PR #34064 完整报告

sgl-project/sglang

[Diffusion] Optimize ordinary model weight loading

合并时间：2026-08-10 20:07

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34064

执行摘要

- 一句话：优化 DiT 权重加载：CPU 直载免拷贝，新增 GPU 直载开关
- 推荐动作：值得精读。重点看 _can_assign_cpu_tensor_without_copy 的边界条件与 load_model_from_full_model_state_dict 的分支设计，以及 opt-in 参数三层校验（CLI、server_args、transformer_loader）的取舍。对在 diffusion 场景做启动优化或显存权衡参数化的后续 PR 有直接借鉴价值。

功能与动机

PR body 明确说明：普通 TP=1 diffusion 加载路径付出了不必要的 device 转移与 tensor 拷贝成本，尤其在 DiT 被配置为 CPU offload 时；更激进的 direct-to-GPU 加载能进一步缩短部分模型的冷启动时间，但会显著提高瞬时 GPU 显存且并非普遍更快。因此该 PR 采用“默认低风险优化 + opt-in 激进开关”的双轨策略，并明确拒绝量化、FSDP、TP>1 等不兼容组合。

实现拆解

变更入口在 python/sglang/multimodal_gen/runtime/loader/component_loaders/transformer_loader.py 的 load_customized，随后向 loader 与 server_args 两侧扩散：

1. checkpoint 落盘设备决策：新增 _resolve_checkpoint_load_device，将“未量化 + dit_cpu_offload”组合的 checkpoint 直接解析为 cpu，量化路径仍落 runtime device（量化权重需要 GPU 后处理）。该设备作为 WeightLoadPlan.checkpoint_load_device 传入后续加载。
2. CPU 零拷贝赋值：fsdp_load.py 新增 _can_assign_cpu_tensor_without_copy，在 load_model_from_full_model_state_dict 中命中该函数时，直接把 checkpoint 张量赋给参数数据（sharded_tensor = full_tensor），跳过 torch.empty_like + weight_loader 的整张拷贝；否则保持原有 _make_param_like 路径。
3. opt-in GPU 直载：server_args.py 新增 direct_gpu_weight_loading 字段、--direct-gpu-weight-loading CLI 参数与 _validate_direct_gpu_weight_loading（仅 CUDA、非 offload、非 FSDP、TP=1）；transformer_loader.load_customized 将其透传为 WeightLoadPlan.for_component(load_full_state_dict_on_device=True)，并拒绝带量化配置的组合；fsdp_load.maybe_load_fsdp_model 在该标志下跳过 rank_local_checkpoint 预读，改用带 weight_load_plan 的 safetensors_weights_iterator 直接把完整 state dict 映射到 GPU。

4. 测试与文档配套：test_fsdp_load.py 新增 TestOrdinaryWeightLoading（验证 direct device 路径跳过 rank local 预读、data_ptr 零拷贝断言）；test_transformer_quant.py 覆盖 _resolve_checkpoint_load_device 四种组合与 WeightLoadPlan 新字段；test_server_args.py 验证 CLI 默认关闭与非法组合报错；cli.mdx 补充权衡说明。

关键文件：

- python/sglang/multimodal_gen/runtime/loader/fsdp_load.py（模块 权重加载；类别 source；类型 core-logic；符号 _can_assign_cpu_tensor_without_copy, load_model_from_full_model_state_dict, maybe_load_fsdp_model）：变更核心：新增零拷贝判定函数并改造加载主路径分支。
- python/sglang/multimodal_gen/runtime/loader/component_loaders/transformer_loader.py（模块 加载决策；类别 source；类型 core-logic；符号 _resolve_checkpoint_load_device, load_customized）：加载入口：决定 checkpoint 落盘设备并透传 direct GPU 标志。
- python/sglang/multimodal_gen/runtime/server_args/server_args.py（模块 服务参数；类别 source；类型 configuration；符号 direct_gpu_weight_loading, _validate_direct_gpu_weight_loading）：新增 --direct-gpu-weight-loading 参数及合法性校验，是 opt-in 行为的控制面。
- python/sglang/multimodal_gen/runtime/loader/weight_load_plan.py（模块 加载计划；类别 source；类型 data-contract；符号 WeightLoadPlan, WeightLoadPlan.for_component）：数据契约扩展：新增 load_full_state_dict_on_device 字段贯通加载计划。
- python/sglang/multimodal_gen/test/unit/test_fsdp_load.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestOrdinaryWeightLoading, test_direct_device_loading_skips_rank_local_cpu_checkpoint, test_tp1_unquantized_linear_assigns_checkpoint_tensor_without_copy）：核心行为测试：验证 direct device 加载跳过 rank-local 预读，并以 data_ptr 断言零拷贝赋值。
- python/sglang/multimodal_gen/test/unit/test_transformer_quant.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_weight_load_plan_can_keep_full_state_dict_on_device, test_unquantized_cpu_offload_loads_checkpoint_on_cpu, test_quantized_cpu_offload_keeps_checkpoint_on_runtime_device, test_resident_transformer_loads_checkpoint_on_runtime_device）：覆盖设备决策矩阵与 WeightLoadPlan 新字段，保障量化路径不回退。
- python/sglang/multimodal_gen/test/unit/test_server_args.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestDirectGpuWeightLoading, test_cli_defaults_off_and_parses_explicit_enable, test_rejects_cpu_offload_fsdp_and_tp）：验证新 CLI 参数默认关闭及非法组合报错。
- docs/docs/sglang-diffusion/api/cli.mdx（模块 文档；类别 docs；类型 entrypoint）：向部署者说明 direct GPU loading 的显存与启动时间权衡。

