

PR #39293 完整报告

sgl-project/sglang

[Diffusion] Clean up obsolete worker plumbing, dead helpers, and tests

合并时间: 2026-09-16 12:11

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39293>

执行摘要

- 一句话: 清理 Diffusion 死代码与 PipeFusion 遗留, 净删 1264 行
- 推荐动作: 值得精读, 尤其是 `parallel_state.py` 与 `group_coordinator.py` 的合并方式: 它展示了如何用通用实现替换专用实现并同时修复资源泄漏, 新增的 Gloo 生命周期回归测试是验证分布式资源释放的很好模板。对于维护 diffusion 模块的工程师, 建议关注 PP = 2 场景在合并后的通信行为, 以及被删除环境变量是否有外部依赖。

功能与动机

PR body 明确指出: Diffusion 仍创建没有任何 worker 消费的任务 / 结果管道, 同时携带旧 FastVideo/xDiT 的 PipeFusion 实现与当前 ZMQ/ 分布式调度器并存; 专门的 pipeline coordinator 还会覆盖其父级已分配的进程组, 导致继承的析构函数泄漏进程组。该 PR 延续 #32651 与 #37436 的基于证据的清理路线, 目标是移除死代码并消除进程组泄漏。

实现拆解

整个变更按四个步骤推进:

1. 移除 worker 管道并合并生成分支: `runtime/managers/scheduler.py` 的 `Scheduler.__init__` 删除 `task_pipes_to_slaves` 与 `result_pipes_from_slaves` 参数, 删除 `_broadcast_task` 与 `_collect_slave_results` 两个死方法;
`runtime/managers/gpu_worker.py` 的 `run_scheduler_process` 删除对应管道参数、断言与 master/slave 职责注释; `runtime/launch_server.py` 合并原本重复的 master/slave spawn 分支, 保留 readiness 管道、local/global rank 映射与 standalone/pool 两种 disaggregation GPU 映射。这样调度器回归到单一的 ZMQ 分布式路径, 消除两条代码路径的漂移风险。
2. 用通用 GroupCoordinator 替代专用 PipelineGroupCoordinator:
`runtime/distributed/group_coordinator.py` 删除约 430 行的 `PipelineGroupCoordinator` 类 (含 `reset_buffer`、`set_config`、`set_recv_buffer`、`set_extra_tensors_recv_buffer`、`skip` 张量收发与双 device group 等 PipeFusion 专用逻辑), 同时删除 `skip_rank` / `group_skip_rank` 属性, 并将 `send` / `recv` 的 `world_size == 2` 双组选择分支简化为单一 `device_group`。`runtime/distributed/parallel_state.py` 将 `_PP` 标注从 `PipelineGroupCoordinator` 改为 `GroupCoordinator`, `init_parallel_group_coordinator` 的 pipeline 分支不再走专用类, `get_pp_group` 返回类型同步调整; 同时删除 `_split_tensor_dict`、`_register_group`、`all_reduce`、`all_reduce_fake`、

is_the_same_node_as 等无人调用的分布式 helper。对应新增

test/unit/test_single_rank_device_group.py 中的

test_pipeline_group_releases_process_groups，用真实 Gloo 生命周期验证销毁后进程组回到基线。

3. 删除遗留 helper 与死代码：删除 models/dits/causal_wanvideo.py 的 _prepare_blockwise_causal_attn_mask (causal-Wan mask 构造器)、pipelines_core/stages/model_specific_stages/ltx_2/denoising.py 的 _run_legacy_one_stage_multi_branch_cfg_parallel (LTX-2 遗留 CFG 路径)、entrypoints/cli/utils.py 的 launch_distributed、layers/attention/selector.py 的 SGLANG_DIFFUSION_ATTENTION_BACKEND 环境变量入口及其 backend_name_to_enum / get_env_variable_attn_backend / record_component_attn_backend、layers/attention/backends/attention_backend.py 的 asdict_zerocopy 与 AttentionLayer Protocol，以及 common.py 的 configure_ipv6、gpu_worker.py 的 _default_workload_for_hint、layernorm.py 的 _get_weight_fallback、layers/utils.py 的 get_token_bin_counts_and_mask、longlive2.py 的 expand_longlive2_shot_prompts。每次删除前均检查调用者、导入、注册表和动态分发路径。
4. 删除冗余测试并保留回归覆盖：删除 test_model_catalog.py (AST/regex 解析 model catalog 的镜像测试)、test_utility_ownership.py (模型导入所有权检查)、test_qwen3_encoder.py 中的 Qwen norm 类检查，以及 test_transformer_loader_fallback.py 中两个已被完整 loader 入口覆盖的 fallback 用例；保留数值、加载、状态、API 与分布式回归测试，并新增进程组生命周期回归用例。

