PR #30211 完整报告

sgl-project/sglang

[diffusion] encoder_parallel: unify encoder folding and batch data-parallel encoding

合并时间：2026-07-30 20:15

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30211

执行摘要

- 一句话：统一 diffusion 编码器并行策略，新增 DP 和 auto 模式
- 推荐动作：值得精读。该 PR 展示了如何在复杂分布式环境中做性能驱动的设计决策：基于实测数据设定阈值、拓扑感知、自动回退、细致处理默认值。对于维护 diffusion 流水线的工程师，理解 auto 决策逻辑和 dp 实现很有价值。

功能与动机

正式动机来自 PR body：'Across a multi-rank single replica (tp=1), a text/image encoder has three mutually-exclusive layouts, fixed at load time: fold, dp, replicate. This PR unifies the choice under one knob: --encoder-parallel auto|fold|dp|replicate'。目标是按需切换编码器布局，兼顾延迟和吞吐。

实现拆解

1. 新增 encoder_parallel 配置和 CLI（server_args.py）：添加 encoder_parallel 字段，支持 auto/fold/dp/replicate，serve 入口默认 dp，其他入口默认 auto。
2. 重构决策函数（encoders/base.py）：新增 DP_MIN_HIDDEN_SIZE=1024、_encoder_dims_divide、encoder_dp_capable、encoder_dp_worthwhile、group_has_measured_topology；修改 finalize_encoder_folding 接受 policy 和 batched 参数，根据宽度和拓扑决定 fold/dp/replicate。
3. 实现数据并行编码（stages/text_encoding.py）：新增 _data_parallel_text_encode 函数，每个 rank 编码 batch 的 1/world_size 切片后 all-gather 完整结果，包含跨秩批次大小一致性校验和 padding 处理。
4. 集成到 loader（text_encoder_loader.py、image_encoder_loader.py）：在 finalize_encoder_folding 调用中传入 server_args.encoder_parallel 和 server_args.batching_max_size > 1。
5. 默认值修复和拓扑门控：当 policy=dp 且 batching_max_size 未显式设置时将其提升为 replica_size；限制自动 fold/dp 仅在 NVLink 拓扑内启用（group_has_measured_topology）。

关键文件：

- python/sglang/multimodal_gen/runtime/pipelines_core/stages/text_encoding.py（模块 运行时；类别 source；类型 core-logic；符号 _data_parallel_text_encode, _shard,

_gather, _gather_seq）：核心运行时代码：新增 _data_parallel_text_encode 函数实现数据并行编码，以及 _text_encode_dp_group、_log_dp_choice 等辅助逻辑。是 DP 功能的主入口。

- python/sglang/multimodal_gen/runtime/models/encoders/base.py（模块 编码器基础；类别 source；类型 data-contract；符号 encoder_folding_worthwhile, _encoder_dims_divide, finalize_encoder_folding, group_has_measured_topology）：决策中枢：定义 FOLD_MIN_HIDDEN_SIZE 和 DP_MIN_HIDDEN_SIZE 阈值，新增 _encoder_dims_divide、group_has_measured_topology、encoder_dp_capable、encoder_dp_worthwhile，重构 finalize_encoder_folding 使其接受 policy 参数。所有布局选择逻辑内聚于此。
- python/sglang/multimodal_gen/test/unit/test_encoder_world_folding.py（模块 测试；类别 test；类型 test-coverage；符号 _run, _proposed_mode, test_adjust_proposes_regardless_of_policy, test_no_policy_touches_batching_max_size）：单元测试覆盖所有 policy 决策路径，包括 adjust_pipeline_config、finalize_encoder_folding、dp 启停条件、batching_max_size 不变性等，确保纯逻辑正确。
- python/sglang/multimodal_gen/runtime/server_args/server_args.py（模块 服务参数；类别 source；类型 core-logic）：新增 encoder_parallel 配置字段、CLI 参数 --encoder-parallel，以及在 adjust_pipeline_config 中处理 fold 模式提案。同时实现 batching_max_size 自动提升逻辑（dp 策略下但未显式设置时）。
- python/sglang/multimodal_gen/runtime/loader/component_loaders/text_encoder_loader.py（模块 加载器；类别 source；类型 core-logic）：文本编码器加载时调用 finalize_encoder_folding 并传入 policy 和 batched 参数，确定最终布局。
- python/sglang/multimodal_gen/runtime/loader/component_loaders/image_encoder_loader.py（模块 加载器；类别 source；类型 core-logic）：图像编码器加载时同步调用 finalize_encoder_folding，保持一致性。

