PR #34713 完整报告

sgl-project/sglang

[diffusion] Decouple encoder parallelism from the DiT parallel layout

合并时间：2026-08-19 00:16

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34713

执行摘要

- 一句话：encoder 并行与 DiT 布局解耦，新增双并行组
- 推荐动作：值得精读，尤其是对分布式并行系统设计者。重点看三点：(1) parallel_state.py 中 replica/encoder-DP 正交组的构造与 destroy 幂等处理，是“用组分解并行维度、用 alias 省开销”的典型范例；(2) text_encoding.py._text_encode_dp_group 的 gate 设计，把策略（auto/dp/fold）与拓扑能力（group availability）清晰分层；(3) PR body 对旧假设的自我推翻过程，体现了在 centralized encoder TP context 下重新审视并行契约的必要性。若你负责 MiniMax-H3 部署，需同步关注 encoder_parallel 语义变化。

功能与动机

PR body 明确指出：一个 pipeline DP replica 拥有一个 request batch，encoder 并行必须在 replica 内部选择，不能跨 --dp-size replicas 混请求。旧实现错误地假设“纯 DiT TP 下每个 rank 都持有完整 encoder”，但 main 上 centralized 的 encoder TP context 已使 native encoder 权重跨 DiT TP group 分片（"That is not true after main centralized the encoder TP context: native encoder weights are sharded across the DiT TP group"）。因此需要保留该契约，只把 batch DP 组合在独立 encoder copies 之间，并让纯 TP 部署（replica == tp）不再被 gate 拒绝。

实现拆解

实现分 5 步推进，核心在分布式并行组的重新建模与 gate 逻辑改造：

1. 在 parallel_state.py 引入两个一级并行组。新增全局 _REPLICA（轴 tp-sp-pp-cfg，即“共享同一 request batch 的 ranks”）与 _ENCODER_DP（轴 sp-pp-cfg，固定 TP 坐标，连接同一 replica 内多个 TP-sharded encoder copies），并配套 get_replica_group()、get_encoder_data_parallel_group()、_get_encoder_data_parallel_group_ranks() 三个访问入口。初始化时：dp_size == 1 直接复用 world group 作为 replica（避免额外 device communicator 开销）；多 replica 时按非平凡轴组装（只有一个非平凡轴时直接复用该轴 group，否则新建 coordinator）。_ENCODER_DP 在 tp == 1 时等于 replica；tp > 1 且 sp/pp/cfg 有非平凡度时复用单轴组或新建；纯 TP replica（无 SP/PP/CFG）则保持 None，表示只有单个 encoder copy、无法 batch DP。destroy_model_parallel() 相应做了 alias 去重与跳过 world group 的防护。
2. 改造 text_encoding.py 的 _text_encode_dp_group 门控。删除原先 (tp_size or 1) != 1、(dp_size or 1) != 1 的硬性拒绝，改为统一从 get_encoder_data_parallel_group() 取组：

fold 过的 encoder、非 TextEncoder 类型、不支持 DP、组为 None 或 world_size <= 1 时返回 None。_data_parallel_text_encode 的文档同步说明“TP 组内每个 rank 拿同一 slice，正交 encoder-DP 组再 all-gather”，保证 TP-sharded copy 的 slice 一致性。

3. 调整 server_args.py 的参数契约。adjust_pipeline_config 新增 fold_replica 分支：显式 encoder_parallel == "fold" 时把 proposal 扩大到整个 replica（mode = "replica"），而 auto 保持保守（不折叠纯 TP replica）。同时删除 _validate_batching 中 " encoder_parallel=dp 要求 tp_size=1 且 dp_size=1" 的报错，并更新 --encoder-parallel 的 CLI 帮助文本。配套把 EncoderConfig.parallel_folding_mode 的类型字面量扩展为 " sp"|"world"|"replica"，get_folding_tp_group 增加 mode == "replica" 时返回 get_replica_group() 的分支；text_encoder_loader.py 的 prefer_dp 判定也改为依据真实的 encoder-DP group。
4. MiniMax-H3 专用 stage 语义同步。minimax_h3/stages/text_encoding.py 把内部 extra key 从 _MINIMAX_H3_SINGLE_RANK_TEXT_ENCODE_EXTRA_KEY 改为 _MINIMAX_H3_SINGLE_COPY_TEXT_ENCODE_EXTRA_KEY（"rank" → "copy"），注释与日志文案同步，表达“一个 encoder copy 可自身横跨 TP 组”的新语义。
5. 测试与文档配套。新增 test_text_encode_dp_gate.py（9 个用例，覆盖纯 TP 无独立 copy、TP×encoder-DP 组合、folded 排除、多 replica 隔离、policy 门控，以及 tp2 x sp2 x dp2 下 [[0,2],[1,3],[4,6],[5,7]] 的组划分断言）；test_encoder_world_folding.py 增补 test_explicit_fold_proposes_replica_for_any_shape 并把原 policy 无关性测试改为 test_adjust_proposal_policy_dependence；文档更新 encoder_parallel.mdx、CLI 参考和 MiniMax-H3 cookbook。

