PR #35318 完整报告

sgl-project/sglang

[Perf] PaddleOCR-VL: overlap page preprocessing, pack the ViT, enable prefill CUDA graph

合并时间：2026-08-19 08:20

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35318

执行摘要

- 一句话：打包视觉塔 + 并发预处理 + CUDA 图，OCR 吞吐近 2 倍
- 推荐动作：值得精读。设计层面有四个可复用点：宿主端边界避免 device-to-host 同步、LFU 位置编码缓存、kernel==stride 时 conv 到 matmul 的算子降级、多模态架构在 CUDA graph allowlist 上的回归。流程层面，作者对 worker 数和 vLLM 对比数字的两次自我纠偏，以及把“逐位精确”与“GEMM 容差”分开断言的测试哲学，是高质量性能工程的范本。建议后续在引入低精度或新后端时重点回归数值一致性。

功能与动机

PR body 直接给出量化目标：“PaddleOCR-VL served a 1080p page in 235 ms and saturated at 6.7 req/s on an H200 while the GPU sat idle”，且 profiling 显示瓶颈几乎不在模型自身计算。四个根因：QwenVLImageProcessor 的并发 worker 只授予显式 model_type 白名单而 paddleocr_vl 不在其中（87 ms/ 页预处理把吞吐封顶在约 11 req/s）；通用 multimodal 规则禁掉 breakable CUDA graph，连纯文本 prefill 都失去加速；ViT 每图一次设备端 max() 建 rope 表、一次 host sync 切片 encoder 输出、一次位置插值重算（LFU 缓存写了但从未被调用）、一次完整 projector；quant_config 被接受后静默丢弃，--quantization 对 LM 和视觉塔都是 no-op。

实现拆解

步骤 1：图像预处理并发化 — python/sglang/srt/multimodal/processors/paddleocr_vlm.py

PaddleOCRVLImageProcessor 继承自 QwenVLImageProcessor，后者只在显式 model 白名单上授予并发；PR 在子类上声明 supports_mm_processor_concurrency=True、auto_mm_processor_worker_num=2、auto_mm_io_worker_num=16。初版提交是 4 个 worker，作者在合并前用 H200 32 路并发对三种负载做扫描，发现 4 在每个 shape 上都劣于 2，最终以提交 90e24b2 修正为 2 并用单测钉死。文档页预处理约 87 ms，单 worker 时吞吐上限约 11 req/s，这是纯 CPU 重叠优化，预处理结果不变。

步骤 2：视觉塔打包化 — python/sglang/srt/models/paddleocr_vl.py（+229/−265）

这是核心重构，围绕“边界留在宿主端”的设计契约展开：

- 新增 build_packed_2d_position_ids：由宿主端 grid_thws 直接生成整批 [total_patches, 2] 二维位置 id，max_grid_size 也在宿主端推导，避免设备端 max() 触发同步；

- 新增 merge_patch_neighbourhoods：把 2x2 邻域合并抽成独立步骤，逐图一次 strided copy 写进预分配 buffer；
- Projector.forward 重写：pre_norm 整批 → merge → 两个投影各跑一次，N 图 batch 从 4N 个 kernel 降到 N 次拷贝 + 3 个 kernel；
- interpolate_pos_encoding 签名简化并返回 contiguous() 张量；此前写了但从未被调用的 fetch_position_embedding_lfu_cache 真正接入 forward，按 (h, w) 缓存最多 20 个网格（约 12 MiB）；
- patch_embedding 显式走 Conv2dLayer 的 unfold+matmul 路径（kernel == stride、零 padding），避免每 forward 都发起 cuDNN conv；
- 删除 numpy、einops 依赖，模块 docstring 写明打包布局契约。

