

PR #53867 完整报告

vllm-project/vllm

[Feature][PCP] Support decode-only FULL CUDA graphs

合并时间: 2026-09-15 16:27

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53867>

执行摘要

- 一句话: PCP 支持 decode-only FULL CUDA 图并统一捕获输入准备
- 推荐动作: 值得精读, 但重点看三处: 一是 `partition_batch` 中 `num_reqs_after_padding` 的切片语义 (真实行 vs padding 行的零值约定), 它决定 FULL 图重放的 shape 契约; 二是 `PCPManager.prepare_inputs_to_capture` 与 `get_dummy_block_tables` 的持久 buffer / rank-local 表策略, 这是图捕获地址稳定的前提; 三是 `validate_config` 如何用 `separate_routine()` 表达「decode 可以 FULL、prefill 不行」这一约束。建议同时阅读 [piscesk/kk/vllm#8](#) 的重构提案, 理解 maintainer 为何要求把 `InputBatch` 贯穿捕获准备。若计划在生产启用 `FULL_DECODE_ONLY`, 建议先自行复现多步复用 buffer、请求数波动场景下的 `seq_lens padding` 行为。

功能与动机

PR body 明确目标是让 prefill context parallelism (PCP) 部署能用上 decode-only 的 FULL CUDA graph, 从而降低 decode 阶段的图启动与调度开销。最初的质疑来自 maintainer LucasWilkinson: "Im not sure we want this?! I dont it makes sense to have a PCP deployment doing decodes (i.e. PCP is primarily for P/D). Is there a use case im missing?"; 作者以 Ascend 上 GLM-5.2 / DeepSeek-V4 的混合部署经验回应, 指出 PCP 不仅用于 P/D 分离场景, 混合部署下 prefill 提速明显、平均 TPOT 也更好, 并引用 PR#52162 (PCP x DP 聚合部署) 的数据; Lucas 随后认可 "with that context this makes more sense"。

实现拆解

1. 配置校验: 从「一律禁止 FULL」改为「只禁止纯 FULL」

`vllm/v1/worker/gpu/pcp_manager.py` 的 `PCPManager.validate_config` 中, 原先只要 `cudagraph_mode.has_full_cudagraphs()` 就抛 `NotImplementedError`; 现在追加 `and not cudagraph_mode.separate_routine()`, 即只有 FULL 与 PIECEWISE 共用 routine 的纯 FULL 模式被拒绝, `FULL_DECODE_ONLY` / `FULL_AND_PIECEWISE` 放行。原因是 PCP 的 prefill 走单独 dispatch 的执行路径, 只要 decode routine 被拆出来就能安全捕获全图; `sparse MLA + DCP=1` 仍需 `cudagraph_mode=NONE`, 这部分约束未变。

2. batch 分区感知请求维 padding

- 新增静态方法 `PCPManager._resolve_num_reqs_after_padding`: `padded_num_reqs` is None 时返回真实 `num_local_reqs` (PIECEWISE / eager 行为不变); 非 None 时若 `input_batch.has_prefill` 直接抛 `RuntimeError("PCP FULL graphs require a decode-only batch.")`, 并断言图容量不小于 rank-local batch。
- `partition_batch` 新增 `padded_num_reqs` 形参, 请求维张量 (`seq_lens`、`query_start_loc`、`local_is_prefilling_np`、`seq_lens_cpu_upper_bound_np`、`dcp_local_seq_lens_cpu_upper_bound`) 统一按 `num_reqs_after_padding` 分配, padding 行留零; `replace(...)` 写回时 `num_reqs_after_padding` 与图捕获 shape 对齐, `num_reqs` 仍为真实 local 请求数。
- 模块级函数 `maybe_partition_pcp_batch` 的入参从 `padded_num_tokens` 换成 `batch_desc: BatchExecutionDescriptor`, 内部集中推导: 非 NONE 模式给 token 维 padding, FULL 模式再给请求维 padding。

