PR #23736 完整报告

sgl-project/sglang

[Diffusion] Refactor CFG Parallelism Framework to Support Multi-branch CFG for LTX2 Models

合并时间：2026-05-07 22:56

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23736

执行摘要

- 一句话：重构 CFG 并行框架，支持 LTX2 多分支无分类器引导
- 推荐动作：建议深度学习工程师精读本 PR，特别是 cfg_policy.py 和 cfg_parallel_utils.py 中的策略模式与 dispatch-allgather 架构，具有设计参考价值。在使用 LTX2.3 两阶段加速时，建议 --cfg-parallel-size 保持默认 2，避免大于分支数的配置触发已知 bug。生产环境应补充并行度 > 2 的测试覆盖。

功能与动机

Build on #23736's CFG-parallel abstraction so LTX2 can use multiple GPUs for classifier-free-guidance work, then add the missing LTX2.3 two-stage/HQ pieces needed for the no-loss acceleration path.

实现拆解

1. 引入 CFGPolicy 和 CFGBranch 抽象（cfg_policy.py）：新增 CFGBranch 数据类表示一个分支的前向规格（name、is_conditional、kwargs）；CFGPolicy 数据类负责构建分支列表（build()）和合并预测（combine()），默认实现标准双分支 CFG，支持并行 / 串行两种算术顺序以保持 bf16 兼容性。同时将归一化、rescale 等辅助函数提升为模块级函数供复用。
2. 实现通用 CFG 并行引擎（cfg_parallel_utils.py）：新增 run_cfg_parallel 函数，使用 dispatch_branches（round-robin）将分支分配给 CFG ranks，每个 rank 执行分配的 forward，通过 cfg_model_parallel_all_gather 收集结果并重排为分支顺序。空闲 rank（cfg_degree > N branches）运行 dummy forward 获取张量形状并贡献零。另提供 run_two_branch_cfg_parallel 简化的 all-reduce 路径。
3. 在 DenoisingStage 基类中集成 Policy（denoising.py）：在 DenoisingContext 中添加 cfg_policy 字段；修改 _prepare_denoising_loop 通过 pipeline_config.cfg_policy.build() 创建 policy；将原有的 _predict_noise_with_cfg 拆分为 _combine_cfg_serial 和 _combine_cfg_parallel，内部调用 policy.combine，移除了旧的 ad-hoc CFG parallel 实现和相关 helper。
4. 为 LTX2 模型添加多分支 CFG parallel 支持（ltx_2_denoising.py）：新增 parallelism_type 属性（返回 CFG_PARALLEL 或 REPLICATED）；新增 _combine_cfg_parallel_av 处理音视频两分支 all-reduce；新增 _ltx2_combine_guided_x0_parallel 实现 LTX2.3 两阶段 stage-1 的多分支（cond、neg、

perturbed、modality）加权 all-reduce；新增 _run_legacy_one_stage_multi_branch_cfg_parallel 兼容旧模型路径（但 review 指出空闲 rank 仍可能 IndexError）。对于 denoising_av.py 中的 second stage，返回 MAIN_RANK_ONLY 以避免冗余 forward。

5. 添加 CLI 参数并调整并行度计算（server_args.py）：新增 --cfg-parallel-size 参数，优先级高于 --enable-cfg-parallel；在 _adjust_parallelism 中自动设置 enable_cfg_parallel 并乘以 cfg_parallel_degree 计算 SP degree；移除了对 LTX 模型自动开启 CFG parallel 的排除（之前因性能不佳被排除，现在通过 CFG parallel 实现加速）。配套更新了单元测试（test_cfg_policy.py）、精度工具和性能基线。