步骤 3：prefill CUDA graph 白名单与契约修复 — model_config.py、ernie4.py

multimodal_breakable_cuda_graph_supported_model_archs 加入 PaddleOCRVLForConditionalGeneration。带图 batch 在 replay 时被拒绝并退回 eager，因此只利好文本 / 混合流量（2704 token 单流 TTFT 16.1 → 11.5 ms）。同时给 Ernie4Model 新增标准 get_input_embeddings，替换调用侧 hasattr + types.MethodType 动态补丁；quant_config 也改为真正透传，结束 --quantization 静默 no-op。

步骤 4：测试与文档

- test/registered/unit/models/test_paddleocr_vl_vision.py：fp64 下把打包结果钉在逐图参考实现上，覆盖 2x2 合并 permutation 精确性、batch 不变性、位置编码缓存与未缓存插值一致性、LFU 驱逐、conv 与 matmul 路径、打包二维位置 id；
- test/registered/unit/models/test_paddleocr_vl_serving_defaults.py：守卫并发开关、worker 数、基类保守性、CUDA graph allowlist——全是“refactor 会静默丢掉”的配置；
- test/registered/vlm/test_paddleocr_vl_server.py：单图 OCR + 四种不同尺寸并发请求，验证打包前向不跨图串数据；
- 文档：cookbook 页（0.9B/1.5/1.6 共享同一 config.json，一个 recipe 服务三个版本）、supported-models 行、docs 侧栏与 Unlimited-OCR.mdx 交叉引用。

关键文件：

- python/sglang/srt/models/paddleocr_vl.py（模块 视觉塔；类别 source；类型 core-logic；符号 build_packed_2d_position_ids, merge_patch_neighbourhoods, interpolate_pos_encoding, fetch_position_embedding_lfu_cache）：视觉塔从逐图处理重构为整批打包：新增 build_packed_2d_position_ids 与 merge_patch_neighbourhoods，重写 Projector.forward 与 SiglipVisionEmbeddings.forward，接入 LFU 位置编码缓存，patch embedding 走 unfold+matmul 路径，移除 numpy/einops 依赖。这是本次性能提升的核心载体。
- test/registered/unit/models/test_paddleocr_vl_vision.py（模块 单元测试；类别 test；类型 test-coverage；符号 _VisionConfig, _TextConfig, _grid_offsets, _packed_features）：CPU 单测把打包视觉数学钉在 fp64 逐图参考实现上，覆盖 2x2 合并 permutation 精确性、batch 不变性、位置编码 LFU 缓存、conv 与 matmul 路径等；并明确区分“纯数据搬运必须逐位相等”与“GEMM 批量化允许末位容差”两类断言。

