# PR #35070 完整报告

sgl-project/sglang

[PD] Avoid unused PREBUILT prompt tensor transfer

合并时间：2026-08-17 16:48

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35070

## 执行摘要

- 一句话：避免 PREBUILT 批次的未使用 prompt 张量传输，提升 PD 分离性能。
- 推荐动作：该 PR 值得精读，尤其是 prepare_for_prebuilt 的实现细节和注释，反映了对 PD 分离数据流中未使用数据传输的洞察。设计决策值得关注：通过将 input_ids 置为 None 并依赖 relay 重建输入，避免了多余的 GPU 传输，同时保持了元数据的完整性。建议后续在 PD 分离相关优化中参考此模式。

## 功能与动机

在 PD 分离场景中，解码端的 PREBUILT 批次永远不会进入模型前向，但 prepare_for_prebuilt 仍会读取每个转移的 prompt、扁平化并复制到 CUDA input_ids 张量。第一个解码步骤稍后从 relay 元数据重建其输入，因此这次 prompt 传输是未使用的，对于长上下文请求而言成本高昂。PR body 中明确说明了这一动机，并给出了移除未使用传输后每个目标步节省约 1.689 ms 的性能数据。

## 实现拆解

1. 修改 prepare_for_prebuilt 方法：在 python/sglang/srt/disaggregation/decode_schedule_batch_mixin.py 中，移除了原先通过 get_fill_ids() 读取并扁平化所有 prompt token 的代码，并删除 array 导入。现在 extend_num_tokens 直接从 req.extend_range.length 的总和计算（total_size），避免了不必要的列表求和。
2. 将 input_ids 置为 None：原先将扁平化的 token 数组转换为 torch 张量并赋给 self.input_ids，现在直接设置 self.input_ids = None，因为第一个解码输入和投机性扩展输入由 process_prebuilt 通过 FutureMap 种子化，在 merge/forward-entry 时从 relay 重建。此改动保留了 req_pool_indices、seq_lens、out_cache_loc 等标量元数据的填充逻辑，不影响缓存位置、序列长度、请求池索引、采样元数据及 relay 行为。
3. 新增单元测试：在 test/registered/unit/disaggregation/test_disaggregation_wire.py 中添加了 test_prebuilt_skips_unused_prompt_tensor 测试用例，构造一个模拟 PREBUILT 请求和批次，断言 batch.input_ids 为 None、batch.extend_num_tokens 为 3、out_cache_loc 正确，并验证 get_fill_ids 未被调用（通过 side_effect=AssertionError 确保），以捕获回归。

关键文件：

- python/sglang/srt/disaggregation/decode_schedule_batch_mixin.py（模块 调度器；类别 source；类型 dependency-wiring；符号 prepare_for_prebuilt）：核心修改文件：在 prepare_for_prebuilt 中移除未使用的 prompt 张量传输，将 input_ids 置为 None，并调整 extend_num_tokens 推导，是性能优化的关键实现。
- test/registered/unit/disaggregation/test_disaggregation_wire.py（模块 分离测试；类别 test；类型 test-coverage；符号 test_prebuilt_skips_unused_prompt_tensor）：新增回归测试 test_prebuilt_skips_unused_prompt_tensor，验证 PREBUILT 批次不再读取 prompt，确保 input_ids 为 None 且元数据正确。

关键符号：prepare_for_prebuilt

## 关键源码片段

### python/sglang/srt/disaggregation/decode_schedule_batch_mixin.py

核心修改文件：在 prepare_for_prebuilt 中移除未使用的 prompt 张量传输，将 input_ids 置为 None，并调整 extend_num_tokens 推导，是性能优化的关键实现。

```
# python/sglang/srt/disaggregation/decode_schedule_batch_mixin.py
# 已精简为 prepare_for_prebuilt 的核心片段，用于说明 PREBUILT 批次如何避免 prompt 传输。
def prepare_for_prebuilt(self: ScheduleBatch):
    """Prepare a prebuilt extend by populating metadata."""
    self.forward_mode = ForwardMode.PREBUILT
    reqs = self.reqs
    seq_lens = []
    pre_lens = []
    req_pool_indices = []

    # 预计算总大小，extend_num_tokens 直接从 extend ranges 推导，无需读取 prompt 内容
    total_size = sum(req.extend_range.length for req in reqs)
    extend_num_tokens = total_size
    out_cache_loc = torch.empty(total_size, dtype=torch.int64, device=self.device)

    offset = 0
    for i, req in enumerate(reqs):
        req_pool_indices.append(req.req_pool_idx)
        pre_len = len(req.prefix_indices)
        # 从 token pool 中拷贝缓存位置到 out_cache_loc，这是必要的元数据
        chunk = self.req_to_token_pool.req_to_token[req.req_pool_idx][
            pre_len : pre_len + req.extend_range.length
        ]
        assert (
            offset + req.extend_range.length <= total_size
        ), f"Exceeds total size: offset={offset}, req.extend_range.length={req.extend_range.length},
        total_size={total_size}"
        out_cache_loc[offset : offset + req.extend_range.length] = chunk
        offset += req.extend_range.length

        seq_len = len(req.origin_input_ids) + max(0, len(req.output_ids) - 1)
```

```
        seq_lens.append(seq_len)
        if len(req.output_ids) == 0:
            assert (
                seq_len - pre_len == req.extend_range.length
            ), f"seq_len={seq_len}, pre_len={pre_len}, req.extend_range.length={req.extend_range.
            length}"

        if not req.retracted_stain:
            # 避免重复计数：already_computed 由 prefill 报告的 cached_tokens 种子化，
            # 若解码端前缀比 prefill 报告短，不能从 cached_tokens 中减去
            delta = max(0, pre_len - req.already_computed)
            req.cached_tokens += delta
            req.cached_tokens_device += delta
            req.already_computed = seq_len
        req.is_retracted = False
        if getattr(req, "pd_rebootstrap_in_progress", False):
            req.pd_rebootstrap_in_progress = False
        pre_lens.append(pre_len)

    # 设置字段：input_ids 置为 None，因为 PREBUILT 批次不会进入模型前向，
    # 第一个解码输入由 process_prebuilt 通过 FutureMap 重建
    self.input_ids = None
    self.req_pool_indices = torch.tensor(req_pool_indices, dtype=torch.int64, device=self.device)
    self.req_pool_indices_cpu = torch.tensor(req_pool_indices, dtype=torch.int64)
    self.seq_lens = torch.tensor(seq_lens, dtype=torch.int64, device=self.device)
    self.seq_lens_cpu = torch.tensor(seq_lens, dtype=torch.int64)
    self.orig_seq_lens = torch.tensor(
        seq_lens, dtype=torch.int32, device=self.device
    )
    self.out_cache_loc = out_cache_loc
    self.seq_lens_sum = sum(seq_lens)

    if self.return_logprob:
        self.top_logprobs_nums = [r.logprob.top_logprobs_num for r in reqs]
        self.token_ids_logprobs = [r.logprob.token_ids_logprob for r in reqs]

    self.extend_num_tokens = extend_num_tokens
    self.prefix_lens = [len(r.prefix_indices) for r in reqs]
    self.extend_lens = [r.extend_range.length for r in reqs]
    self.extend_logprob_start_lens = None
    self.extend_input_logprob_token_ids = None
    self.multimodal_inputs = [r.multimodal_inputs for r in reqs]

    # 构建采样信息
    self.sampling_info = SamplingBatchInfo.from_schedule_batch(
        self, self.model_config.vocab_size
    )
```

