

PR #49389 完整报告

vllm-project/vllm

[Misc] Remove deprecated calculate_kv_scales runtime KV scale calculation

合并时间: 2026-08-03 23:15

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/49389>

执行摘要

- 一句话: 移除已废弃的 `calculate_kv_scales` 运行时 KV scale 计算路径
- 推荐动作: 该 PR 是一次值得阅读的废弃功能清理范例: 覆盖了配置、算子、runner、测试、文档的完整移除链路, 展示了如何安全地删除一个已废弃的运行时路径。建议关注 `vllm/model_executor/layers/attention/attention.py` 和 `vllm/model_executor/layers/quantization/kv_cache.py` 中 `scale` 解析逻辑的简化, 以及 `gpu_model_runner.py` 中 CUDA graph 约束的移除。mgoin 提到的后续全局扫描值得留意, 若在 main 上发现残留引用应尽快补齐。

功能与动机

PRbody明确说明目的是移除已废弃的`calculate_kv_scales`选项(运行时fp8k/vscale估算), 并指出 fp8 KV cache scales 现在通过单一路径解析: 从 checkpoint 加载, 缺失则默认 1.0。该选项此前已在 `vllm/config/cache.py` 中标注为 deprecated 并计划在 v0.19 移除, 关联变更 #37201 已奠定了 checkpoint 加载优先的基础。

实现拆解

该 PR 按以下步骤完成废弃功能的移除:

1. 配置层移除: 在 `vllm/config/cache.py` 中删除 `CacheConfig.calculate_kv_scales` 字段及其 `_warn_deprecated_calculate_kv_scales` validator; 在 `vllm/engine/arg_utils.py` 中删除 `calculate_kv_scales` 的 `EngineArgs` 字段、`--calculate-kv-scales` CLI 参数以及 `create_engine_config` 中的传递逻辑; 在 `vllm/envs.py` 中删除 `Q_SCALE_CONSTANT`、`K_SCALE_CONSTANT`、`V_SCALE_CONSTANT` 三个环境变量。
2. Attention 层核心移除: 在 `vllm/model_executor/layers/attention/attention.py` 中删除 `self.calculate_kv_scales` 状态、`Attention.forward` 中的 `maybe_calc_kv_scales` 调用、`calc_kv_scales` 方法、`set_default_quant_scales` 中的 `q_range/k_range/v_range` 初始化, 以及自定义算子 `maybe_calc_kv_scales / maybe_calc_kv_scales_fake` 的注册。
3. MLA 层同步移除: 在 `vllm/model_executor/layers/attention/mla_attention.py` 中删除 `calculate_kv_scales` 分支、`MLAAttention.forward` 中的调用、`calc_kv_scales` 方法及 `q_range/k_range/v_range` 初始化, 保持与普通 Attention 行为一致。
4. 量化与 runner 配套: 在 `vllm/model_executor/layers/quantization/kv_cache.py` 的 `process_weights_after_loading` 中, 将 `is_quantized_kv_cache` 分支从依赖 `not`

`layer.calculate_kv_scales` 简化为直接执行，并移除加载后置 `calculate_kv_scales = False` 的语句；在 `vllm/model_executor/models/config.py` 中移除 `HybridAttentionMambaModelConfig` 对 `calculate_kv_scales` 的禁用逻辑；在 `vllm/v1/worker/gpu_model_runner.py` 中删除 `runner` 状态和 `execute_model` 中强制 `CUDAGraphMode.NONE` 的降级分支。

5. 测试、文档与 CI 同步：删除 `tests/compile/fullgraph/test_full_graph.py` 中 `test_fp8_kv_scale_compile`，裁剪 `tests/quantization/test_fp8.py` 中 `test_kv_cache_scale_sync_to_host_copies` 的 `runtime_calc` 参数化分支，清理 `tests/models/quantization/test_per_token_kv_cache.py` 与 `tests/quantization/test_per_token_kv_cache.py` 中的相关引用；更新 `docs/features/quantization/quantized_kvcache.md`、`docs/design/metrics.md`，并调整 `.buildkite/test_areas/compile.yaml`、`.buildkite/test_areas/pytorch.yaml`、`.buildkite/test-amd.yaml` 的测试分组。

关键文件：

- `vllm/model_executor/layers/attention/attention.py`（模块 注意力层；类别 `source`；类型 `data-contract`；符号 `calc_kv_scales`, `maybe_calc_kv_scales`, `maybe_calc_kv_scales_fake`）：核心变更文件：删除 `calculate_kv_scales` 状态、`calc_kv_scales` 方法、`maybe_calc_kv_scales` 自定义算子及其 `fake` 实现，并移除 `set_default_quant_scales` 中的 `q/k/v range` 常量初始化，是运行时 `scale` 计算路径的源头。
- `vllm/model_executor/layers/quantization/kv_cache.py`（模块 量化层；类别 `source`；类型 `data-contract`）：量化方法的关键简化点：`process_weights_after_loading` 不再检查 `layer.calculate_kv_scales`，FP8 KV cache 的 `scale` 解析统一为 `checkpoint` 优先、默认 1.0 的单一路径。
- `vllm/config/cache.py`（模块 配置层；类别 `source`；类型 `core-logic`；符号 `_warn_deprecated_calculate_kv_scales`）：配置源头：删除 `CacheConfig.calculate_kv_scales` 字段及其 `deprecated` 警告 `validator`，是整个移除工作的配置契约起点。