关键文件：

- python/sglang/multimodal_gen/runtime/distributed/parallel_state.py（模块 分布式层；类别 source；类型 core-logic；符号 _get_encoder_data_parallel_group_ranks, get_replica_group, get_encoder_data_parallel_group, initialize_model_parallel）：核心变更：新增 _REPLICA（tp-sp-pp-cfg）与 _ENCODER_DP（sp-pp-cfg）两个并行组及访问函数，重写初始化与 destroy 逻辑，是本次解耦架构的基石。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/text_encoding.py（模块 文本编码；类别 source；类型 core-logic；符号 _text_encode_dp_group, _data_parallel_text_encode）：文本编码 DP 门控 _text_encode_dp_group 从 world group + tp/dp 尺寸校验改为基于正交 encoder-DP group，是行为放开的直接落点。
- python/sglang/multimodal_gen/runtime/server_args/server_args.py（模块 参数配置；类别 source；类型 configuration；符号 adjust_pipeline_config, _validate_batching, add_cli_args）：参数契约变更：显式 fold 扩大到 replica 提案，删除 encoder_parallel=dp 的 tp/dp 校验，更新 CLI 帮助文本。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/minimax_h3/stages/text_encoding.py（模块 H3 管线；类别 source；类型 data-contract；符号 _MINIMAX_H3_SINGLE_COPY_TEXT_ENCODE_EXTRA_KEY, run_grouped_requests, _encode_ref2va）：MiniMax-H3 专用 stage 同步新语义：extra key 从 single_rank 改为 single_copy，文档说明 encoder copy 可自身横跨 TP 组。

- python/sglang/multimodal_gen/test/unit/test_text_encode_dp_gate.py（模块 编码并行；类别 test；类型 test-coverage；符号 _enc, _gate, test_dp_engages_for_replicated_encoder, test_dit_tp_composes_with_encoder_dp_group）：新增核心单测文件，覆盖 DP 门控在 TP 组合、纯 TP 无独立 copy、folded 排除、多 replica 隔离、policy 门控以及组划分断言。
- python/sglang/multimodal_gen/test/unit/test_encoder_world_folding.py（模块 编码器折叠；类别 test；类型 test-coverage；符号 test_explicit_fold_proposes_replica_for_any_shape, test_adjust_proposal_policy_dependence）：folding 提案测试更新：显式 fold 在任意多 rank replica 提 replica 组，policy 依赖关系修正。
- python/sglang/multimodal_gen/configs/models/encoders/base.py（模块 编码器配置；类别 source；类型 data-contract；符号 EncoderConfig.parallel_folding_mode）：数据契约扩展：parallel_folding_mode 字面量加入 replica 模式。
- python/sglang/multimodal_gen/runtime/loader/component_loaders/text_encoder_loader.py（模块 模型加载；类别 source；类型 core-logic；符号 load_customized）：prefer_dp 判定从 tp/dp 尺寸条件改为真实 encoder-DP group 能力。
- python/sglang/multimodal_gen/runtime/models/encoders/base.py（模块 编码器基类；类别 source；类型 core-logic；符号 get_folding_tp_group）：get_folding_tp_group 增加 replica 模式分支，返回 replica group。
- python/sglang/multimodal_gen/runtime/distributed/__init__.py（模块 分布式层；类别 source；类型 core-logic；符号 get_replica_group, get_encoder_data_parallel_group）：导出新并行组访问函数，保证对外 API 一致。
- docs/docs/sglang-diffusion/encoder_parallel.mdx（模块 文档；类别 docs；类型 documentation）：文档主更新：描述 replica 边界、纯 TP 行为与 TP+encoder-DP 组合方式。

