PR #34599 完整报告

sgl-project/sglang

[diffusion] Optimize Pi0.5 inference and bounded graph serving

合并时间：2026-08-29 14:47

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34599

执行摘要

- 一句话：优化 Pi0.5 推理并引入有界 CUDA 图缓存
- 推荐动作：该 PR 值得精读，尤其是有界 LRU 图缓存的 CUDAGraph.reset() 生命周期管理、exact 与 bucket 两种模式的回退路径，以及 MergedReplicatedLinear 的 shard 加载设计。关注点：BF16 舍入差异与容量回退带来的服务质量抖动。可作为 VLA 推理优化的参考实现。

功能与动机

PR body 明确指出 H200 eager trace 显示 Pi0.5 推理是 launch-bound：单请求启动 9,995 个 CUDA 内核，CUDA 工作耗时约 50.059 ms（请求约 190 ms）。非 TP 的 PiGemma 层分别发出 Q/K/V 与 gate/up 投影，每个 action step 重建 mask、position 与 sinusoidal scaling；混合 prompt 长度导致 action 图无界驻留；masked LIBERO prefix 捕获时还会执行 device-to-host 的 .item()，且 masked prefix 无法使用 action 图。因此需要合并投影减少内核数、消除重复布局计算，并为图缓存设置容量上限，同时将该 PR 与 #34588 合并。

实现拆解

1. 权重打包与加载适配：在 python/sglang/multimodal_gen/runtime/layers/linear.py 新增 MergedReplicatedLinear，作为非 TP 的合并线性层；PiGemmaMLP 与 PiGemmaAttention 的非 TP 分支改用它合并 gate/up、Q/K/V，forward 只发一次 GEMM。weight_loader 支持 q/k/v 分片映射、output_dim narrow 与标量转数组，projection_dtype 也统一指向合并权重。
2. 布局与缩放计算上提：pi05_core.py 的 prepare_denoise_layout 将每个 denoise step 重复构建的 mask、position 与 sinusoidal scaling 提升为一次计算并在各 step 间复用，减少 eager 路径的重复内核启动。
3. 有界 CUDA 图缓存：cuda_graph.py 新增 _BoundedCaptureCache（OrderedDict LRU）与 VLAGraphCacheInfo 统计信息；prefix/action runner 分别受 prefix_cuda_graph_max_entries 与 action_cuda_graph_max_entries 限制，容量满时 exact 模式回退 eager；替换或淘汰条目前调用 CUDAGraph.reset() 释放资源；VLADenoiseGraphSignature 增加 prefix_full_attention 区分 masked 与 full-attention 布局。
4. Prompt 分桶（opt-in）：新增 prompt_bucketing.py，提供 effective_token_length、select_prompt_token_bucket、bucket_prompt_tokens；配置启用 prompt_token_buckets 后右填充并正确掩码，使 prefix 与 action 图共享稳定 shape；

action 图重放前刷新可变 prefix K/V 与 mask。

5. 配置、元数据与测试配套：pi05.py 的 __post_init__ 与 _validate_cuda_graph_config 校验分桶必须为正整数、严格递增且不超过 max_token_len；protocol.py 与 vla.py 阶段暴露有效图可用性元数据；新增 / 扩展 test_pi05_runtime_helpers.py、test_parallel_linear_weight_loading.py、test_pi05_action_api.py，共 53 个用例覆盖缓存容量、LRU 驱逐、回退、分片加载与 metadata。

