PR #29593 完整报告

sgl-project/sglang

[CPU][QUANT] add amx cpu support for auto-round

合并时间：2026-08-13 15:51

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29593

执行摘要

- 一句话：AutoRound INT4 模型 CPU 推理支持上线（Intel AMX）
- 推荐动作：值得精读，尤其是 auto_round.py 里“配置默认假设显式化 + 一次性日志 + 守卫校验”的组合是量化后端接入的样板：先约束平台 / 位宽 /act-order，再透传完整配置。建议关注三点：get_gptq_config_kwargs 的默认值机制、CPU 分支复用 AWQCPUConfig/CPUGPTQConfig 的接线方式、以及 register_cpu_ci(suite="base-b-test-cpu") 的 CI 分层约定。对要在其他平台（如 NPU/XPU）接入 AutoRound 的开发者有直接借鉴意义。

功能与动机

issue #27568 明确提出“Support auto-round INT4 model offline inference on CPU”，PR body 说明“Currently supports offline inference for CUDA devices with auto-round quantized models, but CPU inference is not yet supported. This is a blocker for users who want to run INT4 quantized models on CPU-only machines or in CPU-based serving pipelines using SGLang as the backend.”即该变更直接服务于无 GPU 环境下的 INT4 模型离线推理与服务化部署诉求。

实现拆解

1. 扩展 AutoRoundConfig 配置契约

在 python/sglang/srt/layers/quantization/auto_round.py 中为 AutoRoundConfig 新增 lm_head_quantized、desc_act、dynamic、checkpoint_format、true_sequential、static_groups 等 GPTQ 风格字段，from_config 通过局部函数 has_any_key 检测 checkpoint 配置中缺失的键并记录到 gptq_defaulted_config_keys，用于后续一次性日志提示默认假设。

2. 新增 CPU/AMX 支持守卫与 GPTQ 配置聚合

新增 check_cpu_support（仅允许 4-bit 且要求 cpu_has_amx_support()）、check_gptq_support（拒绝 desc_act=True）、log_gptq_default_assumptions_once 与 get_gptq_config_kwargs。在 apply_awq_quant_layer 与 apply_gptq_quant_layer 中分别插入 _is_cpu 分支，将权重位宽与后端检查通过后，复用 AWQCPUConfig/CPUGPTQConfig 及其 Method 完成 CPU 加载。

3. 注册 CPU 量化方法并修正 MoE/marlin 分支

在 python/sglang/srt/layers/quantization/__init__.py 的 CPU_QUANTIZATION_METHODS 注册 auto-round，在 python/sglang/srt/configs/model_config.py 的 optimized_quantization_methods 加入 auto-round；同时修正 FusedMoE 层使用 marlin 后端时误走 else: use_marlin = False 的分支判断，并补上 amx_utils.py 中 is_conv_weight 初始化。

4. 测试与文档配套

新增 test/registered/cpu/quant/test_autoround.py：TestAutoRoundCPUConfig 验证 GPTQ 默认值与 desc_act 拒绝逻辑，TestAutoRoundCPU 以 AMX 为条件（skipUnless(cpu_has_amx_support())）启动 server 跑 MMLU（32 例，阈值 0.25），并注册到 base-b-test-cpu CI suite。文档 docs/docs/advanced_features/quantization.mdx 补充 CPU serving 示例命令与"仅限 Intel AMX 4-bit"的边界说明。

关键文件：

- python/sglang/srt/layers/quantization/auto_round.py（模块 量化层；类别 source；类型 core-logic；符号 AutoRoundConfig, has_any_key, check_cpu_support, log_gptq_default_assumptions_once）：核心实现：新增 GPTQ 风格配置透传、CPU/AMX 守卫、CPU 分支复用 AWQCPUConfig/CPUGPTQConfig，并修正 FusedMoE+marlin 分支逻辑
- test/registered/cpu/quant/test_autoround.py（模块 量化测试；类别 test；类型 test-coverage；符号 TestAutoRoundCPUConfig, test_gptq_defaults_are_explicit, test_gptq_desc_act_is_rejected, TestAutoRoundCPU）：新增 CPU AutoRound 精度测试，覆盖 GPTQ 默认值 /desc_act 拒绝的单元断言与端到端 MMLU 推理，并注册 Intel base-b CI suite
- python/sglang/srt/layers/quantization/__init__.py（模块 量化层；类别 source；类型 configuration；符号 CPU_QUANTIZATION_METHODS）：将 auto-round 注册进 CPU_QUANTIZATION_METHODS，是 CPU 引擎能识别该量化方法的入口
- python/sglang/srt/configs/model_config.py（模块 模型配置；类别 source；类型 configuration）：将 auto-round 加入 optimized_quantization_methods，与 CUDA 侧保持一致的优化路径判定
- python/sglang/srt/layers/amx_utils.py（模块 AMX 工具；类别 source；类型 bugfix；符号 _amx_process_weight_after_loading）：修复 is_conv_weight 未初始化导致的潜在 NameError/ 误判，涉及 AMX 权重复包路径
- docs/docs/advanced_features/quantization.mdx（模块 量化文档；类别 docs；类型 documentation）：补充 CPU serving 使用说明并明确仅限 Intel AMX 4-bit 的边界，避免用户误用