关键符号：get_replica_group, get_encoder_data_parallel_group, _get_encoder_data_parallel_group_ranks, TextEncodingStage._text_encode_dp_group, ServerArgs.adjust_pipeline_config, get_folding_tp_group

关键源码片段

python/sglang/multimodal_gen/runtime/distributed/parallel_state.py

核心变更：新增 _REPLICA（tp-sp-pp-cfg）与 _ENCODER_DP（sp-pp-cfg）两个并行组及访问函数，重写初始化与 destroy 逻辑，是本次解耦架构的基石。

```
# 并行组常量：replica 是所有非 DP 轴（tp-sp-pp-cfg）的 ranks；
# encoder-DP 是固定 TP 坐标下的 sp-pp-cfg（TP 正交、replica 内局部）。
_REPLICA_PARALLEL_AXES = "tp-sp-pp-cfg"
_ENCODER_DP_PARALLEL_AXES = "sp-pp-cfg"


def get_replica_group() -> GroupCoordinator:
    """返回共享同一个 request batch 的 ranks —— 即当前 pipeline replica。

    当 dp_size > 1 时，world group 横跨多个 replicas，所以 replica 级
```

```
    encoder folding 必须用本组而不是 world group。只有进程只初始化了
    world group（未启用 model parallel）时，才回退到 world group。
    """
    if _REPLICA is not None:
        return _REPLICA
    return get_world_group()


def get_encoder_data_parallel_group() -> GroupCoordinator | None:
    """返回同一 replica 内可切分 batch 的 encoder copies 正交组。

    非折叠 native encoder 使用 DiT TP group，因此 batch DP 只能发生在
    replica 中与 TP 正交的 ranks 之间：TP 坐标相同的 ranks 接收同一
    batch slice，同一 TP group 内每个 rank 都拿到该 slice 的完整输出。
    纯 TP replica 只有一个 encoder copy，返回 None 表示无法 batch DP。
    """
    if _ENCODER_DP is not None:
        return _ENCODER_DP
    if not model_parallel_is_initialized():
        return get_world_group()
    return None


# 在 initialize_model_parallel 中构造两个新组。
if data_parallel_size == 1:
    # 单 replica 部署直接 alias world group，避免额外创建 communicator
    # 及其 GPU 缓冲 —— 常见路径不能承受多余开销。
    _REPLICA = get_world_group()
else:
    # 多 replica 时组装所有非平凡轴；只有一个非平凡轴时直接复用该轴组。
    replica_axis_groups = [
        group
        for degree, group in (
            (tensor_parallel_degree, _TP),
            (sequence_parallel_degree, _SP),
            (pipeline_parallel_degree, _PP),
            (classifier_free_guidance_degree, _CFG),
        )
        if degree > 1
    ]
    if len(replica_axis_groups) <= 1:
        _REPLICA = replica_axis_groups[0] if replica_axis_groups else _TP
    else:
        _REPLICA = init_parallel_group_coordinator(
            group_ranks=rank_generator.get_ranks(_REPLICA_PARALLEL_AXES),
            local_rank=get_world_group().local_rank,
            backend=backend,
            parallel_mode="replica",
        )
```

评论区精华

本 PR 没有人类 reviewer 的评审评论（review_comments_count = 0），唯一 Issue 评论是 mintlify bot 的文档预览通知。最有价值的设计论证来自 PR body 与 commit message 的自我迭代：