配套验证包括 H200 上的完整 diffusion 单元套件 (2,894 通过 / 113 跳过 / 8 失败，与 main 一致)、真实 Gloo 生命周期测试、pre-commit 与全量符号引用审计。

关键文件：

- python/sglang/multimodal_gen/runtime/distributed/group_coordinator.py (模块 组协调器；类别 source；类型 dependency-wiring；符号 skip_rank, group_skip_rank, PipelineGroupCoordinator, init)：核心变更文件：删除约 430 行的专用 PipelineGroupCoordinator 类，简化 send/recv 的 device group 选择，并移除 skip_rank/group_skip_rank 等 PipeFusion 专用属性，是进程组泄漏修复的主战场。
- python/sglang/multimodal_gen/runtime/distributed/parallel_state.py (模块 并行状态；类别 source；类型 dependency-wiring；符号 _split_tensor_dict, _register_group, all_reduce, all_reduce_fake)：与 group_coordinator.py 配对的核心文件：将 _PP 类型从 PipelineGroupCoordinator 改为 GroupCoordinator，pipeline 分支统一走通用构造路径，并删除一批无人调用的分布式 helper。
- python/sglang/multimodal_gen/runtime/launch_server.py (模块 启动入口；类别 source；类型 core-logic)：合并 dispatcher 生成 master/slave 分支，删除了管道创建与双路启动逻辑，是启动路径收敛的入口。
- python/sglang/multimodal_gen/runtime/managers/scheduler.py (模块 调度器；类别 source；类型 core-logic；符号 _broadcast_task, _collect_slave_results)：调度器构造函数删除 task/result 管道参数，并移除 _broadcast_task 与 _collect_slave_results 两个

死方法，回归单一 ZMQ 调度路径。

- python/sglang/multimodal_gen/runtime/managers/gpu_worker.py (模块 工作进程; 类别 source; 类型 core-logic; 符号 _default_workload_for_hint) : run_scheduler_process 移除管道参数与断言，并删除 per-request residency hint 的缓存逻辑 _default_workload_for_hint。
- python/sglang/multimodal_gen/test/unit/test_single_rank_device_group.py (模块 分布式测试; 类别 test; 类型 test-coverage; 符号 TestSingleRankDeviceGroup, _check_pipeline_group_lifecycle, test_pipeline_group_releases_process_groups) : 新增 test_pipeline_group_releases_process_groups，用真实 Gloo 生命周期验证进程组释放，是本次唯一新增的回归测试，也是修复泄漏的关键证据。
- python/sglang/multimodal_gen/runtime/models/dits/causal_wanvideo.py (模块 模型实现; 类别 source; 类型 data-contract; 符号 _prepare_blockwise_causal_attn_mask, attention_mask) : 删除未使用的 causal-Wan blockwise 因果 mask 构造器 _prepare_blockwise_causal_attn_mask 及 create_block_mask 导入。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/ltx_2/denoising.py (模块 去噪阶段; 类别 source; 类型 data-contract; 符号 _run_legacy_one_stage_multi_branch_cfg_parallel) : 删除不可达的 legacy LTX-2 单阶段多分支 CFG 并行实现 _run_legacy_one_stage_multi_branch_cfg_parallel，并清理相关 all_gather 导入。
- python/sglang/multimodal_gen/runtime/layers/attention/selector.py (模块 注意力选择; 类别 source; 类型 core-logic; 符号 backend_name_to_enum, get_env_variable_attn_backend, record_component_attn_backend) : 删除 SGLANG_DIFFUSION_ATTENTION_BACKEND 环境变量入口及 backend_name_to_enum、get_env_variable_attn_backend、record_component_attn_backend，改变了 attention 后端选择的外部可配置面。
- python/sglang/multimodal_gen/runtime/layers/attention/backends/attention_backend.py (模块 注意力后端; 类别 source; 类型 dependency-wiring; 符号 asdict_zero_copy, AttentionLayer, forward) : 删除 AttentionMetadata.asdict_zero_copy 与 AttentionLayer Protocol，清理无人引用的抽象契约。
- python/sglang/multimodal_gen/runtime/entrypoints/cli/utils.py (模块 CLI 工具; 类别 source; 类型 core-logic; 符号 launch_distributed) : 删除过时的 launch_distributed，该函数仍指向已不存在的 sgl_diffusion/sample 脚本路径。
- python/sglang/multimodal_gen/test/unit/test_model_catalog.py (模块 模型目录测试; 类别 test; 类型 deletion; 符号 _registered_model_ids, _catalog_model_ids, test_explicit_registry_model_ids_are_documented) : 整体删除 AST/regex 解析的 model catalog 镜像测试，属于测试覆盖收缩，需关注文档一致性检测能力的丢失。

