PR #26786 完整报告

sgl-project/sglang

[GPTQ] Refactor CPU quantization schemes

合并时间：2026-06-08 18:14

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26786

执行摘要

- 一句话：重构 GPTQ CPU 量化方案，拆分内核与 scheme 并迁移目录
- 推荐动作：该 PR 是量化模块归档重构的典型范例，值得量化子系统开发人员精读。重点关注 _init_kernel 工厂模式如何实现平台解耦，以及如何在保持现有 API 不变的情况下重组目录。后续新增 CPU 量化方案应遵循 hardware_backend/cpu/quantization 放内核、layers/quantization/<algo>/schemes 放 scheme 的约定。

功能与动机

PR 旨在使 GPTQ 量化方案的代码组织与已有的 AWQ 方案对齐，保持平台特定内核在后端初始化而非包导入时加载，消除重复配置，便于维护和扩展。

实现拆解

实现过程分为以下步骤：

1. 创建 CPU AMX 内核模块：在 hardware_backend/cpu/quantization/ 下新建 gptq_kernels.py 和 awq_kernels.py，将先前内联在 scheme 中的 process_weights_after_loading 和 apply 逻辑提取为独立的 Kernel 类（例如 GPTQIntelAMXLinearKernel、GPTQIntelAMXMoEKernel）。
2. 搬迁 GPTQ CPU scheme：将旧顶层文件 layers/quantization/gptq_cpu.py（含 CPUGPTQConfig、GPTQLinearIntelAMXMethod）移动至 layers/quantization/gptq/schemes/gptq_cpu.py，改造为继承 GPTQLinearScheme 的 GPTQIntelAMXLinearScheme，通过 _init_kernel 工厂方法创建对应的 Kernel 实例。
3. 主配置类调整：在 gptq.py 中新增 CPUGPTQConfig 子类，统一处理 CPU 场景的 scheme 分发；同时将原有的 GPTQMoEAscendMethod 和 GPTQLinearAscendMethod 分别泛化为 GPTQMoEMethod 和 GPTQLinearMethod，消除平台紧耦合。
4. AWQ CPU scheme 对称清理：将 awq_cpu.py 中内联的 AWQIntelAMXLinearKernel 和 AWQIntelAMXMoEKernel 定义移至 awq_kernels.py，scheme 文件只保留对内核的引用和 _init_kernel 工厂。
5. 更新导出与导入路径：调整所有相关 __init__.py 的导出，并修复 gptq_marlin.py、awq_marlin.py 等文件的内核初始化方式，统一采用 _init_kernel 延迟加载模式。
6. 测试与配置配套：本次改动不直接包含测试文件，但需确保现有 CI（标签 run-ci）通过；后续发现 MXFP4 配置被误删，已通过 #27782 修复。

关键文件：

- python/sglang/srt/layers/quantization/gptq/schemes/gptq_cpu.py（模块 量化层；类别 source；类型 rename-or-move；符号 CPUGPTQConfig, get_supported_act_dtypes, get_quant_method, _check_cpu_amx_support）：核心搬迁文件，将 GPTQ CPU scheme 从旧顶层模块迁移到 schemes 子目录，并重构为继承 GPTQLinearScheme 的模式，通过 _init_kernel 工厂获取内核实例。
- python/sglang/srt/hardware_backend/cpu/quantization/gptq_kernels.py（模块 硬件后端；类别 source；类型 core-logic；符号 GPTQIntelAMXLinearKernel, init, process_weights_after_loading, apply）：新创建的 GPTQ CPU 内核类，封装权重处理和 INT4 矩阵乘法，实现与 scheme 的解耦。
- python/sglang/srt/hardware_backend/cpu/quantization/awq_kernels.py（模块 硬件后端；类别 source；类型 core-logic；符号 AWQIntelAMXLinearKernel, init, process_weights_after_loading, apply）：对称新增 AWQ CPU 内核类，与 GPTQ 内核结构一致，统一 CPU 量化内核存放位置。
- python/sglang/srt/layers/quantization/gptq/gptq.py（模块 量化层；类别 source；类型 core-logic；符号 CPUGPTQConfig, get_supported_act_dtypes, get_quant_method, get_linear_scheme）：主配置类新增 CPUGPTQConfig 子类，并泛化了 Ascend 类命名，统一 method 选择逻辑。
- python/sglang/srt/layers/quantization/awq/schemes/awq_cpu.py（模块 量化层；类别 source；类型 core-logic；符号 AWQIntelAMXLinearKernel, init, process_weights_after_loading, apply）：原有 AWQ CPU scheme 大幅简化，内核定义移出至 hardware_backend，scheme 只保留工厂方法。