- 旧假设被推翻的论证（PR body）："The previous version of this PR assumed that every rank under pure DiT TP held a complete encoder. That is not true after main centralized the encoder TP context... This update preserves that contract and composes batch DP only across independent encoder copies." 这是对"TP 决不分片 encoder"这一隐含假设的明确纠正。
- 单 replica 部署的零成本优化（commit 285db1f）："the common case must not pay for an extra device communicator and its GPU buffers"——dp_size == 1 时 _REPLICA 直接 alias world group，destroy 路径跳过 alias。
- fold 是唯一的 policy 例外（commit c122e26 与测试注释）：adjust_pipeline_config 本应只读并行度不读 policy，但显式 fold 必须例外，否则纯 TP replica 从未被 proposal 过 fold group，finalize 无法凭空生成。
- encoder DP 是否允许出现在 DiT TP 下 (design): 接受新架构：batch DP 只在独立 encoder copies 之间组合，TP-sharded 的 encoder copy 内部保持 DiT TP 契约不变。
- replica group 的实现成本 (performance): dp_size==1 零额外运行时开销，destroy 增加去重列表与跳过 world 的防护。
- folded encoder 与 DP 的互斥 (correctness): fold 与 DP 严格互斥，由 gate 保证；显式 fold 在任意多 rank replica 提 replica 组作为 fold 例外。

风险与影响

- 风险：风险集中在分布式组构建与契约放宽上：

1. 组初始化路径复杂，collective 错配风险高（parallel_state.py）。_REPLICA 与 _ENCODER_DP 有 alias、复用单轴组、新建 coordinator 三条路径，任一分支对 RankGenerator 轴顺序理解错误都会导致集合通信进程配错（死锁或 hang）。单测只覆盖了 tp2 x sp2 x dp2 一种组合，cfg 轴参与的多轴组合（如 tp×sp×cfg）缺少实测。
2. get_encoder_data_parallel_group() 返回值语义分叉。未初始化 model parallel 时返回 world group，初始化后纯 TP replica 返回 None；所有调用方（text_encoding.py、text_encoder_loader.py、MiniMax-H3 stage）都必须显式处理 None，未来新增调用点容易遗漏。
3. server_args 校验放宽导致静默回退。移除 encoder_parallel=dp 的 tp/dp 硬校验后，错误拓扑不再启动时报错，而是运行时 gate 返回 None 静默退化为 replicated forward，运维排障成本上升。
4. MiniMax-H3 extra key 更名。_MINIMAX_H3_SINGLE_RANK_* → _MINIMAX_H3_SINGLE_COPY_* 是运行时内部 key，风险低，但若有持久化或跨版本序列化引用旧 key 会失效。
5. destroy 去重逻辑缺少针对性单测。alias 场景（_REPLICA is _WORLD、_ENCODER_DP is _TP）靠 destroyed_groups 去重，若某个 alias 未被正确识别会双 destroy 崩溃。 - 影

响：影响范围为 diffusion 子系统（sglang/multimodal_gen），不涉及 SRT 主推理路径，共 13 个文件、388 行新增。对用户 / 运维：--encoder-parallel dp 现在可与 --tp-size>1、--dp-size>1 组合，纯 TP 与 TP×SP 部署的批量文本编码吞吐路径被打开，miniMax-H3 等模型的多卡部署获得更多并行选择；CLI 帮助文本与文档同步更新。对系统：新增 replica_group 与 encoder_dp_group 两个进程组，但 dp_size == 1 的常见部署零额外开销（alias world）。对团队：确立了"encoder 布局独立于 DiT 并行布局"的契约，为后续 encoder 多副本显式部署、更多模型的 encoder 并行扩展打下基础。 - 风险标记：核心路径变更，分布式组初始化复杂，CLI 契约放宽，collective 通信依赖分组正确

关联脉络

- PR #34581 [Diffusion] Optimizing MiniMax-H3 for consumer-level GPUs: INT8 Linear + pluggable DiT attention backends: 同属 diffusion 子系统且同样修改 MiniMax-H3 cookbook 与 encoder/loader 相关路径，是 MiniMax-H3 生态的相邻能力扩展。
- PR #35107 [diffusion] Filter transformer safetensors by index.json to drop duplicate shard variants: 同为 diffusion 模型加载路径（transformer_load_utils / loader 域），本 PR 的 text_encoder_loader.py 改动与其在同一加载链路上。
- PR #34933 [diffusion] Per-section LoRA adapters on fused linear layers: 同属 diffusion runtime 并行 / 加载生态，LoRA 组合与 encoder 并行都依赖 fused layer 与权重路由的布局一致性。
- PR #31453 [Diffusion][Refactor] Refactor and extract complex RoPE implementation to layers/rotary_embedding for MOVA DiT: 同属 diffusion runtime 重构线，关注并行布局与共享 layer 工具化，与本 PR 的 encoder 布局解耦方向一致。