关键符号: init_parallel_group_coordinator, send, recv, get_pp_group, run_scheduler_process, _broadcast_task, _collect_slave_results, launch_distributed, _prepare_blockwise_causal_attn_mask, _run_legacy_one_stage_multi_branch_cfg_parallel, test_pipeline_group_releases_process_groups

关键源码片段

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

核心变更文件：删除约 430 行的专用 PipelineGroupCoordinator 类，简化 send/recv 的 device group 选择，并移除 skip_rank/group_skip_rank 等 PipeFusion 专用属性，是进程组泄漏修复的主战场。

```
# python/sglang/multimodal_gen/runtime/distributed/group_coordinator.py
# 删除 PipeFusion 专用协调器后，点对点收发统一走单个 device_group:
# 旧实现为 world_size == 2 额外创建 0->1 与 1->0 两个独立组来避免通信停滞，
# 该技巧只服务于 PipeFusion 的流水线调度，已随之一并移除。
```

```
def send(self, tensor: torch.Tensor, dst: Optional[int] = None) -> None:
    """Send a tensor to the destination rank."""
    # NOTE: `dst` 是组内编号 (rank_in_group 语义)，这里映射到全局 rank。
    if dst is None:
        dst = self.group_next_rank
    torch.distributed.send(tensor, self.ranks[dst], group=self.device_group)
```

```
def recv(
    self, size: torch.Size, dtype: torch.dtype, src: Optional[int] = None
) -> torch.Tensor:
    """Receive a tensor from the src rank."""
    # NOTE: `src` 同样是组内编号，实际全局 rank 由 self.ranks 转换。
    if src is None:
        src = self.group_prev_rank
    tensor = torch.empty(size, dtype=dtype, device=self.device)
    torch.distributed.recv(tensor, self.ranks[src], group=self.device_group)
    return tensor
```