关键符号：_data_parallel_text_encode, finalize_encoder_folding, encoder_folding_worthwhile, _encoder_dims_divide, encoder_dp_worthwhile, group_has_measured_topology, adjust_pipeline_config

关键源码片段

python/sglang/multimodal_gen/runtime/pipelines_core/stages/text_encoding.py

核心运行时代码：新增 _data_parallel_text_encode 函数实现数据并行编码，以及 _text_encode_dp_group、_log_dp_choice 等辅助逻辑。是 DP 功能的主入口。

```
# 数据并行文本编码：每个 rank 编码 1/world_size 切片，然后 all-gather 完整结果
# 每行由相同 kernel 计算（未分片编码器），输出与 replicate 路径位精确一致
def _data_parallel_text_encode(forward_fn, forward_kwargs: dict, group):
    world = group.world_size
    rank = group.rank_in_group
    input_ids = forward_kwargs["input_ids"]
```

```
        bs = input_ids.shape[0]
        # 快速检测跨 rank 的 batch 大小不一致，避免挂在 gather 上
        bs_sum = int(
            group.all_reduce(
                torch.tensor([bs], device=input_ids.device, dtype=torch.int64)
            ).item()
        )
        assert bs_sum == bs * world, (
            f"data-parallel text-encode batch size desynced across ranks "
            f"(rank {rank} bs={bs}, group sum={bs_sum} != {bs * world})"
        )
        chunk = (bs + world - 1) // world
        pad = chunk * world - bs

        def _shard(t):
            # 只对 batch 维度的 tensor 进行切分，否则保持原样
            if not torch.is_tensor(t) or t.shape[0] != bs:
                return t
            if pad:
                t = torch.cat([t, t[:1].expand(pad, *t.shape[1:])], dim=0)
            return t[rank * chunk : (rank + 1) * chunk]

        local_out: BaseEncoderOutput = forward_fn(
            {k: _shard(v) for k, v in forward_kwargs.items()}
        )

        def _gather(t):
            if t is None:
                return None
            # all_gather 后截断到真实 bs
            return group.all_gather(t.contiguous(), dim=0)[:bs]

        def _gather_seq(seq):
            return tuple(_gather(t) for t in seq) if seq is not None else None

        return BaseEncoderOutput(
            last_hidden_state=_gather(local_out.last_hidden_state),
            pooler_output=_gather(local_out.pooler_output),
            hidden_states=_gather_seq(local_out.hidden_states),
            attentions=_gather_seq(local_out.attentions),
            attention_mask=_gather(local_out.attention_mask),
        )
```

python/sglang/multimodal_gen/runtime/models/encoders/base.py

决策中枢：定义 FOLD_MIN_HIDDEN_SIZE 和 DP_MIN_HIDDEN_SIZE 阈值，新增 _encoder_dims_divide、group_has_measured_topology、encoder_dp_capable、encoder_dp_worthwhile，重构 finalize_encoder_folding 使其接受 policy 参数。所有布局选择逻辑内聚于此。

```
# ---- 折叠（tensor-parallel）和数据并行（dp）的静态门控 ----

# 折叠仅在宽编码器上有利：T5-XXL (-20%)，窄编码器反效果
FOLD_MIN_HIDDEN_SIZE = 4096
# DP 在宽编码器上批处理有利（hidden >= 1024）
DP_MIN_HIDDEN_SIZE = 1024


def _encoder_dims_divide(config: EncoderConfig, group_size: int) -> bool:
    """编码器的 head 数和中间层大小能否被 group 整除——这是 fold 的硬性要求。"""
    _, heads, inter = _encoder_dims(config)
    return (
        group_size > 1
        and heads is not None and heads % group_size == 0
        and inter is not None and inter % group_size == 0
    )


def encoder_folding_worthwhile(config: EncoderConfig, group_size: int) -> bool:
    """只有在大小上值得折叠时才 fold。"""
    hidden, _, _ = _encoder_dims(config)
    return (
        _encoder_dims_divide(config, group_size)
        and hidden is not None and hidden >= FOLD_MIN_HIDDEN_SIZE
    )


def group_has_measured_topology(group) -> bool:
    """当前 group 的拓扑是否经过测量（单节点 NVLink）。

    fold/dp 的收益依赖高速互联；若非 NVLink，auto 模式回退到 replicate。
    """
    local_devices = torch.cuda.device_count()
    if group.world_size <= 1 or group.world_size > local_devices:
        return False
    return all(
        torch.cuda.can_device_access_peer(0, peer)
        for peer in range(1, group.world_size)
    )


def encoder_dp_capable(config: EncoderConfig) -> bool:
    """编码器宽度足够 DP 可能有益。"""
    hidden, _, _ = _encoder_dims(config)
    return hidden is not None and hidden >= DP_MIN_HIDDEN_SIZE


def encoder_dp_worthwhile(
    config: EncoderConfig, batch_size: int, measured_topology: bool
```

```
) -> bool:
    """运行时判断当前批处理是否值得用 DP。"""
    return measured_topology and batch_size > 1 and encoder_dp_capable(config)


def finalize_encoder_folding(
    config: EncoderConfig, policy: str = "auto", batched: bool = False
) -> None:
    """loader 在确定真实维度后调用，根据 policy 确定 fold/dp/replicate。
    `policy` 来自 --encoder-parallel，`batched` 表示当前 batching 容量 > 1。
    """
    if config.parallel_folding_mode is None:
        return
    group = get_folding_tp_group(config)
    # ...（折叠宽度校验和回退逻辑）
```

