PR #34304 完整报告

sgl-project/sglang

Remove the torchao integration (--torchao-config)

合并时间：2026-08-14 21:49

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34304

执行摘要

- 一句话：移除失效的 torchao 集成与 --torchao-config 参数，精简量化链路
- 推荐动作：该 PR 值得快速浏览作为 " 功能移除 " 的范例：体现了在依赖升级导致功能失效后，直接删除而不是保留死代码的决策逻辑。对于关心量化功能现状的读者，重点看 server_args.py 中参数删除方式、loader.py 中分层加载分支的收敛，以及 docs/docs/advanced_features/quantization.mdx 中支持矩阵的更新。不建议作为架构参考深入精读。

功能与动机

PR body 明确引用 issue #34295：自 torchao pin 升级到 0.17.0 后，--torchao-config 对每个可接受值都会抛 ImportError，因此没有任何可用的功能需要走弃用流程，直接移除更合理。这既清理了死代码，也消除了一个长期误导用户的 CLI 参数和依赖负担。

实现拆解

该 PR 的变更可拆解为 5 步：

1. 删除核心工具模块：移除 python/sglang/srt/layers/torchao_utils.py（95 行），其中包含 apply_torchao_config_to_model、proj_filter、proj_filter_conv3d 等全部 torchao 量化入口。
2. 主流程去耦合：
   - python/sglang/srt/model_loader/loader.py 的 LayeredModelLoader.load_model 删除 import、torchao_config = get_exec().graph.torchao_config、fill_module 内按 fqn 含 proj 即量化的分支，以及 model.torchao_applied = True 标记。
   - python/sglang/srt/model_executor/model_runner.py 的 maybe_apply_post_load_model_transforms 删除 torchao_applied 检查和 apply_torchao_config_to_model 调用，并移除模块级 import。
3. 参数与依赖契约收敛：
   - python/sglang/srt/server_args.py 删除 torchao_config 字段（位于 exec.graph 命名空间），同时更新 # Torch compile and torchao 注释为 # Torch compile。
   - python/sglang/check_env.py 的依赖列表移除 torchao；python/pyproject.toml、pyproject_cpu.toml、pyproject_npu.toml、pyproject_xpu.toml、pyproject_other.toml、3rdparty/amd/wheel/sglang/pyproject.toml 均移除对应依赖声

明；docker/xpu.Dockerfile 移除安装步骤。

4. 模型量化分支修正：python/sglang/srt/models/paddleocr_vl.py 将可量化判断从 ["bitsandbytes", "torchao"] 收缩为 "bitsandbytes"，避免残留引用。
5. 测试与文档清理：删除 test/manual/quant/test_torchao.py（92 行），删除 test/manual/test_srt_engine_with_quant_args.py 中的 test_2_torchao_args 方法；python/sglang/test/runners.py 移除 torchao_config 构造参数。文档侧更新 docs/docs/advanced_features/quantization.mdx、server_arguments.mdx、nvidia_jetson.mdx、ascend-npus/reference/support_features.mdx，移除 torchao 相关说明和支持矩阵行。