3. 持久输入与 rank-local block table

- 新增 `PCPManager.prepare_inputs_to_capture`: 用 `replace` 返回新的 `InputBatch`, 但 `input_ids` / `positions` / `is_padding` / `query_start_loc` / `seq_lens` 全部指向 `PCPManager.input_buffers` 的切片副本, 保证捕获时记录的地址与重放时写入的地址一致。
- 新增 `PCPManager.get_dummy_block_tables`: 返回 `self._local_block_tables` 的零值视图, 避免 dummy 阶段写脏全局 block table。

4. dummy / capture 路径线性化

- `vllm/v1/worker/gpu/cudagraph_utils.py::prepare_inputs_to_capture` 改为调用 `pcp_manager.prepare_inputs_to_capture(input_batch)`, 并用 `block_table_provider = pcp_manager or block_tables` 同时提供 dummy block table 与 dummy slot mapping, 消除「输入来自 PCP、metadata 来自全局表」的不一致。
- `vllm/v1/worker/gpu/model_runner.py` 中 `execute_model` 的 dummy run 分支重排为: 先落持久 buffer → 再判 `skip_attn_for_dummy_run` → 再 `prepare_dummy_attn` → 最后 `set_dummy_context`; `prepare_dummy_attn` 也改用 `self.pcp_manager or self.block_tables` 作为 provider。
- `vllm/v1/worker/gpu/input_batch.py::set_dummy_context` 新增可选参数 `input_block_tables: Sequence[torch.Tensor] | None`, 未显式传入时回落全局表, 调用方可以把 PCP-local 表传进来而不污染全局状态。

5. 测试与验证配套

`tests/v1/worker/test_gpu_pcp_manager.py` 新增 `_make_config` / `_make_capture_manager` 两个 helper, 以及 `test_validate_config_accepts_decode_only_full_graphs`、`test_validate_config_rejects_full_graph_for_prefills`、`test_partition_padding_is_derived_from_batch_descriptor`、`test_capture_uses_pcp_persistent_inputs`、`test_dummy_context_updates_pcp_local_block_tables` 五个用例; 原有 DCP metadata 用例同步更新为 padding 后长度 (`[17, 25, 0, 0]`)。作者另在 H100 x2 / GLM-4.7-Flash / TP=1、PCP=2、EP=2 上完成 GSM8K 全量 1319 题的四模式对比。

关键文件：

- `vllm/v1/worker/gpu/pcp_manager.py`（模块 PCP 管理器；类别 source；类型 dependency-wiring；符号 `PCPManager.validate_config`, `_resolve_num_reqs_after_padding`, `partition_batch`, `prepare_inputs_to_capture`）：本 PR 的核心：放开 FULL 图配置校验、新增请求维 padding 解析与持久捕获输入，并把 `partition` 入口改为接收 `BatchExecutionDescriptor`。
- `vllm/v1/worker/gpu/model_runner.py`（模块 模型执行器；类别 source；类型 data-contract；符号 `execute_model`, `prepare_dummy_attn`, `prepare_inputs`）：`dummy run` 与捕获路径的入口重排：先把 `dummy batch` 落持久 buffer，再准备 `attention metadata`，并把 PCP-local block table 传给 `set_dummy_context`。
- `tests/v1/worker/test_gpu_pcp_manager.py`（模块 PCP 测试；类别 test；类型 test-coverage；符号 `_make_config`, `_make_capture_manager`, `test_validate_config_accepts_decode_only_full_graphs`, `test_validate_config_rejects_full_graph_for_prefills`）：新增配置校验、padding 推导、持久输入与 `dummy block table` 的用例，并按 `codex review` 要求把独立测试文件合并回该 suite。
- `vllm/v1/worker/gpu/cudagraph_utils.py`（模块 图捕获工具；类别 source；类型 core-logic；符号 `prepare_inputs_to_capture`）：捕获入口改为复用 `PCPManager` 的持久输入与 `block-table/slot-mapping provider`，是 full graph 捕获能成立的关键衔接点。
- `vllm/v1/worker/gpu/input_batch.py`（模块 输入批次；类别 source；类型 data-contract；符号 `set_dummy_context`）：`set_dummy_context` 新增 `input_block_tables` 参数，使 `dummy context` 可以写入 PCP-local 表而不触碰全局 `block table`。