关键文件：

- python/sglang/multimodal_gen/runtime/vla/cuda_graph.py（模块 图缓存；类别 source；类型 core-logic；符号 VLAGraphCacheInfo, _BoundedCaptureCache, init, _release）：核心改动：新增有界 LRU 图缓存 _BoundedCaptureCache 与 VLAGraphCacheInfo，限制 prefix/action CUDA 图驻留并支持精确回退，图签名区分 masked/full-attention 布局。
- python/sglang/multimodal_gen/runtime/vla/prompt_bucketing.py（模块 分桶；类别 source；类型 dependency-wiring；符号 effective_token_length, select_prompt_token_bucket, bucket_prompt_tokens）：新增 prompt 分桶工具，实现有效长度计算、桶选择与右填充掩码，是 opt-in 分桶模式的基础。
- python/sglang/multimodal_gen/configs/pipeline_configs/pi05.py（模块 配置；类别 source；类型 core-logic；符号 post_init, _validate_cuda_graph_config, check_pipeline_config, prefix_cuda_graph_available）：新增 prompt_token_buckets 与 action_cuda_graph_max_entries 配置及校验，决定图模式可用性。
- python/sglang/multimodal_gen/runtime/models/vlas/pi05_core.py（模块 模型层；类别 source；类型 data-contract；符号 prepare_denoise_layout）：非 TP 的 QKV 与 gate/up 投影改用 MergedReplicatedLinear，合并 GEMM 并上提 denoise 布局计算。
- python/sglang/multimodal_gen/runtime/models/vlas/pi05_policy.py（模块 策略；类别 source；类型 data-contract；符号 _prompt_token_bucketing_enabled）：集成图 runner 配置、prompt 分桶开关与 prefix/action 图可用性判断。
- python/sglang/multimodal_gen/runtime/layers/linear.py（模块 线性层；类别 source；类型 core-logic；符号 MergedReplicatedLinear, init, weight_loader）：新增 MergedReplicatedLinear，支撑非 TP 投影合并与 shard-aware 权重加载。
- python/sglang/multimodal_gen/test/unit/test_pi05_runtime_helpers.py（模块 单元测试；类别 test；类型 test-coverage；符号 _denoise_signature, test_denoise_graph_capacity_falls_back_without_capturing_new_signature, test_zero_denoise_graph_capacity_disables_runner, _FakeGraph）：集中覆盖图缓存容量、LRU 驱逐、回退、零容量禁用等生命周期行为。
- python/sglang/multimodal_gen/runtime/entrypoints/action/protocol.py（模块 协议；类别 source；类型 core-logic）：action metadata 暴露有效图可用性设置，便于调用方感知 exact/bucket 模式。

关键符号：_BoundedCaptureCache.get, _BoundedCaptureCache.put, _BoundedCaptureCache.prepare_admission, _BoundedCaptureCache._release, effective_token_length, select_prompt_token_bucket, bucket_prompt_tokens, Pi05PipelineConfig._validate_cuda_graph_config, Pi05PipelineConfig.prefix_cuda_graph_available,

Pi05PipelineConfig.action_cuda_graph_available, PiGemmaMLP.forward,PiGemmaAttention.project_qkv, MergedReplicatedLinear.weight_loader,Pi05ActionExpert._prompt_token_bucketing_enabled, prepare_denoise_layout