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

与 group_coordinator.py 配对的核心文件：将 _PP 类型从 PipelineGroupCoordinator 改为 GroupCoordinator，pipeline分支统一走通用构造路径，并删除一批无人调用的分布式helper。

```
# python/sglang/multimodal_gen/runtime/distributed/parallel_state.py
# pipeline 并行不再构造专用 PipelineGroupCoordinator，而是复用通用
# GroupCoordinator：旧类会覆盖父级已分配的进程组，且析构时没有完整
# 释放自建组，造成进程组泄漏。统一类型后 destroy() 走基类实现，
# 组生命周期回归测试见 test_single_rank_device_group.py。
```

```
def init_parallel_group_coordinator(
    group_ranks: List[List[int]],
    local_rank: int,
    backend: str,
    parallel_mode: str,
    **kwargs,
) -> GroupCoordinator:
    """按并行模式返回对应的组协调器。"""
```

```

assert parallel_mode in [
    "data",
    "pipeline",
    "tensor",
    "sequence",
    "classifier_free_guidance",
    "vae_decode",
    "replica",
    "encoder_data",
], f"parallel_mode {parallel_mode} is not supported"

# sequence 仍是特例：需要 ulysses / ring 两组元数据。
if parallel_mode == "sequence":
    return SequenceParallelGroupCoordinator(
        group_ranks=group_ranks,
        local_rank=local_rank,
        torch_distributed_backend=backend,
        group_name="sp_group",
        **kwargs,
    )

# 其余模式（含 pipeline）统一走 GroupCoordinator；pipeline 不再
# 创建额外的 skip 资源与双 device group，通信语义由基类提供。
group_name = {
    "pipeline": "pp_group",
    "tensor": "tp_group",
    "vae_decode": "vae_decode_group",
    "replica": "replica_group",
    "encoder_data": "encoder_dp_group",
}.get(parallel_mode, "cfg_group")
return GroupCoordinator(
    group_ranks=group_ranks,
    local_rank=local_rank,
    torch_distributed_backend=backend,
    use_device_communicator=parallel_mode != "tensor",
    use_srt_custom_allreduce=parallel_mode == "tensor",
    group_name=group_name,
)

```

评论区精华

该 PR 没有 reviewer 评论与审核线程（review_comments_count = 0），唯一的 2 条评论均为 CI 重跑命令 [/tag-and-rerun-ci](#)（分别来自 mickqian 与 BBuf），不涉及技术讨论。技术论证主要通过 PR body 的验证表格展开：新增的生命周期回归测试在 main 上失败、在本分支通过；main 会遗留 4 个进程组而本分支只保留初始的 1 个；H200 上连续三轮 create / all-reduce / nested tensor-dict broadcast / point-to-point / destroy 循环后进程组计数回到基线。

- 暂无高价值评论线程

风险与影响

- 风险：主要风险集中在三处：

1. 并行组行为变更：`group_coordinator.py` 的 `send / recv` 移除了 `world_size == 2` 时的双 device group 选择逻辑，统一使用单一 `device_group`；这对 `PP = 2` 场景的通信路径有实际行为变化，若 PipeFusion 之外仍有隐藏调用方依赖旧双组语义，可能出现行为差异。PR 通过符号引用审计与 Gloo 生命周期测试降低了该风险，但其他并行后端（NCCL 之外）未验证。
2. 环境变量入口被删除：`SGLANG_DIFFUSION_ATTENTION_BACKEND` 及相关枚举解析函数被移除，若有外部部署脚本依赖该变量强制选择 attention 后端，会静默回落到自动选择路径。
3. 测试覆盖收缩：删除了 5 个测试用例，其中 model catalog 镜像测试的删除意味着未来新增注册模型不再有自动化的文档一致性检查；虽然声称已由完整 loader 入口覆盖 fallback 用例，但检测漂移的能力下降。

此外，H200 之外的其他加速器后端与多主机网络未在本 PR 中验证；若某平台在动态导入路径中仍引用被删 helper，只会在特定环境暴露。

- 影响：影响范围集中在 `sglang/multimodal_gen` (diffusion 子系统)，净删 1264 行。对外部用户而言，模型前向、kernel、检查点格式与质量门均无变化，不产生精度或吞吐承诺；对系统而言，调度器启动路径从双分支收敛为单一路径，并行组生命周期更干净，进程组泄漏被修复。对团队而言，本 PR 确立了 "先审计调用者 / 导入 / 注册 / 动态分发，再删除" 的清理范式，并为后续继续剥离 FastVideo/xDiT 遗留代码（#32651、#37436 系列）铺路。
- 风险标记：大范围死代码删除（净删 1264 行），并行组行为变更（`world_size == 2`），删除环境变量开关，移除部分回归测试

关联脉络

- 暂无明显关联 PR