关键符号：`PCPManager.validate_config`, `PCPManager._resolve_num_reqs_after_padding`, `PCPManager.partition_batch`, `PCPManager.prepare_inputs_to_capture`, `PCPManager.get_dummy_block_tables`, `maybe_partition_pcp_batch`, `prepare_dummy_attn`, `prepare_inputs_to_capture`, `set_dummy_context`

关键源码片段

`vllm/v1/worker/gpu/model_runner.py`

`dummy run` 与捕获路径的入口重排：先把 `dummy batch` 落持久 buffer，再准备 `attention metadata`，并把 PCP-local block table 传给 `set_dummy_context`。

```
# vllm/v1/worker/gpu/model_runner.py (节选: dummy / capture 分支的线性化)
```

```
def prepare_dummy_attn(self, input_batch, valid_state_slots=False):
    # PCP 场景下 block table 的提供者换成 PCPManager: 它返回 rank-local 表;
    # 非 PCP 场景仍是原来的 self.block_tables, 行为不变。
    block_table_provider = self.pcp_manager or self.block_tables
    block_tables = block_table_provider.get_dummy_block_tables(input_batch.num_reqs)
    # ... (valid_state_slots 分支与 dummy slot mapping 略) ...
    return block_tables, slot_mappings
```

```
# prepare_inputs 内的分区调用: padding 维度推导下沉到 PCP 模块
```

```

input_batch = pcp.maybe_partition_pcp_batch(
    self.pcp_manager,
    input_batch,
    batch_desc, # 原来只传 padded_num_tokens=batch_desc.num_tokens
)

# execute_model 内的 dummy run 分支
else:
    # No actual tokens to run. A dummy run for DP or memory profiling.
    dummy_num_reqs = batch_desc.num_reqs or num_reqs
    input_batch = InputBatch.make_dummy(
        dummy_num_reqs,
        batch_desc.num_tokens,
        self.input_buffers,
        max_query_len=batch_desc.max_query_len,
    )
    # 先落到 PCP 持久 buffer, 再决定是否准备 attention metadata:
    # 捕获与重放必须看到同一组指针。
    if self.pcp_manager is not None:
        input_batch = self.pcp_manager.prepare_inputs_to_capture(input_batch)
    if skip_attn_for_dummy_run:
        assert batch_desc.cg_mode != CUDAGraphMode.FULL, (
            "Attention metadata must be prepared for dummy runs when using "
            "FULL cudagraph mode."
        )
        block_tables = None
        slot_mappings = None
    else:
        block_tables, slot_mappings = self.prepare_dummy_attn(
            input_batch, valid_dummy_state_slots
        )
    if not skip_attn_for_dummy_run and context_len:
        assert block_tables is not None
        # 关键: 把 PCP-local 的 block table 交给 set_dummy_context,
        # 否则它会去改全局 self.block_tables.input_block_tables。
        set_dummy_context(
            input_batch,
            self.block_tables,
            context_len,
            self.kv_cache_config.num_blocks,
            self.max_model_len,
            input_block_tables=block_tables,
        )