关键源码片段

python/sglang/multimodal_gen/runtime/vla/prompt_bucketing.py

新增 prompt 分桶工具，实现有效长度计算、桶选择与右填充掩码，是 opt-in 分桶模式的基础。

```
# python/sglang/multimodal_gen/runtime/vla/prompt_bucketing.py
# 提供 Pi0.5 prompt 分桶工具：计算有效 token 长度、选择桶、执行裁剪 / 右填充。
import torch
import torch.nn.functional as F
from collections.abc import Sequence


def effective_token_length(token_masks: torch.Tensor) -> int:
    """返回 batch 中最后一个可见 token 的位置（即有效长度）。"""

    if token_masks.ndim != 2:
        raise ValueError(
            f"Pi0.5 token masks 必须是 [batch, seq] 形状，当前为 {token_masks.shape}"
        )
    if token_masks.shape[1] == 0:
        return 0

    # positions 从 1 开始，非可见位置置 0，取每行最大值即为有效长度
    positions = torch.arange(
        1, token_masks.shape[1] + 1, device=token_masks.device, dtype=torch.long
    )
    lengths = torch.where(token_masks.to(torch.bool), positions, 0).amax(dim=1)
    return int(lengths.max().item())


def select_prompt_token_bucket(token_length: int, buckets: Sequence[int]) -> int | None:
    """选择能容纳 token_length 的最小桶；无匹配桶时返回 None（保持精确长度）。"""

    if token_length < 0:
        raise ValueError("token_length 必须非负")
    return next((int(bucket) for bucket in buckets if token_length <= bucket), None)


def bucket_prompt_tokens(
    tokens: torch.Tensor,
    token_masks: torch.Tensor,
    buckets: Sequence[int],
    *,
```

```
    pad_token_id: int = 0,
) -> tuple[torch.Tensor, torch.Tensor, int, int | None]:
    """将 prompt 裁剪或右填充到稳定的 CUDA 图桶长度。

    返回 (处理后 tokens, 处理后 masks, 逻辑长度, 命中的桶或 None)。
    """

    if tokens.ndim != 2:
        raise ValueError(f"Pi0.5 tokens 必须是 [batch, seq] 形状，当前为 {tokens.shape}")
    if token_masks.shape != tokens.shape:
        raise ValueError(
            "Pi0.5 tokens 与 token masks 形状必须一致，当前为 "
            f"{tokens.shape} 与 {token_masks.shape}"
        )

    logical_length = effective_token_length(token_masks)
    bucket = select_prompt_token_bucket(logical_length, buckets)
    target_length = bucket if bucket is not None else logical_length

    # 保留空 prompt 的既有回退行为：未命中任何桶时保持原始序列长度
    if target_length == 0 and bucket is None:
        target_length = tokens.shape[1]

    if tokens.shape[1] >= target_length:
        return (
            tokens[:, :target_length],
            token_masks[:, :target_length],
            logical_length,
            bucket,
        )

    padding = target_length - tokens.shape[1]
    # 右填充：token 用 pad_token_id，mask 用 False，保证填充位置被掩码忽略
    return (
        F.pad(tokens, (0, padding), value=pad_token_id),
        F.pad(token_masks, (0, padding), value=False),
        logical_length,
        bucket,
    )
```

python/sglang/multimodal_gen/runtime/layers/linear.py

新增 MergedReplicatedLinear，支撑非 TP 投影合并与 shard-aware 权重加载。

```
# python/sglang/multimodal_gen/runtime/layers/linear.py
# MergedReplicatedLinear：非 TP 下将多个逻辑投影合并为单个物理权重，
# 一次 GEMM 产出全部分片，同时保持权重加载时按 shard 写入。

class MergedReplicatedLinear(ReplicatedLinear):
    """Packed replicated linear layers with shard-aware weight loading.
```

```
这是 MergedColumnParallelLinear 的非张量并行版本：
独立逻辑投影存储在同一个物理权重里，eager 推理只发一个 GEMM。
"""

def __init__(
    self,
    input_size: int,
    output_sizes: list[int],
    bias: bool = True,
    skip_bias_add: bool = False,
    params_dtype: torch.dtype | None = None,
    quant_config: QuantizationConfig | None = None,
    prefix: str = "",
):
    self.output_sizes = output_sizes
    super().__init__(
        input_size=input_size,
        output_size=sum(output_sizes),
        bias=bias,
        skip_bias_add=skip_bias_add,
        params_dtype=params_dtype,
        quant_config=quant_config,
        output_sizes=output_sizes,
        prefix=prefix,
    )

def weight_loader(
    self,
    param: Parameter,
    loaded_weight: torch.Tensor,
    loaded_shard_id: int | str | None = None,
) -> None:
    if loaded_shard_id is None:
        return super().weight_loader(param, loaded_weight)

    # 字符串 shard 名转数字索引，例如 q/k/v 对应 0/1/2
    if isinstance(loaded_shard_id, str):
        try:
            loaded_shard_id = {"q": 0, "k": 1, "v": 2}[loaded_shard_id]
        except KeyError as exc:
            raise ValueError(f"Invalid merged shard id: {loaded_shard_id}") from exc
    if not 0 <= loaded_shard_id < len(self.output_sizes):
        raise ValueError(f"Invalid merged shard id: {loaded_shard_id}")

    param_data = param.data
    output_dim = getattr(param, "output_dim", None)
    if output_dim is not None:
        # 常规权重：按 output_dim 窄化到对应分片区间
        shard_offset = sum(self.output_sizes[:loaded_shard_id])
```

```
            shard_size = self.output_sizes[loaded_shard_id]
            param_data = param_data.narrow(output_dim, shard_offset, shard_size)
        elif getattr(param, "is_metadata", False):
            shard_size = loaded_weight.shape[0]
            param_data = param_data.narrow(0, loaded_shard_id * shard_size, shard_size)
        elif getattr(param, "needs_scalar_to_array", False):
            param_data, loaded_weight = adjust_scalar_to_fused_array(
                param_data, loaded_weight, loaded_shard_id
            )

        if tuple(param_data.shape) != tuple(loaded_weight.shape):
            raise ValueError(
                f"Tried to load merged shard of size {loaded_weight.size()} "
                f"to a parameter slice of size {param_data.size()}"
            )
        param_data.copy_(loaded_weight)
```

