

PR #53464 完整报告

vllm-project/vllm

[Pooling] Improve BGE-M3 sync pooling throughput by up to 3.13%

合并时间: 2026-08-24 20:33

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53464>

执行摘要

- 一句话: MRV2 pooling 免除 GPU token ID 拷贝, BGE-M3 吞吐最高 +3.13%
- 推荐动作: 值得精读。重点关注三个设计决策: 一是在 CPU 上完成 doc/query 索引计算、仅异步拷贝最终索引 (async_tensor_h2d), 这是减少 GPU-CPU 间无谓数据搬运的通用模式; 二是 PoolingCursor 预计算 CPU 状态 (partial_prefill、finished_mask) 并随切片重算, 避免每步 .tolist(); 三是 v1→v2 移除 consistent batch 的架构讨论, 理解为什么 prompt_token_ids 不应作为默认 GPU 元数据存在。

功能与动机

PR body 明确指出: "BGE-M3's sparse embedding path uses SPLADE, which only needs prompt token IDs on CPU, so copying the full prompt-token-ID batch to GPU adds unnecessary serving overhead", 即当前 MRV2 在 requires_token_ids=True 时会完整拷贝 prompt token ID 到 GPU, 而对 BGE-M3/Jina 这类只在 CPU 上做 token 级后处理的池化任务而言属于纯开销。yewentao256 在 review 中补充了架构动机: "One design change from v1 to v2 is that we remove the consistent batch", 即该优化也是 MRV2 去 consistent-batch 设计的一部分。

实现拆解

1. 在 vllm/model_executor/models/jina.py 的 JinaForRankingPool 中新增 get_pooling_updates, 声明 requires_token_ids=True; forward 改为通过 pooling_metadata.get_prompt_token_ids_cpu() 读取 CPU 端 token IDs, 在 CPU 上完成 doc/query 分隔 token 的匹配与索引拼接, 再通过 async_tensor_h2d 只将最终选中索引拷到 GPU 后做 projector 投影, 同时移除 gpu_sync_allowed 包装。
2. 在 vllm/v1/worker/gpu/pool/pooling_runner.py 的 _get_pooling_metadata 中删除 prompt_token_ids 的 GPU 副本创建, PoolingMetadata.prompt_token_ids 固定传 None; prompt_token_ids_cpu 的 pin_memory 改为仅在非 CPU 设备启用; _dummy_pooler_run_task 同步改为只创建 CPU 张量。pool() 中 finished_mask 改用缓存的 get_finished_mask()。
3. 在 vllm/v1/worker/gpu_input_batch.py 中, add_request 不再把 pooling_params.requires_token_ids 写入 logits_processing_needs_token_ids (恒为 False), 并更新注释明确“设备端 prompt tokens 仅用于 penalties 或显式请求 GPU token IDs 的池化方法”; get_pooling_metadata 同样将 prompt_token_ids 置 None。

4. 在 `vllm/v1/pool/metadata.py` 中为 `PoolingCursor` 新增 `partial_prefill` 与 `finished_mask` 字段, `build_pooling_cursor` 用 `numpy` 一次性预计算, `__getitem__` 切片时重算, 并提供 `get_finished_mask()`; `vllm/v1/worker/gpu_model_runner.py` 的 `_pool` 与 `pooling_runner.pool` 均改用该方法, 避免每步执行 `.tolist()` 与张量比较。
5. 测试配套: 新增 `test_build_pooling_cursor_caches_cpu_status` 与 `test_pooling_params_update_combines_token_id_requirements` (`tests/model_executor/layers/test_pooler_methods.py`), 并更新 `tests/v1/worker/test_gpu_input_batch.py` 与 `tests/models/language/pooling/test_sparse_pooler.py` 以适配新数据契约。

关键文件:

- `vllm/model_executor/models/jina.py` (模块 模型实现; 类别 `source`; 类型 `data-contract`; 符号 `get_pooling_updates`, `forward`): 核心优化点: `JinaForRankingPool` 新增 `get_pooling_updates` 声明 `requires_token_ids`, `forward` 改为 CPU 端计算 `doc/query` 索引, 仅 H2D 拷贝选中索引, 移除 `gpu_sync_allowed`。
- `vllm/v1/pool/metadata.py` (模块 池化元数据; 类别 `source`; 类型 `core-logic`; 符号 `PoolingCursor`, `get_finished_mask`, `build_pooling_cursor`): `PoolingCursor` 新增预计算的 `partial_prefill` 与 `finished_mask`, 并提供 `get_finished_mask()`, 消除 `pooling` 每步的设备比较与 `.tolist()` 开销。
- `vllm/v1/worker/gpu/pool/pooling_runner.py` (模块 池化执行; 类别 `source`; 类型 `core-logic`; 符号 `_get_pooling_metadata`, `pool`, `_dummy_pooler_run_task`): 执行路径核心: `_get_pooling_metadata` 不再生成 GPU 版 `prompt_token_ids`, `PoolingMetadata.prompt_token_ids` 固定为 `None`; `finished_mask` 改为读取缓存。
- `vllm/v1/worker/gpu_input_batch.py` (模块 输入批处理; 类别 `source`; 类型 `core-logic`): 输入批处理中 `pooling` 请求的 `logits_processing_needs_token_ids` 不再随 `requires_token_ids` 置位, 设备端 `token IDs` 语义收紧。
- `vllm/v1/worker/gpu_model_runner.py` (模块 模型执行器; 类别 `source`; 类型 `data-contract`; 符号 `_pool`): 池化收尾路径: `_pool` 中 `finished_mask` 由手动 `zip` 推导改为 `get_finished_mask()`, 统一使用缓存结果。
- `tests/model_executor/layers/test_pooler_methods.py` (模块 池化方法; 类别 `test`; 类型 `test-coverage`; 符号 `test_build_pooling_cursor_caches_cpu_status`, `test_pooling_params_update_combines_token_id_requirements`): 新增两个核心单测: 验证 `PoolingCursor` CPU 状态缓存与切片重算, 以及 `PoolingParamsUpdate` 的 `token ID` 需求合并逻辑。

