PR #35455 完整报告

sgl-project/sglang

[Quant] Load compressed-tensors kv_cache_scheme scales

合并时间：2026-08-20 19:18

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35455

执行摘要

- 一句话：修复 compressed-tensors KV cache scale 加载缺失问题
- 推荐动作：值得精读。该 PR 修复了一个静默的正确性问题，展示了如何在不破坏兼容性的前提下补全量化方案分支：支持时加载 scale，不支持时降级并告警，而非拒绝启动；通过 duck-typed 属性让 auto 模式正确解析 dtype。对理解 SGLang 量化加载架构（QuantizationConfig → QuantizeMethodBase 分派）很有帮助。

功能与动机

PR body 指出，compressed-tensors 检查点声明的 kv_cache_scheme 携带校准的 per-tensor k_scale/v_scale，但加载时被静默丢弃（报 Parameter model.layers.N.k_scale not found in params_dict），导致 fp8 KV cache 未缩放。三个缺口叠加：from_config 未转发 scheme、get_quant_method 无 RadixAttention 分支、qwen3_5 mapper 只认 ModelOpt 风格名称。这是一个正确性修复，而非精度提升。

实现拆解

1. 转发 kv_cache_scheme：在 CompressedTensorsConfig.from_config 中从 config 字典读取 kv_cache_scheme 并传入构造函数，使后续逻辑能感知声明的方案。
2. 新增 KV cache 量化方法：新增 CompressedTensorsKVCacheMethod(BaseKVCacheMethod)，在 get_quant_method 中增加 RadixAttention 分支：声明且被 is_supported_scheme 支持时返回该方法，从而创建 k_scale/v_scale 参数并加载；不支持时降级为 None 并告警，避免启动失败。
3. 解析 auto dtype：新增 kv_cache_quant_algo 属性，configure_kv_cache_dtype 通过 duck typing 读取它，使 --kv-cache-dtype auto 能根据声明的支持方案解析为 FP8 池，避免落入 bf16 池导致 scale 与池不匹配。
4. 名称映射：在 QWEN3_5_KV_SCALE_MAPPER 中新增 .self_attn.k_scale / .self_attn.v_scale 到 .attn.k_scale / .attn.v_scale 的映射，并在 Qwen3.5 MTP loader 的 load_weights 入口处应用该 mapper，保证 MTP 模块也能正确加载 scale。
5. 清理与测试：移除 apply_weight_name_mapper 中对 kv_cache_scheme 的 apply_dict 调用（scheme 字段不含模块名，且可能误删键），新增 CPU 单元测试覆盖声明 / 未声明 / 不支持方案和 auto dtype 解析四种场景。

关键文件：

- python/sglang/srt/layers/quantization/compressed_tensors/compressed_tensors.py（模块 量化加载；类别 source；类型 core-logic；符号 kv_cache_quant_algo, CompressedTensorsKVCacheMethod, is_supported_scheme, from_config）：核心逻辑文件：新增 kv_cache_quant_algo 属性、CompressedTensorsKVCacheMethod 类、RadixAttention 分支，并转发 kv_cache_scheme，是修复的关键。
- test/registered/unit/layers/quantization/test_compressed_tensors_kv_cache.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestCompressedTensorsKVCacheMethod, test_declared_scheme_gets_kv_cache_method, test_no_scheme_returns_none, test_kv_cache_quant_algo_resolves_auto_dtype）：新增单元测试，覆盖声明方案、无方案、不支持方案、auto dtype 解析，为修复提供回归保护。
- python/sglang/srt/models/qwen3_5.py（模块 模型映射；类别 source；类型 data-contract；符号 QWEN3_5_KV_SCALE_MAPPER）：在 QWEN3_5_KV_SCALE_MAPPER 中新增 compressed-tensors 风格的 attention 级 scale 名称映射，使这些参数的名称能正确对应到 RadixAttention 上的参数。
- python/sglang/srt/models/qwen3_5_mtp.py（模块 MTP 加载；类别 source；类型 data-contract；符号 load_weights）：在 MTP 加载权重时应用同一 KV scale 映射，保证 MTP 模型的 scale 也能正确加载。

关键符号：CompressedTensorsConfig.from_config, CompressedTensorsConfig.get_quant_method, CompressedTensorsConfig.kv_cache_quant_algo, CompressedTensorsKVCacheMethod.is_supported_scheme, Qwen3_5MTP.load_weights