关键符号：AutoRoundConfig.from_config, AutoRoundConfig.check_cpu_support, AutoRoundConfig.check_gptq_support, AutoRoundConfig.log_gptq_default_assumptions_once, AutoRoundConfig.get_gptq_config_kwargs, AutoRoundConfig.apply_awq_quant_layer, AutoRoundConfig.apply_gptq_quant_layer

关键源码片段

python/sglang/srt/layers/quantization/auto_round.py

核心实现：新增 GPTQ 风格配置透传、CPU/AMX 守卫、CPU 分支复用 AWQCPUConfig/CPUGPTQConfig，并修正 FusedMoE+marlin 分支逻辑

```
# python/sglang/srt/layers/quantization/auto_round.py
# 模块级一次性探测 CPU / AMX 能力，避免在每个 layer 上重复调用
_is_cpu = is_cpu()
_is_cpu_amx_available = cpu_has_amx_support()

# AutoRound 的 auto_gptq 导出通常不写全 GPTQ 字段，
# 这里集中定义 SGLang 侧采用的默认值，便于在日志中一次性说明假设
_GPTQ_DEFAULTS = {
    "lm_head_quantized": False,
    "desc_act": False,
    "dynamic": {},
    "checkpoint_format": "",
    "true_sequential": False,
    "static_groups": False,
}


class AutoRoundConfig(QuantizationConfig):

    @classmethod
    def from_config(cls, config: dict[str, Any]) -> "AutoRoundConfig":
        # 记录 checkpoint 配置中缺失的 GPTQ 键，
        # 后续用 log_gptq_default_assumptions_once 提示用户“这里用了默认值”
        def has_any_key(keys: list[str]) -> bool:
            return any(key in config for key in keys)

        gptq_config_keys = {
            "lm_head_quantized": ["lm_head", "lm_head_quantized"],
            "desc_act": ["desc_act"],
            "dynamic": ["dynamic"],
            "checkpoint_format": ["checkpoint_format"],
            "true_sequential": ["true_sequential"],
            "static_groups": ["static_groups"],
        }
        gptq_defaulted_config_keys = tuple(
            name for name, keys in gptq_config_keys.items() if not has_any_key(keys)
        )
        # ... 其余字段解析与原来一致，仅新增以下透传项 ...
        return cls(
            # ... 原有 bits/group_size/sym/packing_format/backend 等 ...
            lm_head_quantized=cls.get_from_keys_or(
                config, ["lm_head", "lm_head_quantized"], False
            ),
```

```
            desc_act=cls.get_from_keys_or(config, ["desc_act"], False),
            dynamic=cls.get_from_keys_or(config, ["dynamic"], {}) or {},
            gptq_defaulted_config_keys=gptq_defaulted_config_keys,
        )

    def check_cpu_support(self, weight_bits: int) -> None:
        # CPU 推理路径当前只支持 4-bit，且必须是 Intel AMX；
        # 其他位宽或非 AMX CPU（含 AMD CPU）直接拒绝，避免错误结果
        if weight_bits != 4:
            raise ValueError(
                "SGLang's AutoRound CPU inference path currently supports "
                "only 4-bit AWQ/GPTQ checkpoints because it uses the Intel "
                f"AMX INT4 backend, but got {weight_bits}-bit."
            )
        if not _is_cpu_amx_available:
            raise ValueError(_CPU_AMX_REQUIRED_MSG)

    def get_gptq_config_kwargs(self, weight_bits: int, group_size: int) -> dict[str, Any]:
        # 把“默认假设日志”和“desc_act 守卫”收敛到一处，
        # CPU / NPU 分支统一从这里拿参数，避免各分支硬编码默认值
        self.log_gptq_default_assumptions_once()
        self.check_gptq_support()
        return {
            "weight_bits": weight_bits,
            "group_size": group_size,
            "lm_head_quantized": self.lm_head_quantized,
            "desc_act": self.desc_act,
            "dynamic": self.dynamic,
            "checkpoint_format": self.checkpoint_format,
            "true_sequential": self.true_sequential,
            "static_groups": self.static_groups,
        }

    def apply_gptq_quant_layer(self, layer, prefix: str, backend: str = "auto"):
        # ... 前面先解析 weight_bits/group_size/sym ...
        if _is_cpu:
            # CPU 分支：限 4-bit + AMX，直接复用 CPUGPTQConfig 与对应 Method
            self.check_cpu_support(weight_bits)
            from sglang.srt.layers.quantization.gptq import CPUGPTQConfig

            quant_args = CPUGPTQConfig(
                **self.get_gptq_config_kwargs(weight_bits, group_size),
            )
            quant_args.sym = sym
            # ... 设置 layer.scheme 并返回 GPTQLinearMethod / GPTQMoEMethod ...
```