```

vllm/v1/worker/gpu/cudagraph_utils.py

捕获入口改为复用 PCPManager 的持久输入与 block-table/slot-mapping provider, 是 full graph 捕获能成立的关键衔接点。

vllm/v1/worker/gpu/cudagraph_utils.py (节选: 捕获时统一 provider)

```

def prepare_inputs_to_capture(
    num_reqs: int,
    num_tokens: int,
    model_state: ModelState,
    input_buffers: InputBuffers,
    block_tables: BlockTables,
    attn_groups: list[list[AttentionGroup]],
    kv_cache_config: KVCacheConfig,
    full_cudagraph: bool,
    max_query_len: int | None = None,
    pcp_manager: "PCPManager | None" = None,
) -> AttentionState:
    input_batch = InputBatch.make_dummy(
        num_reqs, num_tokens, input_buffers, max_query_len=max_query_len
    )
    if pcp_manager is not None:
        # 捕获用的 dummy batch 也换到 PCP 持久 buffer 上，与运行期共用同一组地址。
        input_batch = pcp_manager.prepare_inputs_to_capture(input_batch)

    # block table 与 slot mapping 共用同一个 provider，避免出现
    # 「token 输入来自 PCP、attention metadata 来自全局表」这类不一致。
    block_table_provider = pcp_manager or block_tables
    input_block_tables = block_table_provider.get_dummy_block_tables(num_reqs)
    slot_mappings = block_table_provider.get_dummy_slot_mappings(num_tokens)
    slot_mappings_by_layer = build_slot_mappings_by_layer(
        slot_mappings, kv_cache_config
    )
    # ... (DCP local seq lens 推导与 AttentionState 组装略) ...

```

评论区精华

1) 用例合理性之争：LucasWilkinson 首轮 review 直接质疑 PCP 跑 decode 的意义，review 状态为 DISMISSED；作者以混合部署的吞吐与 TPOT 数据回应，并引用 PR#52162 (PCP x DP 聚合部署)，Lucas 承认 "oh sorry i missed #52162 ... with that context this makes more sense, thanks!"，最终 APPROVED。2) 重构方向之争：Lucas 提议 "thoughts on something like pisceskkk/vllm#8 ?"，即把 InputBatch 贯穿捕获准备、用统一的 block-table provider 共享 dummy attention 准备；作者采纳该方案并落地为 commit 997afd8 (Co-authored-by Lucas Wilkinson)，把原本分散的 slot mapping / block table 选择收敛为 `pcp_manager or block_tables`。3) 测试组织：chatgpt-codex-connector[bot] 给出 P1 建议，要求把新增的 `tests/v1/worker/test_gpu_pcp_full_decode_only.py` 合并进既有 `test_gpu_pcp_manager.py` (引用 AGENTS.md:L88-L89)；作者随后以 "test(pcp): consolidate full graph coverage" 合并，最终 diff 中不再存在独立测试文件。4) 过程摩擦：mergify 多次提示合并冲突、pre-commit 失败，作者反复 `/ci run` 与 rebase，最终在 Buildkite #88985 上重试通过；这解释了 16 个 commit 中大量 "Merge upstream main" 的来源。

- PCP 部署是否需要 decode 与 FULL 图（用例合理性）（design）：用例被接受，review 从 DISMISSED 转为 APPROVED ("LGTM, thanks for the contribution!")。
- 把 InputBatch 贯穿捕获准备的重构方案（design）：以 maintainer 方案为准，slot mapping / block table 选择统一为 pcp_manager or block_tables（见 cudagraph_utils.prepare_inputs_to_capture 与 model_runner.prepare_dummy_attn）。
- 测试应合并进既有 PCP manager suite（testing）：作者以 "test(pcp): consolidate full graph coverage" 合并用例，最终 PR 只修改 test_gpu_pcp_manager.py。
- FULL 图限制的边界：只放行 decode-only routine（design）：FULL_DECODE_ONLY / FULL_AND_PIECEWISE 放行，纯 FULL 仍被拒绝并给出明确错误提示；sparse MLA + DCP=1 依旧要求 NONE。
- 合并冲突与 CI 反复重跑（other）：经多轮 rebase（16 个 commit 中大部分是 Merge upstream main 与 mergify 合并）后 Buildkite CI #88985 触发并重试通过。