关键符号: `JinaForRankingPool.get_pooling_updates`, `JinaForRankingPool.forward`, `PoolingCursor.get_finished_mask`, `PoolingCursor.getitem`, `PoolingMetadata.build_pooling_cursor`, `PoolingRunner._get_pooling_metadata`, `PoolingRunner.pool`, `GPUModelRunner._pool`, `GPUInputBatch.get_pooling_metadata`

关键源码片段

`vllm/model_executor/models/jina.py`

核心优化点：JinaForRankingPool 新增 get_pooling_updates 声明 requires_token_ids, forward 改为 CPU 端计算 doc/query 索引，仅 H2D 拷贝选中索引，移除 gpu_sync_allowed。

```
class JinaForRankingPool(StepPool):
    def __init__(self, projector: nn.Sequential):
        super().__init__()
        # doc/query 分隔 token id, 用于在 prompt 中定位文档与查询边界
        self.doc_token_id = 151670
        self.query_token_id = 151671
        self.projector = projector

    def get_supported_tasks(self) -> set[PoolingTask]:
        return {"token_embed"}

    def get_pooling_updates(self, task: PoolingTask) -> PoolingParamsUpdate:
        # 声明本池化器需要 prompt token ids; MRV2 会据此在 CPU 上保留该元数据
        return PoolingParamsUpdate(requires_token_ids=True)

    def forward(
        self,
        hidden_states: torch.Tensor,
        pooling_metadata: PoolingMetadata,
    ) -> list[TokenPoolingMethodOutputItem]:
        pooled_data_lst = super().forward(hidden_states, pooling_metadata)
        # 读取 CPU 端 token ids, 避免把完整 batch 拷到 GPU
        prompt_token_ids_cpu = pooling_metadata.get_prompt_token_ids_cpu()

        embeds_list = list[torch.Tensor | None]()
        for data, token_ids_cpu in zip(pooled_data_lst, prompt_token_ids_cpu):
            # 未完成的 chunked prefill 返回 None, 直接透传
            if data is None:
                embeds_list.append(None)
                continue

            # 分隔 token 匹配全部在 CPU 上完成
            doc_match = torch.eq(token_ids_cpu, self.doc_token_id)
            docs_indexes_cpu = torch.where(doc_match)[0]
            query_indexes_cpu = torch.where(
                torch.eq(token_ids_cpu, self.query_token_id)
            )[0]

            # JinaForRanking 模型先拼接文档再拼接查询, 保持该顺序
            indexes_cpu = torch.cat([docs_indexes_cpu, query_indexes_cpu])
            # 只把最终选中的索引异步拷到 GPU, 随后在 hidden states 上投影
            indexes = async_tensor_h2d(
                indexes_cpu, device=data.device, dtype=torch.long
            )
            embeds = self.projector(data[indexes])
            embeds_list.append(embeds)
```

```
return embeds_list
```

vllm/v1/pool/metadata.py

PoolingCursor 新增预计算的 partial_prefill 与 finished_mask, 并提供 get_finished_mask(), 消除 pooling 每步的设备比较与 .tolist() 开销。

```
@dataclass
class PoolingCursor:
    first_token_indices_gpu: torch.Tensor
    last_token_indices_gpu: torch.Tensor
    prompt_lens_cpu: torch.Tensor
    seq_lens_cpu: torch.Tensor
    num_scheduled_tokens_cpu: torch.Tensor
    # 预计算的 CPU 侧状态, 避免每次调用都做设备同步或张量比较
    partial_prefill: bool
    finished_mask: list[bool]

    def __getitem__(self, indices: slice) -> "PoolingCursor":
        prompt_lens_cpu = self.prompt_lens_cpu[indices]
        seq_lens_cpu = self.seq_lens_cpu[indices]
        num_scheduled_tokens_cpu = self.num_scheduled_tokens_cpu[indices]

        # 切片时同步重算 CPU 状态, 保证子 cursor 语义一致
        return PoolingCursor(
            first_token_indices_gpu=self.first_token_indices_gpu[indices],
            last_token_indices_gpu=self.last_token_indices_gpu[indices],
            prompt_lens_cpu=prompt_lens_cpu,
            seq_lens_cpu=seq_lens_cpu,
            num_scheduled_tokens_cpu=num_scheduled_tokens_cpu,
            partial_prefill=not torch.equal(prompt_lens_cpu, num_scheduled_tokens_cpu),
            finished_mask=torch.eq(prompt_lens_cpu, seq_lens_cpu).tolist(),
        )

    def is_partial_prefill(self) -> bool:
        return self.partial_prefill

    def is_finished(self) -> torch.Tensor:
        return self.prompt_lens_cpu == self.seq_lens_cpu

    def get_finished_mask(self) -> list[bool]:
        return self.finished_mask
```