关键符号：_can_assign_cpu_tensor_without_copy, load_model_from_full_model_state_dict, maybe_load_fsdp_model, _resolve_checkpoint_load_device, _validate_direct_gpu_weight_loading,

WeightLoadPlan.for_component

关键源码片段

python/sglang/multimodal_gen/runtime/loader/fsdp_load.py

变更核心：新增零拷贝判定函数并改造加载主路径分支。

```
# 新增的零拷贝判定函数：只有 TP=1 的未量化标准 linear 才可能免拷贝
def _can_assign_cpu_tensor_without_copy(
    actual_param: torch.nn.Parameter,
    full_tensor: torch.Tensor,
    target_param: torch.Tensor,
) -> bool:
    """Return whether a TP=1 linear loader would only copy this CPU tensor."""
    # checkpoint 张量本身必须在 CPU 上，才有直接复用的前提
    if full_tensor.device.type != "cpu":
        return False
    # 通过参数上挂载的 weight_loader 反查宿主 linear 层
    weight_loader = actual_param.__dict__.get("weight_loader")
    if not isinstance(weight_loader, MethodType):
        return False

    owner = weight_loader.__self__
    # 仅支持三类标准 linear，且必须是未量化路径
    if not isinstance(
        owner,
        (ReplicatedLinear, ColumnParallelLinear, RowParallelLinear),
    ):
        return False
    if not isinstance(owner.quant_method, UnquantizedLinearMethod):
        return False
    # TP 切分过的 Column/Row parallel 层仍需 weight_loader 计算分片
    if not isinstance(owner, ReplicatedLinear) and owner.tp_size != 1:
        return False
    # 自定义 Parameter 子类或带特殊标记的参数不能直接复用 checkpoint 张量
    if type(actual_param) is not nn.Parameter:
        return False
    if any(
        actual_param.__dict__.get(attribute, False)
        for attribute in ("is_metadata", "is_sharded_weight", "needs_scalar_to_array")
    ):
        return False
    # 最后一道防线：shape 与 dtype 必须完全一致
    return full_tensor.shape == target_param.shape and full_tensor.dtype == target_param.dtype


# load_model_from_full_model_state_dict 中的赋值分支（节选整理）
if weight_loader is not None:
    assert actual_param is not None
```

```
    if _can_assign_cpu_tensor_without_copy(actual_param, full_tensor, meta_sharded_param):
        # 命中零拷贝条件时，直接把 checkpoint 张量作为参数数据
        sharded_tensor = full_tensor
    else:
        # 原有路径：分配目标 shape/dtype 的空张量，再由 weight_loader 填充
        sharded_tensor = torch.empty_like(
            meta_sharded_param,
            device=checkpoint_load_device,
            dtype=target_dtype,
        )
        requires_grad = getattr(meta_sharded_param, "requires_grad", False)
        temp_param = _make_param_like(actual_param, sharded_tensor)
        if not (sharded_tensor.is_floating_point() or sharded_tensor.is_complex()):
            requires_grad = False
        temp_param.requires_grad = requires_grad
        try:
            weight_loader(temp_param, full_tensor)
        except AssertionError as exc:
            raise AssertionError(
                f"Failed to shard/load parameter {target_param_name}: "
                f"full_tensor.shape={tuple(full_tensor.shape)}, "
                f"meta_sharded_param.shape={tuple(meta_sharded_param.shape)}, "
                f"temp_param.shape={tuple(temp_param.shape)}, "
                f"param_cls={type(actual_param).__name__}"
            ) from exc
        sharded_tensor = temp_param.data
```

python/sglang/multimodal_gen/runtime/loader/component_loaders/transformer_loader.py

