTED）："Rather than deferring to #34298, please fold the rest in here: read_bytes needs the matching sum(c * p) * _kv_cache_bytes_per_token term (decode already charges it), and the pair count should move into a shared helper since _prefill_sol_suffix needs it too." 作者据此在第二个提交中合入 read_bytes 修复，并抽出共享 pair 计数 helper。

2. 自我修正 read_bytes 建模（COMMENTED）："Correcting my earlier comment on read_bytes: sum(c*p) doesn't hold at chunk scale. It corresponds to naive attention with zero KV reuse, while FA-class kernels stream the prefix roughly once per query block. At c=8192 / p=128K on a 70B-class TP8 config that's ~44 TB of KV reads for one chunk; divided by the chunk latency the implied bandwidth is well above HBM peak." 他建议以 sum(prefix_lens) 作为逻辑下限，并提到 issue 中的 p + c/2 也可行但 c/2 与现有 tokens * _prefill_attn_act_read_per_token 项重叠。最终方案采纳了 sum(prefix_lens)，第三个提交即由此产生。

- 要求把 read_bytes 修复与共享 helper 合入本 PR (design): 作者在第二个提交中合入 read_bytes 项并抽出共享 pair 计数 helper，原先的计划是留给后续 issue 处理。
- read_bytes 按 sum(c*p) 计费会超出 HBM 峰值带宽，需改用逻辑下限 (performance): 第三个提交由 hnyls2002 直接落地为 sum(prefix_lens) * _kv_cache_bytes_per_token，测试相应改为 test_prefix_kv_is_read_once_per_chunk。
- CI 触发与测试范围确认 (testing): github-actions 回报测试通过（ubuntu-latest 1 test ⚠）。

风险与影响

- 风险：风险主要集中在指标语义与运维依赖上：
- 指标绝对值大幅抬升：PR body 明确提示 sglang:estimated_flops_per_gpu_total 的绝对值在长上下文下可能上升 1-2 个数量级。任何基于旧值校准的告警阈值、面板 Y 轴、容量规划脚本都会失效，需要同步重校。
- read_bytes 语义为逻辑下限而非物理流量：最终采用 sum(prefix_lens)，代表 " 每个 chunk 对前缀 KV 流式读取一遍 " 的建模下限；真实的 FA/Triton 内核因 tiling 与 L2 复用会有差异，estimated_read_bytes_per_gpu_total 与 HBM 实际流量之间的换算关系并不严格，误用会导致带宽估算偏差。
- 依赖列表索引对齐：_prefill_attention_pairs 与 read_bytes 均依赖 extend_lens 与 prefix_lens 按请求一一对应；PR 有意省略了 issue 提出的 fallback 与 c <= 0 守卫，理由是两条列表在所有写入路径上同写。若未来出现单独修改其中一条列表的代码路径，可能引入静默错位。
- 架构相关常量未建模：issue 中左开放的 MLA、MoE、稀疏 attention 对 pair 数或 FLOPs 系数的影响仍未处理，_attn_dot_flops_coeff 仍是统一系数，估算精度对特定架构（如 DeepSeek）仍有偏差。
- 影响：影响范围集中在 observability 层，不改变任何推理行为：
- 用户侧：est. prefill TFLOPS/s 在 chunked-prefill 下恢复物理含义，能正确反映 prefix 增长带来的计算量变化；观测面板与告警需按新绝对值重校准。
- 系统侧：估算器位于日志路径，改动仅把一次 batch 级 sum 换成一次对两个 list 的 generator 遍历，开销可忽略；无内存、显存或调度行为变化。
- 团队侧：_prefill_sol_suffix 的钩子契约变化（从 " 子类自行计算 " 到 " 调用共享 helper"），继承 SchedulerMetricsReporter 的架构子类（如 DSpark 等）需要同步调整以获得一致的 pair 计数。

- 风险标记：指标绝对值预期上升 1-2 个数量级，read_bytes 采用逻辑下限语义，依赖 extend_lens 与 prefix_lens 索引对齐，架构相关 FLOPs 常量未建模

关联脉络

- PR #34298 [Bug] Prefill FLOPs estimate ignores prefix_lens, so est. prefill TFLOPS/s degenerates into 1/latency across chunked-prefill chunks: 这是本 PR 直接修复的关联 issue（非 PR）：全部动机、两个数学错误的分析、read_bytes 讨论的 p + c/2 备选方案均来自该 issue；PR body 明确标注 "Fixes #34298"。