风险与影响

- 风险：1) 核心捕获路径回归：partition_batch 的请求维切片直接决定 FULL 图重放时的 shape 与 KV 读写行；seq_lens[num_local_reqs:] 只在 num_local_tokens == 0 的 else 分支显式清零，真实 batch 非空时 padding 行的零值依赖持久 buffer 不被写脏的不变量。单测通过新建 buffer 覆盖到的是零值初始态，多步复用同一 buffer、且 rank-local 请求数先大后小的场景缺少直接断言，建议补一条复用 buffer 的回归用例。2) 校验强度：_resolve_num_reqs_after_padding 用 assert 做容量检查（padded_num_reqs >= num_local_reqs），在 python -O 下会被剥离；has_prefill 检查用的是 RuntimeError，两者强度不一致。3) 配置组合复杂：validate_config 现在同时约束 sparse MLA、DCP backend、speculative（DSpark / single-module MTP）、separate_routine()，任一组合的语义都靠抛错兜底；FULL_AND_PIECEWISE 与 FULL_DECODE_ONLY 都需要 decode-only batch 这一隐含前提，若上游调度把混合 batch 送进 decode routine，会在运行期抛 RuntimeError 而非启动期拒绝。4) 正确性证据来源：仓库内仅有 CPU 单测与一个 skipif(not torch.cuda.is_available()) 的用例，四模式准确率对齐（偏差 0.61 pp 内）完全依赖作者手工 GSM8K 验证，缺少 CI 化端到端覆盖。5) 跨模块契约变更：set_dummy_context 新增 input_block_tables 参数、maybe_partition_pcp_batch 换签名、prepare_inputs_to_capture（cudagraph_utils 与 PCPManager 同名不同层）三处接口同时变动，其他调用方若按旧签名调用会静默走默认分支。
- 影响：对用户：启用 MRV2 + PCP 的部署现在可以选择 FULL_DECODE_ONLY / FULL_AND_PIECEWISE，decode 阶段获得全图重放收益，同时保留 prefill 走 PIECEWISE/eager 的路径；不配 PCP 的路径行为不变。对系统：改动集中在 MRV2 的 GPU worker（vllm/v1/worker/gpu/）四个源码文件，input_batch.set_dummy_context 与 cudagraph_utils.prepare_inputs_to_capture 属于被多个调用点复用的公共工具函数，影响面超出 PCP 本身，但通过 input_block_tables=None 的回落与 pcp_manager or block_tables 的默认分支保证非 PCP 场景等价。对团队：这是 MRV2 PCP 能力矩阵的一次扩展（此前 PCP 只允许 PIECEWISE 或 NONE），也确立了一种「capture 输入必须来自持久 buffer」的实现范式，后续 PCP/HiSparse 方向的图支持可以复用 prepare_inputs_to_capture 与 provider 抽象。

- 风险标记：核心捕获路径变更，FULL 图与 PCP prefill 路径互斥，持久 buffer 零值不变量依赖，容量校验使用 assert，端到端正确性依赖手工 GSM8K 验证

关联脉络

- PR #52162 PCP x DP 聚合部署（讨论中引用，标题依据评论措辞）：Lucas 在讨论中确认该 PR 让 PCP 在混合部署中跑 decode 变得合理，是本 PR 用例成立的前提；作者也用它作为混合部署吞吐 /TPOT 的证据。
- PR #56825 [Bugfix][SparseMLA] Fix piecewise cudagraph capture crash in index group: 同样处于 PCP / sparse MLA 的 cudagraph capture 路径（index_group.py、forward_context.py），与本 PR 修改的捕获期 attention metadata 准备高度相邻。
- PR #56382 [Bugfix] Carry over queued work when materializing the dedicated stream: 涉及专用流与 cudagraph 捕获 / 重放的顺序依赖，本 PR 的 FULL 图持久 buffer 策略对跨流写入顺序敏感，属于同一类图重放正确性问题。