关键文件：

- python/sglang/srt/layers/torchao_utils.py（模块 量化工具；类别 source；类型 deletion；符号 proj_filter, proj_filter_conv3d, apply_torchao_config_to_model）：torchao 集成的核心入口，整个文件（95 行）被删除，包含 apply_torchao_config_to_model 和两个过滤函数，是本次移除的主目标。
- test/manual/quant/test_torchao.py（模块 量化测试；类别 test；类型 deletion；符号 TestTorchAO, TestTorchAOForVLM, test_throughput, test_vlm_generate）：torchao 功能的手动量化测试，随功能一并删除，避免保留无法运行的用例。
- python/sglang/srt/model_loader/loader.py（模块 模型加载；类别 source；类型 data-contract；符号 LayeredModelLoader.load_model, fill_module）：LayeredModelLoader.load_model 中按层量化分支被移除，是 torchao 在模型加载主路径上的唯一耦合点。
- python/sglang/srt/model_executor/model_runner.py（模块 模型执行；类别 source；类型 data-contract；符号 maybe_apply_post_load_model_transforms）：maybe_apply_post_load_model_transforms 是加载后模型变换的入口，删除 torchao 应用逻辑后该转换链更干净。
- python/sglang/srt/server_args.py（模块 服务参数；类别 source；类型 core-logic；符号 ServerArgs.torchao_config）：删除 torchao_config 字段，这是对外 CLI 契约的关键变更点，直接影响用户启动参数。
- python/sglang/srt/models/paddleocr_vl.py（模块 多模态；类别 source；类型 data-contract）：量化可用性判断中移除 torchao，避免模型层残留引用影响后续加载流程。
- python/sglang/check_env.py（模块 环境检查；类别 source；类型 core-logic）：环境依赖检查列表移除 torchao，避免用户被误导以为该依赖仍被支持。
- python/sglang/test/runners.py（模块 测试运行；类别 test；类型 test-coverage）：测试 runner 构造参数移除 torchao_config 转发，保证测试侧与 CLI 契约一致。
- test/manual/test_srt_engine_with_quant_args.py（模块 量化参数；类别 test；类型 test-coverage；符号 test_2_torchao_args）：删除 test_2_torchao_args 用例，是量化参数测试集中专门针对 torchao 的覆盖点。
- docs/docs/advanced_features/quantization.mdx（模块 量化文档；类别 docs；类型 documentation）：量化文档移除 torchao 在线量化章节和支持矩阵行，避免文档继续引导用户使用已失效功能。

- python/pyproject.toml（模块 依赖配置；类别 config；类型 configuration）：主包依赖清单移除 torchao，是依赖收敛的代表性配置文件。
- docker/xpu.Dockerfile（模块 镜像构建；类别 infra；类型 infrastructure）：XPU 镜像构建步骤中移除 torchao 安装，保证多平台镜像与依赖声明一致。

关键符号：apply_torchao_config_to_model, proj_filter, proj_filter_conv3d, maybe_apply_post_load_model_transforms, test_2_torchao_args