- test/registered/unit/models/test_paddleocr_vl_serving_defaults.py（模块 默认守卫；类别 test；类型 test-coverage；符号 test_processor_opts_into_concurrency, test_worker_count_stays_at_the_measured_optimum, test_concurrency_opt_in_is_not_inherited_by_accident, test_prefill_breakable_cuda_graph_is_allowlisted）：专门守卫“refactor 会静默丢掉”的 serving 默认值：并发开关必须声明在 PaddleOCRVLImageProcessor 自己身上、worker 数钉死在实测最优的 2、基类保持保守、CUDA graph allowlist 成员不可丢。
- test/registered/vlm/test_paddleocr_vl_server.py（模块 端到端测试；类别 test；类型 test-coverage；符号 TestPaddleOCRVLServer, _font, _make_ocr_image_url, _ocr）：端到端 OpenAI API 测试：单图 OCR 验证功能，四种不同尺寸的并发请求验证打包前向不会跨图串数据，是对打包布局正确性的最直接集成验证。
- python/sglang/srt/multimodal/processors/paddleocr_vlm.py（模块 图像预处理；类别 source；类型 core-logic；符号 PaddleOCRVLImageProcessor）：让文档页预处理跨 worker 并发：在处理器类上声明 supports_mm_processor_concurrency、2 个 worker 与 16 个 IO 线程，并附上 H200 实测的取舍说明。这是吞吐翻倍的另一个关键杠杆。
- python/sglang/srt/configs/model_config.py（模块 模型配置；类别 source；类型 data-contract；符号 multimodal_breakable_cuda_graph_supported_model_archs）：把 PaddleOCRVLForConditionalGeneration 加入 multimodal_breakable_cuda_graph_supported_model_archs 全局白名单，让多模态架构恢复 prefill CUDA graph；带图 batch 在 replay 阶段被拒并回 eager，因此只影响文本 / 混合流量。
- python/sglang/srt/models/ernie4.py（模块 语言主干；类别 source；类型 data-contract；符号 get_input_embeddings）：给 Ernie4Model 增加标准的 get_input_embeddings 方法，替换 PaddleOCR-VL 调用侧 hasattr + types.MethodType 的动态补丁，属于公共模型接口的契约修复。
- docs/src/snippets/configs/PaddlePaddle/paddleocr-vl.jsx（模块 部署脚本；类别 source；类型 configuration）：cookbook 部署配置：0.9B/1.5/1.6 共享同一 config.json 所以一个 recipe 服务三个版本，按 Page Resolution 暴露 --mm-process-config 档位，并在 flags 中落到实测调优过的调度参数。
- docs/cookbook/autoregressive/Baidu/PaddleOCR-VL.mdx（模块 文档；类别 other；类型 documentation；符号 render, parse）：新增 PaddleOCR-VL 完整 cookbook 页，包含部署步骤、配置建议（页分辨率成本、并发预处理、CUDA graph、radix cache、饱和吞吐 flags）与 H200 实测数据，是文档主线。
- docs/docs/supported-models/multimodal_language_models.mdx（模块 支持列表；类别 other；类型 documentation）：在 supported-models 表格中登记 PaddleOCR-VL（0.9B/1.5/1.6），并提示任务由 prompt 选择、不要开 --trust-remote-code，属于模型支持矩阵的契约更新。

关键符号：build_packed_2d_position_ids, merge_patch_neighbourhoods, interpolate_pos_encoding, fetch_position_embedding_lfu_cache, Projector.forward, SiglipVisionEmbeddings.forward, encode_image, get_input_embeddings

关键源码片段

test/registered/unit/models/test_paddleocr_vl_vision.py

CPU 单测把打包视觉数学钉在 fp64 逐图参考实现上，覆盖 2x2 合并 permutation 精确性、batch 不变性、位置编码 LFU 缓存、conv 与 matmul 路径等；并明确区分“纯数据搬运必须逐位相等”与“GEMM 批量化允许末位容差”两类断言。

```
# ============ test/registered/unit/models/test_paddleocr_vl_vision.py ============
# 测试把“必须逐位一致”和“允许末位差异”分开断言：
# merge 是纯数据搬运，必须 bit-exact；投影则不然——把 N 个逐图 GEMM 合成
# 一个会改变分块与求和顺序，末位会有差异（fp64 实测最大约 3e-14 相对误差，
# 且依赖 BLAS 实现：Apple silicon 上相等，x86 上不等）。permutation 错误
# 会把数值移动约 1 个量级，1e-12 容差依然能一锤定音地抓住它。
_GEMM_REORDER_RTOL = 1e-12
_GEMM_REORDER_ATOL = 1e-12


def test_projector_merge_permutation_is_exact():
    """2x2 重排只搬数据不做运算，因此必须逐位相等。"""
    torch.manual_seed(1)
    projector = _build_projector()
    packed = _packed_features()
    normed = projector.pre_norm(packed)

    actual = merge_patch_neighbourhoods(normed, GRIDS, projector.merge_kernel_size)

    # 逐图做 rearrange 的结果拼起来作为参照
    m1, m2 = projector.merge_kernel_size
    expected = torch.cat(
        [
            rearrange(
                normed[offset : offset + t * h * w],
                "(t h p1 w p2) d -> (t h w) (p1 p2 d)",
                t=t,
                h=h // m1,
                p1=m1,
                w=w // m2,
                p2=m2,
            )
            for offset, (t, h, w) in zip(_grid_offsets(), GRIDS)
        ],
        dim=0,
    )

    assert actual.shape == expected.shape
    # 纯数据搬运，用 torch.equal 做逐位比较
    assert torch.equal(actual, expected)


def test_projector_packed_merge_matches_per_image_reference():
```

```
    """打包后的 batch 投影必须等于逐图投影的拼接（允许 GEMM 重排容差）。"""
    torch.manual_seed(1)
    projector = _build_projector()
    packed = _packed_features()

    actual = projector(packed, GRIDS)
    expected = _reference_projector_output(projector, packed)

    assert actual.shape == expected.shape
    assert actual.shape[0] == sum(t * h * w for t, h, w in GRIDS) // 4
    assert actual.shape[1] == _TextConfig.hidden_size
    torch.testing.assert_close(
        actual, expected, rtol=_GEMM_REORDER_RTOL, atol=_GEMM_REORDER_ATOL
    )
```