关键文件：

- python/sglang/multimodal_gen/runtime/pipelines_core/stages/ltx_2_denoising.py（模块 去噪阶段；类别 source；类型 core-logic；符号 parallelism_type, _combine_cfg_parallel_av, _run_legacy_one_stage_multi_branch_cfg_parallel, _ltx2_combine_guided_x0_parallel）：LTX2 模型 CFG parallel 主要实现，包含多分支合并逻辑和空闲 rank 处理
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising.py（模块 去噪阶段；类别 source；类型 core-logic；符号 _rescale_noise_cfg, _apply_cfg_normalization, _apply_cfg_normalization_parallel, _apply_guidance_rescale_parallel）：基础 DenoisingStage 重构，集成 CFGPolicy，统一串行与并行 combine 逻辑
- python/sglang/multimodal_gen/runtime/distributed/cfg_parallel_utils.py（模块 并行引擎；类别 source；类型 core-logic；符号 _run, run_cfg_parallel, run_two_branch_cfg_parallel, dispatch_branches）：新增通用 CFG 并行引擎，提供 run_cfg_parallel 和 dispatch_branches
- python/sglang/multimodal_gen/runtime/distributed/cfg_policy.py（模块 策略层；类别 source；类型 core-logic；符号 CFGBranch, configure_batch, CFGPolicy, build）：新增 CFGPolicy/CFGBranch 抽象，定义灵活的多分支 CFG 策略
- python/sglang/multimodal_gen/runtime/server_args.py（模块 配置层；类别 source；类型 configuration；符号 cfg_parallel_degree, enable_cfg_parallel, _adjust_parallelism, _model_default_uses_cfg）：添加 --cfg-parallel-size 参数并调整自动并行度分配逻辑

关键符号：CFGPolicy.build, CFGPolicy.combine, CFGBranch.configure_batch, run_cfg_parallel, dispatch_branches, _combine_cfg_parallel_av, _ltx2_combine_guided_x0_parallel, _combine_cfg_parallel, _combine_cfg_serial, parallelism_type

关键源码片段

python/sglang/multimodal_gen/runtime/pipelines_core/stages/ltx_2_denoising.py

LTX2 模型 CFG parallel 主要实现，包含多分支合并逻辑和空闲 rank 处理

```
@staticmethod
def _combine_cfg_parallel_av(
```

```
    video: torch.Tensor,
    audio: torch.Tensor,
    guidance_scale: float,
    cfg_rank: int,
) -> tuple[torch.Tensor, torch.Tensor]:
    """All-reduce video and audio predictions across CFG ranks.

    Rank 0 (cond) contributes ``guidance_scale * pred``.
    Rank 1 (uncond) contributes ``(1 - guidance_scale) * pred``.
    Higher CFG ranks, if configured for multi-pass guidance, contribute
    zeros on the two-branch path.
    The sum reconstructs ``uncond + guidance_scale * (cond - uncond)``.
    """
    if cfg_rank == 0:
        video_partial = guidance_scale * video
        audio_partial = guidance_scale * audio
    elif cfg_rank == 1:
        video_partial = (1.0 - guidance_scale) * video
        audio_partial = (1.0 - guidance_scale) * audio
    else:
        # 当 CFG world size > 2 时，额外 rank 贡献零；但 reviewer 指出此逻辑
        # 在 size > 2 时可能仍不正确，因为所有非 0 rank 都接收 negative prompt
        video_partial = torch.zeros_like(video)
        audio_partial = torch.zeros_like(audio)
    return (
        cfg_model_parallel_all_reduce(video_partial),
        cfg_model_parallel_all_reduce(audio_partial),
    )
```