test/registered/unit/disaggregation/test_disaggregation_wire.py

新增回归测试 test_prebuilt_skips_unused_prompt_tensor，验证 PREBUILT 批次不再读取 prompt，确保 input_ids 为 None 且元数据正确。

```
# test/registered/unit/disaggregation/test_disaggregation_wire.py
# 新增的回归测试，验证 PREBUILT 批次不再读取 prompt。
def test_prebuilt_skips_unused_prompt_tensor(self):
    # 构造一个 PREBUILT 请求，get_fill_ids 被 mock 为抛出 AssertionError，
    # 用于确保 prompt 不会被读取
    req = SimpleNamespace(
        req_pool_idx=0,
        prefix_indices=[0, 1],
        extend_range=SimpleNamespace(length=3),
        origin_input_ids=[0, 1, 2, 3, 4],
        output_ids=[],
        retracted_stain=True,
        is_retracted=True,
        multimodal_inputs=None,
        get_fill_ids=Mock(side_effect=AssertionError("prompt should not be read")),
    )
    batch = SimpleNamespace(
        reqs=[req],
        device="cpu",
        req_to_token_pool=SimpleNamespace(
            req_to_token=torch.arange(5, dtype=torch.int64).reshape(1, 5)
        ),
        return_logprob=False,
        model_config=SimpleNamespace(vocab_size=32),
    )

    with patch(
        "sglang.srt.disaggregation.decode_schedule_batch_mixin."
        "SamplingBatchInfo.from_schedule_batch",
        return_value=Mock(),
    ):
        ScheduleBatchDisaggregationDecodeMixin.prepare_for_prebuilt(batch)

    # 验证 input_ids 为 None，extend_num_tokens 正确，且 get_fill_ids 未被调用
    self.assertIsNone(batch.input_ids)
    self.assertEqual(batch.extend_num_tokens, 3)
    self.assertTrue(torch.equal(batch.out_cache_loc, torch.tensor([2, 3, 4])))
    req.get_fill_ids.assert_not_called()
```

## 评论区精华

Review 审核中，ShangmingCai 批准了 PR（LGTM），没有针对实现细节的公开讨论。CI 重跑过程中，SGLang机器人执行了/rerun-groupdisaggregation，相关测试全部通过，无争议。

- Review 审核通过 (question): PR 被批准，无未解决问题。

## 风险与影响

- 风险：主要风险在于将 input_ids 置为 None 后，任何依赖 prepare_for_prebuilt 之后 input_ids 非空的代码路径可能导致崩溃。但根据 PR 描述，PREBUILT 批次不会进入模型前向，且第一个 *decode* 步骤的输入由 process_prebuilt 通过 FutureMap 重建，因此该风险被控制在合理范围。需要确认所有 PREBUILT 路径都不会在 prepare_for_prebuilt 之后直接访问 input_ids。此外，*extend_num_tokens* 的推导方式改变，需要确保下游依赖该值的逻辑（如调度或流水线）行为一致。由于改动集中在 PD 分离的特定环节，回归风险较低，同时新增了针对性测试，风险可控。
- 影响：影响范围限于 PD 分离部署中的 PREBUILT 批次处理，主要受益场景为长上下文分离请求，可显著降低 CPU-GPU 之间的数据拷贝开销和显存占用。实测数据表明，在 24 GB300 GPU、并发 896 的生产负载下，*decode* TPS 提升约 12.7%，每个目标步 GPU 可见时间线节省约 1.689 ms。对非分离场景无影响。对团队而言，这是一个低风险高收益的性能优化，可作为 PD 分离性能调优的参考案例。
- 风险标记：核心路径变更，缺少测试覆盖

## 关联脉络

- PR #36219 [Performance] Tune FlashInfer EXTEND for DP prefill: 涉及 PD 分离中 prefill 性能优化，与本 PR 同属性能改进方向，可能相关。