评论区精华

核心讨论围绕“是否值得为 GPU token IDs 保留单独路径”展开: yewentao256 在 vllm/pooling_params.py 的评论中提出能否删除 requires_token_ids_gpu 属性, 因为 v1 到 v2 的设计方向是移除 consistent batch; 随后又建议“在 CPU 上计算 query_indexes, 只 H2D 拷贝最终的 indexes”, 作者回复采纳并实现。noooop 在 metadata.py 上建议

partial_prefill 与 finished_mask 不应为 None 以简化比较，作者将其收紧为始终具体值并在切片时重算。另外 nooooop 质疑 pin_memory 条件修改 (PIN_MEMORY and device.type != "cpu") 的必要性，作者解释是为了 CPU-only 单测便利并同意 revert，最终提交 "Restorepinned pooling cursor allocation" 恢复原状。

- PoolingParams.requires_token_ids_gpu 是否应保留 (design): 采纳: 不再为 GPU token IDs 保留单独需求标志，统一由 CPU 侧索引计算替代。
- doc/query 索引计算放在 CPU 还是 GPU (performance): 采纳: CPU 完成匹配与拼接，仅异步拷贝选中索引。
- PoolingCursor 的 partial_prefill/finished_mask 是否允许 None (design): 采纳: 去除 None 分支，构建时预计算、切片时重算。
- pin_memory 条件修改是否必要 (question): revert: 保持原有 pin_memory 行为。

风险与影响

- 风险: 契约风险: PoolingMetadata.prompt_token_ids 现在恒为 None，任何仍依赖 model-device token IDs 的池化方法（非 CPU 索引计算类）会直接拿到 None 而报错；PR 通过 requires_token_ids / 移除 GPU 路径来约束，但 vllm/model_executor/layers/pooler/tokwise/methods.py 与 common.py 的小改动说明该契约仍在迁移期，后续新增池化方法容易踩坑。回归风险: JinaForRankingPool.forward 移除 gpu_sync_allowed 后，CPU 索引计算与 GPU 投影之间的同步依赖 async_tensor_h2d 的异步语义，chunked prefill 未完成时 data 为 None 虽有保护，但若设备驱动或后端变更可能引入隐式同步差异，需保留数值精度测试。性能风险: PR 提供的 benchmark 中 concurrency 32 时 p99 从 568.8 ms 升到 633.8 ms (约 +11%)，PR 未解释原因，高并发下可能出现尾部延迟波动。兼容性: CPU-only 路径下 prompt_token_ids_cpu 不再启用 pin_memory，该写法最终被 revert，实际 GPU 行为不变，但 PoolingCursor 构造签名新增两个必填字段，所有直接构造 PoolingCursor 的代码（包括测试）必须同步更新。
- 影响: 用户影响: 启用 MRV2 的 BGE-M3 / Jina 类 embedding 与 ranking 服务在 sync 模式低并发下吞吐提升约 3% (concurrency 16 时 +3.13%，concurrency 32 时约 +1%)；SPLADE 稀疏输出与 dense 输出数值误差均在 5e-4 容差内，行为保持一致。系统影响: PoolingMetadata 的数据契约发生实质变化 (prompt_token_ids 恒为 None)，影响 pooling_runner、gpu_input_batch、gpu_model_runner 三个执行路径；PoolingCursor 的 CPU 状态缓存让 finished_mask 的计算从 per-step 变成 per-cursor，减少重复设备比较。团队影响: 为 MRV2 pooling 后续彻底删除 prompt_token_ids 字段铺路，并沉淀了“CPU 上算索引、只 H2D 结果”的优化模式；review 中确认了 MRV2 无 consistent batch 的设计方向。
- 风险标记: 池化元数据契约变更，核心执行路径修改，数值精度依赖验证，高并发 p99 抖动

关联脉络

- PR #53306 [Model Runner V2] Reserve CUDA graph memory: 同为 MRV2 演进系列，均修改 vllm/v1/worker/gpu/model_runner.py 与 cudagraph 路径，体现 MRV2 执行路径的持续优化方向。