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

新增 CFGPolicy/CFGBranch 抽象，定义灵活的多分支 CFG 策略

```
from dataclasses import dataclass, field
from typing import Any, TYPE_CHECKING

if TYPE_CHECKING:
    from sglang.multimodal_gen.runtime.pipelines_core.schedule_batch import Req

@dataclass
class CFGBranch:
    """一个 CFG 分支的不可变规格。在去噪循环前构建，整个运行期间只读。"""

    name: str # 分支名称（如 "conditional", "unconditional"）
    is_conditional: bool # 是否为条件分支
    kwargs: dict[str, Any] # forward 所需的参数

    def configure_batch(self, batch: "Req") -> None:
        """设置该分支前向前的 batch 状态。"""
        batch.is_cfg_negative = not self.is_conditional
```

```
@dataclass
class CFGPolicy:
    """拥有一个生成运行的所有 CFG 分支，并负责合并它们的预测。

    在去噪循环前通过 ``build()`` 构建一次，然后所有步骤中只读。
    子类可重写 ``build()`` 和 ``combine()`` 以实现自定义 CFG 方案。
    """

    branches: list[CFGBranch] = field(default_factory=list)

    def build(
        self,
        batch: "Req",
        image_kwargs: dict[str, Any],
        pos_cond_kwargs: dict[str, Any],
        neg_cond_kwargs: dict[str, Any],
    ) -> "CFGPolicy":
        """返回一个填充了分支的新 policy。

        默认行为：如果 batch 启用 CFG，则创建 cond 和 unconditional 两个分支；
        否则只创建一个 cond 分支。
        """
        branches = [
            CFGBranch("conditional", True, {**image_kwargs, **pos_cond_kwargs})
        ]
        if batch.do_classifier_free_guidance:
            branches.append(
                CFGBranch("unconditional", False, {**image_kwargs, **neg_cond_kwargs})
            )
        return dataclasses.replace(self, branches=branches)

    def combine(
        self,
        predictions: list[torch.Tensor | tuple[torch.Tensor, ...]],
        batch: "Req",
        cfg_scale: float,
        pipeline_config: Any,
        *,
        cfg_parallel: bool = False,
    ) -> torch.Tensor | tuple[torch.Tensor, ...]:
        """合并分支预测为最终噪声估计。

        cfg_parallel=True 时使用并行友好的算术顺序（乘以权重然后相加），
         保持与旧 CFG parallel 实现的数值一致。
        默认为标准串行 CFG 公式。
        """
        if len(predictions) == 1:
            return predictions[0]
        pos_t = _wrap(predictions[0])
```

```
    neg_t = _wrap(predictions[1])
    if cfg_parallel:
        # 并行模式：scale * pos + (1 - scale) * neg
        results = [
            cfg_scale * p + (1 - cfg_scale) * n for p, n in zip(pos_t, neg_t)
        ]
    else:
        # 串行模式：neg + scale * (pos - neg)
        results = [n + cfg_scale * (p - n) for p, n in zip(pos_t, neg_t)]
    # 对第一个输出（噪声预测）应用后处理（归一化、rescale 等）
    results[0] = _apply_cfg_postprocess(
        results[0], pos_t[0], batch, pipeline_config
    )
    return _unwrap(tuple(results))
```

评论区精华

gemini-code-assist[bot] 指出了两个 high-priority 问题，均未在合入前修复：

- all-reduce 正确性：在 _combine_cfg_parallel_av 中，all_reduce 逻辑仅对 CFG world size 为 2 正确。若 size > 2，所有非 0 rank 会同时贡献 negative 分量，导致求和错误。
- 空闲 rank IndexError：在 _run_legacy_one_stage_multi_branch_cfg_parallel 中，空闲 rank 的 my_indices 为空，随后访问 local_videos[0] 会触发 IndexError。需要像 run_cfg_parallel 那样运行 dummy forward。
  - _combine_cfg_parallel_av all-reduce 逻辑仅支持 2 路并行 (correctness): 未修复，PR 已合并但未回应
  - 传统多分支路径空闲 rank IndexError (correctness): 未修复，PR 已合并但未回应

风险与影响

- 风险：
  - 核心路径正确性风险：_combine_cfg_parallel_av 和 _ltx2_combine_guided_x0_parallel 在并行度 > 2 时可能产生错误汇总结果（review 指出未修复）。
  - 空闲 rank 异常：_run_legacy_one_stage_multi_branch_cfg_parallel 中空闲 rank 的 IndexError 会影响传统一阶段流程，若用户设置并行度大于分支数可能触发。
  - 数值精度变化：CFG parallel 使用与串行不同的算术顺序（cfg_scale * pos + (1 - cfg_scale) * neg），虽测试中达标，但非 bit-exact，对精度敏感场景需验证。
  - 配置冲突：--cfg-parallel-size 与 --enable-cfg-parallel 同时出现时优先级明确，但可能造成用户困惑。
  - 性能浪费：若设置并行度远大于分支数，空闲 GPU 仍会运行 dummy forward 并参与 all-gather padding，造成资源浪费。
- 影响：

- 用户影响：LTX2.3 两阶段推理用户可获 35-38% 端到端加速，且可通过 --cfg-parallel-size N 灵活指定并行度。HQ 路径同样受益。
- 系统影响：CFGPolicy 框架可供其他扩散模型复用，降低添加新多分支 CFG 策略的成本。但需注意未修复缺陷对扩展性的限制。
- 团队影响：需要维护新的 CFG policy 层次和并行引擎，但代码结构更清晰，可扩展性更好。
- 风险标记：all-reduce 正确性，空闲 rank 异常，并行度匹配，数值精度变化，配置优先级

关联脉络

- 暂无明显关联 PR