关键符号：CPUGPTQConfig.get_quant_method, GPTQIntelAMXLinearScheme._init_kernel, GPTQIntelAMXLinearScheme.create_weights, GPTQIntelAMXLinearKernel.process_weights_after_loading, GPTQIntelAMXLinearKernel.apply, AWQIntelAMXLinearKernel.process_weights_after_loading, AWQIntelAMXLinearKernel.apply, _check_cpu_amx_support

关键源码片段

python/sglang/srt/layers/quantization/gptq/schemes/gptq_cpu.py

核心搬迁文件，将 GPTQ CPU scheme 从旧顶层模块迁移到 schemes 子目录，并重构为继承 GPTQLinearScheme 的模式，通过 _init_kernel 工厂获取内核实例。

```
# SPDX-License-Identifier: Apache-2.0
from __future__ import annotations

from typing import TYPE_CHECKING

import torch

# 从硬件后端导入内核类，而非内联定义
from sglang.srt.hardware_backend.cpu.quantization.gptq_kernels import (
    GPTQIntelAMXLinearKernel,
```

```
    GPTQIntelAMXMoEKernel,
)
from sglang.srt.layers.linear import set_weight_attrs
from sglang.srt.layers.moe import MoeRunnerConfig
from sglang.srt.layers.parameter import (
    ChannelQuantScaleParameter,
    GroupQuantScaleParameter,
    PackedColumnParameter,
    PackedvLLMParameter,
    RowvLLMParameter,
)

from .gptq_linear import GPTQLinearScheme
from .gptq_scheme import GPTQMoESchemeBase

if TYPE_CHECKING:
    from sglang.srt.layers.moe.token_dispatcher import StandardDispatchOutput
    from sglang.srt.layers.quantization.gptq.gptq import GPTQConfig

__all__ = ["GPTQIntelAMXLinearScheme", "GPTQIntelAMXMoEScheme"]


def _check_cpu_amx_support(quant_config: "GPTQConfig") -> None:
    # 验证当前量化配置在 CPU AMX 上的限制
    if quant_config.desc_act and not (
        quant_config.true_sequential and quant_config.static_groups
    ):
        raise ValueError(
            "Currently, desc_act (True) is only supported with sequential "
            "and static group on CPU with AMX."
        )
    if quant_config.weight_bits != 4:
        raise ValueError("Currently, only 4bits is supported on CPU with AMX.")
    if quant_config.checkpoint_format == "gptq_v2":
        raise ValueError("Currently, gptq_v2 is not supported on CPU with AMX.")


class GPTQIntelAMXLinearScheme(GPTQLinearScheme):
    """Linear scheme for GPTQ on Intel CPU with AMX."""

    def _init_kernel(self, quant_config: "GPTQConfig"):
        # 工厂方法：返回平台特定的内核实例
        return GPTQIntelAMXLinearKernel(quant_config)

    def create_weights(
        self,
        layer: torch.nn.Module,
        input_size_per_partition: int,
        output_partition_sizes: list[int],
```

```
        input_size: int,
        params_dtype: torch.dtype,
        weight_loader,
        **kwargs,
    ):
        # 先进行前置校验，尽早失败
        _check_cpu_amx_support(self.quant_config)

        if input_size_per_partition % self.quant_config.group_size != 0:
            raise ValueError(
                "The input size is not aligned with the quantized "
                "weight shape. This can be caused by too large "
                "tensor parallel size."
            )
        # ... 后续权重参数创建逻辑
```