关键源码片段

python/sglang/srt/layers/torchao_utils.py

torchao 集成的核心入口，整个文件（95 行）被删除，包含 apply_torchao_config_to_model 和两个过滤函数，是本次移除的主目标。

```
# 该文件随 PR#34304 整体删除：torchao 固定到 0.17.0 后，
# --torchao-config 的每个取值都会在 import 阶段抛 ImportError，
# 因此不再需要这套量化工具。

import logging
from typing import Callable, Optional

import torch

logger = logging.getLogger(__name__)


def proj_filter(module: torch.nn.Module, fqn: str):
    """过滤函数：仅量化名字中含 ``proj`` 的投影层。"""
    return "proj" in fqn


def proj_filter_conv3d(module: torch.nn.Module, fqn: str):
    """过滤函数：跳过 Conv3d，避免量化不支持的卷积层。"""
    if isinstance(module, torch.nn.Conv3d):
        logger.warning(f"Quantize: skipping {fqn} because it's a Conv3d")
        return False
    return "proj" in fqn


def apply_torchao_config_to_model(
    model: torch.nn.Module,
    torchao_config: str,
    filter_fn: Optional[Callable] = proj_filter,
):
    """按配置字符串对模型应用 torchao 量化。

    支持 int8wo / int8dq / int4wo-<group_size> / fp8wo / fp8dq-*，
    全部在此统一入口分派；删除后服务端不再具备该能力。
```

```
    """
    if torchao_config == "" or torchao_config is None:
        return model

    # 惰性导入原本用于抑制 torchao 的启动告警
    from torchao.quantization import (
        float8_dynamic_activation_float8_weight,
        float8_weight_only,
        int4_weight_only,
        int8_dynamic_activation_int8_weight,
        int8_weight_only,
        quantize_,
    )
    from torchao.quantization.observer import PerRow, PerTensor

    if "int8wo" in torchao_config:
        quantize_(model, int8_weight_only(), filter_fn=proj_filter_conv3d)
    elif "int8dq" in torchao_config:
        quantize_(model, int8_dynamic_activation_int8_weight(), filter_fn=filter_fn)
    elif "int4wo" in torchao_config:
        group_size = int(torchao_config.split("-")[-1])
        assert group_size in [32, 64, 128, 256], \
            f"int4wo groupsize needs to be one of [32, 64, 128, 256] but got {group_size}"
        quantize_(model, int4_weight_only(group_size=group_size), filter_fn=filter_fn)
    elif "fp8wo" in torchao_config:
        # 需要较新硬件：fp8e4nv 在 CUDA arch < 89 上不支持
        quantize_(model, float8_weight_only(), filter_fn=proj_filter_conv3d)
    elif "fp8dq" in torchao_config:
        granularity = torchao_config.split("-")[-1]
        granularity_map = {
            "per_row": PerRow(),
            "per_tensor": PerTensor(),
        }
        assert granularity in granularity_map, \
            f"Supported granularity are: {granularity_map.keys()}, got {granularity}"
        quantize_(
            model,
            float8_dynamic_activation_float8_weight(granularity=granularity_map[granularity]),
            filter_fn=proj_filter_conv3d,
        )
    else:
        raise ValueError(f"Unexpected config: {torchao_config}")

    return model
```

评论区精华

该 PR 没有传统 review 评论，评论区仅两条非技术性内容：

- mintlify[bot] 发布了文档预览部署链接（lmsysorg-remove-torchao），供核对 quantization 文档渲染效果。
- 作者 b8zhong 指出 CI 上出现 "Flake on main"（run #31640770400），并附链接，说明该失败来自 main 分支的既有不稳定问题，与本次改动无关。

无遗留技术争议或未解决问题。

- CI flake on main (testing): 该失败与 torchao 移除无关，未引发进一步讨论或代码调整。

风险与影响

- 风险：风险主要来自以下三方面：

1. 公开 CLI 参数移除：--torchao-config 被删除后，仍使用该参数的用户脚本会直接启动失败。不过由于该参数此前在所有取值下都会抛 ImportError，实际可用用户几乎为零，回归影响有限。
2. 跨模块引用残留：删除涉及 server_args.py、loader.py、model_runner.py、paddleocr_vl.py、runners.py 等 5 个以上源码文件，若存在对 get_exec().graph.torchao_config 或 model.torchao_applied 的其他引用，可能在运行时触发 AttributeError。本 PR 依靠代码搜索清理，但未提供自动化验证。
3. 多平台依赖移除：pyproject_npu.toml、pyproject_xpu.toml、AMD wheel、XPU Dockerfile 均移除 torchao，若这些平台仍有隐含的运行时依赖未声明，可能影响构建或启动。

整体风险较低，因被删除功能已完全不可用。

- 影响：影响范围：
- 用户：不能再使用 --torchao-config 参数，需要改用 SGLang 原生量化通道（如 --quantization fp8、nvfp4_online 等）。
- 系统：启动依赖减少一项，python/sglang/check_env.py 的环境检查列表和多个平台的打包配置同步收窄；LayeredModelLoader 和 maybe_apply_post_load_model_transforms 的控制流简化。
- 团队：后续维护者不再需要维护一套与 torchao API 紧耦合的适配代码，量化功能入口更聚焦于原生实现；文档移除也可避免误导用户尝试不可用的配置。

影响程度：中低，属于契约收敛型清理。

- 风险标记：公开 CLI 参数移除，跨模块引用残留，依赖移除影响多平台构建

关联脉络

- 暂无明显关联 PR