加载入口：决定 checkpoint 落盘设备并透传 direct GPU 标志。

```
# 新增的 checkpoint 加载设备决策：未量化 + CPU offload 时直接落 CPU
def _resolve_checkpoint_load_device(
    runtime_device: torch.device,
    *,
    component_cpu_offload: bool,
    runtime_quant_config: object | None,
) -> torch.device:
    if component_cpu_offload and runtime_quant_config is None:
        return torch.device("cpu")
    # 量化权重通常需要在 GPU 上后处理，checkpoint 仍放 runtime device
    return runtime_device


# load_customized 中构造加载计划的片段（节选整理）
local_torch_device = get_local_torch_device()
checkpoint_load_device = _resolve_checkpoint_load_device(
    local_torch_device,
    component_cpu_offload=bool(component_server_args.dit_cpu_offload),
```

```
        runtime_quant_config=quant_spec.runtime_quant_config,
    )
    direct_gpu_weight_loading = bool(component_server_args.direct_gpu_weight_loading)
    # opt-in 的 GPU 直载只支持未量化 checkpoint，避免量化后处理与整 dict 上 GPU 冲突
    if direct_gpu_weight_loading and quant_spec.runtime_quant_config is not None:
        raise ValueError("--direct-gpu-weight-loading supports only unquantized DiT checkpoints")
    weight_load_plan = WeightLoadPlan.for_component(
        checkpoint_load_device=checkpoint_load_device,
        needs_device_weight_postprocess=quant_spec.needs_device_weight_postprocess,
        component_cpu_offload=bool(component_server_args.dit_cpu_offload),
        load_full_state_dict_on_device=direct_gpu_weight_loading,
    )
```

评论区精华

该 PR 没有实质的 review 评论或讨论线程（review 评论为 0，issue 侧仅有作者两次 /tag-and-rerun-ci 触发 CI 重跑）。所有设计权衡（默认低内存优化 vs opt-in 显存换速度）均由作者在 PR body 中以 B200 三模型 benchmark 形式自证，未出现评审交锋。

- 暂无高价值评论线程

风险与影响

- 风险：
  - 零拷贝判定脆弱性（fsdp_load.py）：_can_assign_cpu_tensor_without_copy 依赖 weight_loader 的 MethodType 检测与参数私有属性（is_metadata、is_sharded_weight、needs_scalar_to_array）。未来新增 linear 类型或量化方法时若不同步维护，可能静默退回拷贝路径（安全）或错误复用张量（数据竞争风险）。当前测试仅直接覆盖 ReplicatedLinear，ColumnParallelLinear/RowParallelLinear 走的是间接断言。
  - 默认路径行为变化：未量化 + CPU offload 的 checkpoint 落盘设备从 runtime device 改为 CPU，影响所有 diffusion 模型的冷启动内存与耗时分布。虽然给出 Z-Image-Turbo、Qwen-Image、Wan2.1 三模型结果，但覆盖仍有限。
  - opt-in 高显存峰值：--direct-gpu-weight-loading 下 Qwen-Image 峰值显存从约 48 GB 升至 84 GB（近翻倍），显存紧张环境误开会直接 OOM；校验仅覆盖 CUDA/offload/FSDP/TP 组合，未做显存容量预检。
  - 校验逻辑双处存在：_validate_direct_gpu_weight_loading（server_args 层）与 transformer_loader.load_customized 内嵌检查都拦截量化 +direct 组合，未来单侧更新可能导致行为不一致。
  - 影响：对用户：diffusion 部署冷启动时间显著缩短（Qwen-Image 减 44.8%，Z-Image-Turbo 减 18.4%），默认路径不增加峰值显存；opt-in 路径面向追求极致启动速度且显存充裕的用户。对系统：加载管线新增 WeightLoadPlan.load_full_state_dict_on_device 语义，FSDP 与 rank-local 预读路径被条件化，safetensors_weights_iterator 开始接收加载计划参数。对团队：提供了一个“显存换启动速度”的开关范式，与 34173 的 torch.compile opt-in 形成一致的参数治理风格。

- 风险标记：条件零拷贝依赖内部属性，默认加载路径行为变更，opt-in 路径显存近翻倍，校验逻辑双处维护

关联脉络

- PR #34173 [diffusion] Make torch.compile opt-in for speed mode: 同为 diffusion 启动 / 部署参数的 opt-in 治理，改动 server_args 与 cli.mdx，参数风格一致。
- PR #34248 [diffusion] expose architecture config at the DiT runtime boundary: 同属 diffusion 运行时边界重构，与加载计划 WeightLoadPlan 的语义演进相关。