python/sglang/srt/hardware_backend/cpu/quantization/gptq_kernels.py

新创建的 GPTQ CPU 内核类，封装权重处理和 INT4 矩阵乘法，实现与 scheme 的解耦。

```
# SPDX-License-Identifier: Apache-2.0
from __future__ import annotations

from typing import TYPE_CHECKING, Optional

import torch

from sglang.srt.layers.amx_utils import (
    CPUQuantMethod,
    _amx_process_weight_after_loading,
)
from sglang.srt.layers.moe import MoeRunnerConfig

if TYPE_CHECKING:
    from sglang.srt.layers.moe.token_dispatcher import StandardDispatchOutput
    from sglang.srt.layers.quantization.gptq.gptq import GPTQConfig

__all__ = ["GPTQIntelAMXLinearKernel", "GPTQIntelAMXMoEKernel"]


class GPTQIntelAMXLinearKernel:
    def __init__(self, quant_config: "GPTQConfig"):
        # 保存量化配置，供后续初始化使用
        self.quant_config = quant_config

    def process_weights_after_loading(self, layer: torch.nn.Module) -> None:
        # 对加载后的权重进行 AMX 格式转换（重排、转置等）
        _amx_process_weight_after_loading(
            layer, ["qweight", "qzeros", "scales"], None, "gptq"
        )
```

```
        # 将转换后的权重封装为不可训练参数，节省显存
        layer.qweight = torch.nn.Parameter(layer.qweight.data, requires_grad=False)
        layer.qzeros = torch.nn.Parameter(layer.qzeros.data, requires_grad=False)
        layer.scales = torch.nn.Parameter(layer.scales.data, requires_grad=False)

    def apply(
        self,
        layer: torch.nn.Module,
        x: torch.Tensor,
        bias: Optional[torch.Tensor] = None,
    ) -> torch.Tensor:
        # 调用底层 CPU INT4 量化矩阵乘法算子
        return torch.ops.sgl_kernel.int4_scaled_mm_cpu(
            x,
            layer.qweight,
            layer.qzeros,
            layer.scales,
            bias,
        )
```

评论区精华

代码审查中，gemini-code-assist[bot]提出了关于属性初始化的建议，要求在 AWQIntelAMXMoEKernel 和 GPTQMoEMarlinKernel 的 __init__ 中预先初始化 moe_runner_config、kernel_config 等属性为 None，以防范 AttributeError；同时在 GPTQAscendLinearScheme.create_weights 中建议立即检查 NPU 对 desc_act 的支持以尽早失败。作者随后提交了修改确认。合并后，yanbing-j发现本 PR 误删了 Xeon 平台已有的 MXFP4 配置（来自 #16775），作者承认失误，该配置通过 #27782 恢复。

- 内核类属性初始化与 NPU desc_act 早期检查建议 (correctness): 作者已按要求修正对应文件并提交。
- MXFP4 配置被误删 (correctness): 通过后续 PR #27782 恢复该配置。

风险与影响

- 风险：
  1. 导入路径变更：大量文件涉及 __init__.py 和模块引用更新，存在遗漏或错误导致运行时 ImportError 的风险。
  2. MXFP4 配置误删：已通过随后的 #27782 修复，但仍提醒需要对该类重构进行更全面的配置覆盖检查。
  3. 属性未初始化：尽管反馈已处理，若其他类似代码未遵循同样模式仍可能触发 AttributeError。
  4. 回归风险：CPU AMX 量化路径在 CI 中的测试覆盖可能不足，需注意是否通过全部测试。- 影响：影响范围集中在 CPU Intel AMX 量化路径，包括 GPTQ 和 AWQ 的 Linear 和 MoE 层。重构后代码结构更清晰，方便未来增加新的 CPU 量化方案。对 GPU 和 NPU 后端无直接影响（仅调整了导入时机和类名泛化）。团队需确保所有平台开发者知晓新的

目录约定。 - 风险标记：MXFP4 配置误删，导入路径变更，属性初始化遗漏

关联脉络

- PR #27782 [Hotfix] Add back MXFP4 config for Xeon platform: 修复本 PR 误删的 MXFP4 配置，是直接关联的后续修复。
- PR #16775 [CPU] Support MXFP4 quantization on Intel Xeon: 本 PR 重构时误删了该 PR 引入的 MXFP4 配置，构成依赖关系。