test/registered/cpu/quant/test_autoround.py

新增 CPU AutoRound 精度测试，覆盖 GPTQ 默认值 /desc_act 拒绝的单元断言与端到端 MMLU 推理，并注册 Intel base-b CI suite

```
# test/registered/cpu/quant/test_autoround.py
# 仅注册到 base-b-test-cpu：该 suite 由 Intel 托管且为 AMX CPU，
# 与 base-a（sglang 托管、x86 无 AMX）区分开
register_cpu_ci(est_time=330, suite="base-b-test-cpu")


class TestAutoRoundCPUConfig(CustomTestCase):
    # 验证缺失字段时 SGLang 采用显式默认值，而不是静默传入 None
    def test_gptq_defaults_are_explicit(self):
        quant_config = AutoRoundConfig.from_config(
            {
                "bits": 4,
                "group_size": 128,
                "sym": True,
                "packing_format": "auto_round:auto_gptq",
            }
        )
        gptq_kwargs = quant_config.get_gptq_config_kwargs(4, 128)
        self.assertFalse(gptq_kwargs["desc_act"])
        self.assertFalse(gptq_kwargs["lm_head_quantized"])
        self.assertEqual(gptq_kwargs["dynamic"], {})

    # AutoRound 的 auto_gptq 导出不支持 act-order，
    # 遇到 desc_act=True 必须报错并引导用户改用 gptq/gptq_marlin
    def test_gptq_desc_act_is_rejected(self):
        quant_config = AutoRoundConfig.from_config(
            {
                "bits": 4,
                "group_size": 128,
                "sym": True,
                "packing_format": "auto_round:auto_gptq",
                "desc_act": True,
            }
        )
        with self.assertRaisesRegex(ValueError, "desc_act=False only"):
            quant_config.get_gptq_config_kwargs(4, 128)


# 非 AMX CPU（如 AMD CPU）直接跳过，避免在错误的硬件上误报失败
@unittest.skipUnless(
    cpu_has_amx_support(),
    "AutoRound INT4 CPU inference requires the Intel AMX CPU backend.",
)
class TestAutoRoundCPU(CustomTestCase):
    @classmethod
    def setUpClass(cls):
```

```
        cls.base_url = DEFAULT_URL_FOR_TEST

    # 端到端：拉起 CPU server，跑 32 例 MMLU，score 阈值 0.25
    def test_mmlu(self):
        device = "cpu"
        for model in DEFAULT_AUTOROUND_MODEL_NAME_FOR_TEST:
            with self.subTest(model=model):
                process = popen_launch_server(
                    model,
                    self.base_url,
                    timeout=DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH,
                    other_args=["--trust-remote-code", "--quantization", "auto-round"],
                    device=device,
                )
                try:
                    args = SimpleNamespace(
                        base_url=self.base_url,
                        model=model,
                        eval_name="mmlu",
                        num_examples=32,
                        num_threads=32,
                        device=device,
                    )
                    metrics = run_eval(args)
                    self.assertGreaterEqual(metrics["score"], 0.25)
                finally:
                    kill_process_tree(process.pid)
```