python/sglang/srt/multimodal/processors/paddleocr_vlm.py

让文档页预处理跨 worker 并发：在处理器类上声明 supports_mm_processor_concurrency、2 个 worker 与 16 个 IO 线程，并附上 H200 实测的取舍说明。这是吞吐翻倍的另一个关键杠杆。

```
# ============ python/sglang/srt/multimodal/processors/paddleocr_vlm.py ============
class PaddleOCRVLImageProcessor(QwenVLImageProcessor):
    models = [PaddleOCRVLForConditionalGeneration]

    # 文档页是远比聊天图片更重的预处理单元：resize + normalize + patchify
    # 一整页高清扫描要花几十毫秒，单 worker 会让请求吞吐封顶在
    # 1 / preprocess_time，GPU 再闲也上不去。这里跨 worker 做重叠，
    # 预处理本身的逻辑不变。
    #
    # 选 2 而不是更多：H200 上 32 路并发实测，2 worker 在每个 shape 上都
    # 同时胜过 1 和 4（1080p 页 6.72 -> 9.55 req/s，4 worker 只有 8.92；
    # 360p 页 512 token 输出 22.38 -> 25.20，4 worker 25.06）。超过 2 后，
    # 请求到达被打散对 GPU prefill 批次的碎片化影响，超过了额外重叠能
    # 换来的收益。
    auto_mm_processor_worker_num = 2
    auto_mm_io_worker_num = 16
    supports_mm_processor_concurrency = True
```

评论区精华

PR 没有外部 reviewer 评论，但作者本人留下了三处高价值的技术自我纠偏：

1. worker 数从 4 修正为 2。评论原文：“Corrected the worker count from 4 to 2 … two won every shape while four was consistently worse: 1080p pages 6.72 → 9.55 req/s at two, 8.92 at four; 360p pages with 512-token outputs 22.38 → 25.20 at two, 25.06 at four。Past two workers, spreading request arrivals fragments the GPU prefill batches faster than the extra CPU overlap pays for itself。”

2. 主动推翻与 vLLM 的对比结论。评论原文："That figure does not hold up... it came from a run of the worker=4 configuration（this PR later corrected to 2）with only n=3 per arm。Re-measured with the configuration that actually shipped... 4 independent server instances × 3 reps = n=12 per arm... difference −0.009 (−0.1%)，Welch t = −0.05，95% CI [−0.386, +0.367] → straddles zero。"
3. 提交 cd06180 修正过严的测试容差：投影 GEMM 批量化改变求和顺序，末位差异是 BLAS 依赖的（Apple silicon 相等、x86 不等）。把断言拆成"2x2 合并纯数据搬运必须逐位相等"与"投影允许 1e-12 容差"，并论证 permutation 错误会移动约 1 个量级、容差依然能抓住它。