评论区精华

该 PR 没有任何 review 评论，唯一的 comment 是作者触发的 CI 命令 /tag-and-rerun-ci extra；review 列表为空。核心设计决策（默认 exact、容量满回退 eager、分桶仅作 opt-in）均由 PR body、H200 数据与 53 个单元测试背书。作者在 PR body 中明确提示：padding 会改变 reduction shape，进而可能改变 BF16 舍入，因此分桶模式保持 opt-in，并要求部署方先验证策略质量。

- 暂无高价值评论线程

风险与影响

- 风险：
  - BF16 舍入变化：分桶右填充改变 reduction shape，PR 自测非边界长度最大归一化 action 差异为 0.09589，部署 bucket 模式前需验证策略质量。
  - 图容量回退：exact 模式容量满后，未见过的签名回退 eager，可能导致 p50 与长尾抖动，需要监控 _BoundedCaptureCache 的 hits/misses/evictions 指标。
  - 权重加载契约：非 TP 分支权重布局从独立 nn.Linear 改为合并权重，依赖独立 q_proj/k_proj/gate_proj 的外部加载逻辑可能失效。
  - 配置校验：pi05.py 的 __post_init__ 在启动时校验 buckets，非法配置会直接报错；默认值安全，但旧配置若含负值会启动失败。
  - masked 图捕获：新增 prefix_full_attention 签名与可变 K/V 刷新逻辑，捕获 / 重放顺序错误会产生错误输出，部分路径仅有单元测试覆盖，真实 LIBERO 场景仍需验证。
  - 影响：性能影响集中在 diffusion/Pi0.5 流水线：H200 端到端 p50 在 exact 图模式下降低约 67.7%，eager 模式降低约 9%，内核启动减少约 10.6%。有界图缓存同时限制了 CUDA 图显存驻留，避免无界增长；新增配置项 prompt_token_buckets、action_cuda_graph_max_entries 与 VLAGraphCacheInfo 让运维可观测图命中率与驱逐情况。团队获得可复用的 _BoundedCaptureCache、MergedReplicatedLinear 与

prompt 分桶工具，为后续 VLA 模型优化提供参考实现；测试集中在 diffusion 模块，整体风险可控。

- 风险标记：BF16 舍入变化，图容量回退 *eager*，权重加载契约变更，配置校验破坏兼容，*masked* 图捕获正确性

关联脉络

- PR #34588 （被本 PR 合并并取代的先行 PR）：PR body 声明 Supersedes #34588，本 PR 合并了其中有用的部分并进行了配置、生命周期、fallback、文档与测试的清理。