评论区精华

讨论集中在三处：

- jianan-gu 对 CPU GPTQ 分支中硬编码默认值（lm_head_quantized=False, desc_act=False）提出“Shall we have some logs (or assertion) for this default assumptions?”——这直接催生了 log_gptq_default_assumptions_once 与 get_gptq_config_kwargs 的设计。
- mingfeima 在文档 diff 上指出实现只支持 INT4 + AMX（指向 check_cpu_support 的 R281-R289），要求文档不得暗示通用 x86/AVX512 支持；作者回复“nice catch”并修正文档，明确仅 Intel AMX。
- mingfeima 建议将 CPU CI 注册从 base-a-test-cpu 改为 base-b-test-cpu（base-b 由 Intel 托管且为 AMX CPU），测试文件的最终版本已采纳。 此外 review 状态经历了 mingfeima 的 CHANGES_REQUESTED 到 APPROVED。
- GPTQ 默认假设是否需要日志或断言 (design): 作者新增 log_gptq_default_assumptions_once 与 get_gptq_config_kwargs，集中管理默认值并在首次使用时输出日志；同时 check_gptq_support 对 desc_act=True 抛错并引导用户改用 gptq/gptq_marlin。

- 文档声称通用 x86 支持与实现（仅 AMX）不一致 (documentation): 作者承认“nice catch”，将文档改为仅支持 Intel AMX CPU 的 4-bit AutoRound checkpoint，与非 AMX CPU 路径划清界限。
- CPU CI suite 选择（base-a vs base-b） (infra): 采纳建议，最终 register_cpu_ci(est_time=330, suite="base-b-test-cpu")。

风险与影响

- 风险:
  1. 平台限定风险：CPU 路径硬性要求 Intel AMX（check_cpu_support 中 _is_cpu_amx_available 不满足即抛错），文档已对齐，但用户若误用非 AMX CPU 会得到启动期 ValueError，属预期内但体验较硬。
  2. 行为默认值风险：get_gptq_config_kwargs 会把缺失的 desc_act/lm_head_quantized/dynamic 等按 _GPTQ_DEFAULTS 静默补全，虽然加了“仅记录一次”的日志，但如果 checkpoint 实际是 act-order 的 GPTQModel 导出，check_gptq_support 能拦截 desc_act=True，而 lm_head_quantized 之类的默认 False 并不会校验，仍需依赖测试覆盖。
  3. CPU/MoE 组合风险：文档与代码都限定 MoE 专家仍需 AMX 路径；amx_utils.py 的 is_conv_weight 初始化修复涉及权重复包路径，若 CPU 上加载含卷积权重的模型（如 Mamba/ 混合架构），行为可能受影响，改动仅 1 行但需留意回归。
  4. 测试覆盖面：新增测试仅在 AMX CPU 上运行（非 AMX 直接 skip），且注册在 Intel 托管的 CI suite；SGLang 侧 CI 无法兜底验证，回归检测依赖 Intel 侧 CI 稳定性。
  5. 分支耦合：auto_round.py 中 _is_cpu 分支与现有 CUDA/NPU 分支并存，后续若 AWQ/GPTQ CPU 后端接口变化，需要同步维护。 - 影响：影响范围中等且明确：为 CPU-only 部署（Intel AMX）解锁 AutoRound INT4 推理与 serving 能力，覆盖 auto_round:auto_gptq 与 auto_round:auto_awq 两种 packing 格式；对 CUDA/NPU 现有路径不改变默认行为，但 FusedMoE marlin 分支的条件重构（if isinstance(layer, FusedMoE) 提前返回）会影响所有使用 marlin 后端的 MoE 量化模型，需要回归验证。团队层面，Intel CPU CI（base-b-test-cpu）新增约 330 秒精测任务；文档明确了“仅 Intel AMX + 4-bit”的边界，避免用户误用。 - 风险标记：新增平台分支逻辑，硬性 AMX 依赖，默认值静默补全，CI 依赖 Intel 托管，marlin 分支重构

关联脉络

- PR #27568 [Feature] Support auto-round INT4 model offline inference on CPU: 本 PR 直接解决的关联 issue，前端 PR #22685 与原始 issue #20914 亦为同链条需求
- PR #22685 frontend PR reference for auto-round CPU support: PR body 中引用的 AutoRound CPU 支持前端 PR，属于同一功能线的上游依赖