- 并发预处理 worker 数从 4 修正为 2 (performance): worker 数定为 2（提交 90e24b2），并用 test_worker_count_stays_at_the_measured_optimum 单测钉死防回退。
- 回收与 vLLM 饱和吞吐对比的结论 (performance): PR 不再宣称落后 vLLM，饱和吞吐修正为持平；强调实例间放置效应需要 n≥12 才能平均掉。
- 投影测试 bit-exact 断言过严，拆分为 merge 精确 + GEMM 容差 (testing): 提交 cd06180 落地：新增 test_projector_merge_permutation_is_exact，并重写 test_projector_packed_merge_matches_per_image_reference 的容差语义。

风险与影响

- 风险：
  1. 数值末位差异：打包 GEMM 与逐图 GEMM 求和顺序不同，结果末位有 BLAS 实现依赖的差异。贪婪解码下 10/10 逐字节一致，但引入采样、低精度（FP8/INT8）或新后端后差异可能放大；现有测试只覆盖 fp64。
  2. 位置编码 LFU 缓存：命中条件是 (h, w) 精确匹配；文档尺寸高频变化时会反复驱逐重算。缓存是模块级 dict，多实例并发安全性未被测试覆盖。
  3. 全局 allowlist 变更：multimodal_breakable_cuda_graph_supported_model_archs 是全局列表，放行后带图 batch 在 replay 时被拒并回 eager，行为切换成本未被计入基准；该列表本身是"refactor 静默丢配置"的高危点，靠新增单测守护。
  4. 默认行为变更：并发预处理默认开启（2 worker + 16 IO 线程），低核数容器可能 CPU 争抢；基类保持 1 worker 保守值，只有 PaddleOCR-VL 显式开启。
  5. 量化配置透传：--quantization 从静默 no-op 变为真正生效，此前被掩盖的配置错误可能新暴露为加载失败（预期内修复，需回归观察）。
  6. 契约变更：ernie4.py 新增 get_input_embeddings 属公共接口变化，若存在依赖旧 hasattr 探测的调用方，行为可能改变（当前无证据）。 - 影响：用户侧：PaddleOCR-VL 0.9B/1.5/1.6（共享同一 config.json）单流 TTFT 约 2 倍、32 路并发吞吐约 1.84 倍；纯文本预填因 CUDA graph 提速 40%；与 vLLM 0.27.1 对比，饱和吞吐从"落后 4%"修正为持平（−0.1%，95% CI 跨零），单流 TTFT 领先约 10%。系统侧：paddleocr_vl.py 核心路径重构（+229/−265），vision tower 从逐图处理变为整批打包，建立"宿主端 grid 边界 + 零 D2H 同步"的可复用模式；同时修复了处理器并发、CUDA graph、quant 透传三类此前被静默丢弃的配置，处理器与配置默认行为对全部 PaddleOCR-VL 请求生效。团队侧：新增"打包算法对逐图参考的 fp64 钉死"与"serving allowlist 守卫"两类测试范式，对后续 refactor 有保护价值；基准方法学（双臂交替、

n≥12、事后回收错误结论）值得推广。 - 风险标记：核心视觉塔路径重写，数值末位 BLAS 依赖差异，全局 CUDA graph 白名单变更，并发预处理默认开启，量化配置透传暴露旧问题

关联脉络

- PR #35006 [Diffusion] Reuse SRT Qwen vision and text modules: PaddleOCR-VL 的处理器继承自 QwenVLImageProcessor，并发 worker 取 2 正是参考 qwen_vl 处理器已文档化的 tradeoff；两条线都在收敛 Qwen 视觉编码器体系的复用路径。
- PR #35172 [Quantization] Extract shared checkpoint quant metadata resolver: 本 PR 修复了 --quantization 对 PaddleOCR-VL 静默 no-op 的问题，与量化元数据解析收敛到共享服务（35172/35174）的治理方向一致。
- PR #35174 [Diffusion] Reuse shared checkpoint quant metadata resolver: 与 35172 同源，说明量化配置链路在持续收敛；本 PR 的 quant_config 透传是该治理要覆盖的问题面之一。