关键源码片段

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

核心逻辑文件：新增 kv_cache_quant_algo 属性、CompressedTensorsKVCacheMethod 类、RadixAttention 分支，并转发 kv_cache_scheme，是修复的关键。

```
# 属性：供 configure_kv_cache_dtype duck typing 读取。
# 声明受支持的 kv_cache_scheme 时必须返回 FP8，否则 auto 模式
# 会解析为 bf16 池，导致已加载的 k_scale / v_scale 与池类型不匹配。
@property
def kv_cache_quant_algo(self) -> Optional[str]:
    if (
        self.kv_cache_scheme is not None
        and CompressedTensorsKVCacheMethod.is_supported_scheme(self.kv_cache_scheme)
    ):
        return "FP8"
    return None


class CompressedTensorsKVCacheMethod(BaseKVCacheMethod):
    """从声明了 kv_cache_scheme 的 compressed-tensors 检查点加载校准的
    k_scale / v_scale。仅支持静态、对称、逐张量（per-tensor）的 FP8 方案。"""
```

```
    def __init__(self, quant_config: CompressedTensorsConfig):
        assert self.is_supported_scheme(quant_config.kv_cache_scheme)
        super().__init__(quant_config)

    @staticmethod
    def is_supported_scheme(scheme: Optional[Dict[str, Any]]) -> bool:
        # 只接受 FP8 逐张量静态对称方案；其余方案交由降级路径处理。
        return (
            scheme is not None
            and scheme.get("type") == "float"
            and scheme.get("num_bits") == 8
            and scheme.get("strategy") == "tensor"
            and scheme.get("symmetric") is True
            and scheme.get("dynamic") is False
        )


def get_quant_method(self, layer, prefix):
    # ... 既有 Linear / MoE 分支 ...
    from sglang.srt.layers.radix_attention import RadixAttention

    if isinstance(layer, RadixAttention):
        if self.kv_cache_scheme is None:
            # 未声明方案保持原行为：不创建 scale 参数。
            return None
        if not CompressedTensorsKVCacheMethod.is_supported_scheme(self.kv_cache_scheme):
            # 不支持时降级，避免阻止启动：未缩放的 KV cache 仍可服务。
            logger.warning_once(
                f"Ignoring compressed-tensors kv_cache_scheme "
                f"{self.kv_cache_scheme}: only static symmetric "
                f"per-tensor FP8 scales are supported."
            )
            return None
        return CompressedTensorsKVCacheMethod(self)
```

评论区精华

在 review 讨论中，BBuf 提出两个问题：一是 auto 模式与显式 fp8 的 GSM8K 结果不一致（0.940 vs 0.965），既然两者解析到同一池，理论应一致，Jiminator 重跑后得到 96.0%，确认为 200 题方差噪声；二是 apply_weight_name_mapper 中对 kv_cache_scheme 的 apply_dict 看似 no-op，scheme 只有 type/num_bits/strategy 等字段，并无模块名，Jiminator 承认这是错误并已移除该 remap，留下注释说明原因。

- auto 模式与显式 fp8 的 GSM8K 分数差异 (question): 确认为噪声，两次运行在统计误差内。
- apply_weight_name_mapper 中的 kv_cache_scheme 映射是否为 no-op (design): 移除 kv_cache_scheme 的 apply_dict 调用，并留下注释说明原因。

风险与影响

- 风险：核心加载路径变更：get_quant_method 和 from_config 的修改影响所有 compressed-tensors 检查点的加载，需回归验证无 kv_cache_scheme 的检查点行为不变。不支持方案的降级策略可能掩盖配置错误，但通过告警日志暴露，且这些方案本就无法使用 scale。auto 模式下 KV 池 dtype 可能从 bf16 变为 fp8，改变显存占用与精度，这是预期正确行为但需关注。Qwen3.5 MTP 加载引入新的 mapper 应用，理论上只匹配特定子串，风险低。新增测试为 CPU 单元测试，未覆盖真实 checkpoint 端到端加载，但 PR body 提供了 RTX PRO 6000 实测数据。
- 影响：对用户：运行带 kv_cache_scheme 的 compressed-tensors 检查点（如 Qwen3.5 混合精度）时，fp8 KV cache 将从未缩放变为正确缩放，修复静默精度损失；显式 bf16 覆盖仍保持旧行为。对系统：auto 模式下 KV pool 可能从 bf16 变为 fp8，影响显存占用与吞吐。对团队：补齐了 compressed-tensors 的 KV cache scale 支持，与 ModelOpt 对齐，后续可统一维护。
- 风险标记：核心量化加载路径变更，auto 模式解析行为变化，不支持方案降级可能掩盖配置错误，Qwen3.5 MTP 加载行为变更

关联脉络

- 暂无明显关联 PR