评论区精华

虽然没有逐行 review 评论，但 PR body 和提交历史记录了关键讨论：

- dp 默认静默无操作：最初 serve 默认 dp 但 batching_max_size 默认 1，导致 dp 永不触发。修复：自动提升至 replica_size，同时保持显式设置不变。
- dp 不应改变 DiT 批处理：作者发现 dp 自动提批可能改变 denoise 形状，影响输出。后续提交修复为 encoder_parallel=dp 不改变 batching_max_size，让 operator 自行决定。
- 拓扑限制：基于单节点 H100 NVLink 测试，fold/dp 收益依赖高速互联。group_has_measured_topology 仅在该拓扑下启用 auto，否则回退到 replicate。
- auto 决策所有权：最终设计是 auto 在加载时根据宽度和拓扑决定布局，而不是在入口点固定策略。
  - DP 默认值导致静默无操作 (correctness): 当 policy 为 dp 且 batching_max_size 未经显式设置时，将其自动提升为 replica_size。
  - DP 是否改变 DiT 批处理形状 (design): 修复为 dp 不改变 batching_max_size，operator 必须显式设置 --batching-max-size 才能启用 DP。
  - 拓扑限制：auto 模式仅对 NVLink 启用 (design): 新增 group_has_measured_topology 函数，通过 P2P 检测限制自动 fold/dp 仅对 NVLink 启用，否则回退到 replicate。
  - auto 决策所有权：由入口点还是加载时逻辑决定 (design): auto 拥有最终决策权；serve 的默认策略改为 auto（通过将 dp 默认回退到 auto 的逻辑），确保 fold 提案仍被考虑。

风险与影响

- 风险：
  1. 默认行为变更：serve 入口默认 encoder_parallel=dp，可能影响未指定 --batching-max-size 的老用户，但 batching_max_size 自动提升只发生在 dp 且未显式设置时，且经作者验证位精确。

2. 拓扑依赖：group_has_measured_topology 依赖 torch.cuda.can_device_access_peer(0, peer)，在多节点或非 NVLink 拓扑上 auto 会退化，可能导致性能低于预期，但不会出错。
3. 编码器互斥：fold 和 dp 在加载时固定，不能按请求切换。auto 会选其一，但可能不符合所有请求场景（例如既有延迟敏感场景又有吞吐场景）。
4. DP 条件复杂：encoder_dp_worthwhile 依赖 batch>1、measured_topology、hidden_size>=1024，边界情况可能意外降级。
5. 测试覆盖：新逻辑在 test_encoder_world_folding.py 中有纯逻辑测试，但缺少多 GPU 集成测试。 - 影响：用户：可通过 CLI 控制编码器并行策略，在批处理场景获得显著性能提升（batch=4 时 DP 减 31%）；auto 模式自动选择，降低心智负担。系统：改动集中在 diffusion 流水线，不影响其他部分。batching_max_size 的自动提升可能影响动态批处理行为，但仅在 dp 且未显式设置时。团队：需要维护两条并行路径（fold 和 dp），但决策逻辑内聚在 base.py，易于扩展。

- 风险标记：默认行为变更，性能收益依赖拓扑，编码器并行互斥，batching_max_size 联动，缺少多 GPU 集成测试

关联脉络

- PR #30086 [diffusion] encoder folding (baseline, stacked upon): 本 PR 基于 #30086 的 encoder folding 功能构建，承诺会 rebase。folding 基础功能已合并，本 PR 在其上增加 data-parallel 和统一 policy。