关键符号：`calc_kv_scales`, `maybe_calc_kv_scales`, `maybe_calc_kv_scales_fake`, `set_default_quant_scales`, `process_weights_after_loading`, `verify_and_update_config`, `_warn_deprecated_calculate_kv_scales`

关键源码片段

`vllm/model_executor/layers/attention/attention.py`

核心变更文件：删除 `calculate_kv_scales` 状态、`calc_kv_scales` 方法、`maybe_calc_kv_scales` 自定义算子及其 `fake` 实现，并移除 `set_default_quant_scales` 中的 `q/k/v range` 常量初始化，是运行时 `scale` 计算路径的源头。

```
# vllm/model_executor/layers/attention/attention.py (变更后节选)
```

```
def set_default_quant_scales(layer: nn.Module, register_buffer: bool = False) -> None:
    """为注意力层设置默认量化 scale。"""
    if register_buffer:
```

```

        layer.register_buffer("_k_scale", torch.tensor(1.0, dtype=torch.float32))
        layer.register_buffer("_v_scale", torch.tensor(1.0, dtype=torch.float32))
        layer.register_buffer("_q_scale", torch.tensor(1.0, dtype=torch.float32))
        layer.register_buffer("_prob_scale", torch.tensor(1.0, dtype=torch.float32))
    else:
        layer._k_scale.fill_(1.0)
        layer._v_scale.fill_(1.0)
        layer._q_scale.fill_(1.0)
        layer._prob_scale.fill_(1.0)

```

同时保留 host 内存上的 scale 副本，供 Flashinfer、AITER 等
需要 host scale 的 attention 后端使用。

```

layer._q_scale_float = 1.0
layer._k_scale_float = 1.0
layer._v_scale_float = 1.0
layer._k_scale_cpu = torch.tensor(1.0, dtype=torch.float32)
layer._v_scale_cpu = torch.tensor(1.0, dtype=torch.float32)
layer._prob_scale_float = 1.0
# 注意：此前的 q_range / k_range / v_range 初始化已删除，
# 因为运行时 scale 估算路径（calc_kv_scales）已整体移除。

```

```

def _init_kv_cache_quant(
    layer: nn.Module,
    quant_config: QuantizationConfig | None,
    prefix: str,
) -> None:
    """初始化 KV cache 量化属性（与 MLA 层共享）。"""
    # 注：此前的 calculate_kv_scales 读取与传递逻辑已删除，
    # 不再存在“运行时计算”与“checkpoint 加载”双路径的配置分支。
    ...

```

```

class Attention(nn.Module):
    def __init__(self, ...):
        ...
        # kv_cache_dtype 的解析不再受 calculate_kv_scales 影响，
        # 只保留 cache_config 显式指定和 checkpoint 声明（kv_cache_scheme）的路径。
        kv_cache_scheme = getattr(quant_config, "kv_cache_scheme", None)
        if kv_cache_scheme is not None and kv_cache_dtype == "auto":
            kv_cache_dtype = "fp8"
            if cache_config is not None:
                cache_config.cache_dtype = "fp8"
        ...
        self.kv_cache_dtype = kv_cache_dtype
        # 注意：self.calculate_kv_scales 属性已删除，forward 中也不再
        # 调用 torch.ops.vllm.maybe_calc_kv_scales。

    def forward(self, query, key, value, ...):

```

```

"""前向传播: scale 计算分支已移除, 直接进入 query 量化与 attention 计算."""
if output_dtype is None:
    output_dtype = query.dtype
if self.query_quant is not None:
    ...
# 原 if self.calculate_kv_scales: 调用 maybe_calc_kv_scales 的代码块
# 已整体删除, KV cache scale 完全由 checkpoint 加载决定。
...

```

vllm/model_executor/layers/quantization/kv_cache.py

量化方法的关键简化点: `process_weights_after_loading` 不再检查 `layer.calculate_kv_scales`, FP8 KV cache 的 scale 解析统一为 checkpoint 优先、默认 1.0 的单一路径。

vllm/model_executor/layers/quantization/kv_cache.py (变更后节选)

```

def process_weights_after_loading(self, layer: torch.nn.Module) -> None:
    # skip if there are no weights to process (for example, weight reloading)
    if not hasattr(layer, "q_scale"):
        assert not hasattr(layer, "k_scale")
        assert not hasattr(layer, "v_scale")
        assert not hasattr(layer, "prob_scale")
        return

    # Per-token-head 量化的 KV cache: scale 在 kernel 写缓存时按 (token, head)
    # 动态计算, checkpoint 中的 scale 不使用。
    if kv_cache_uses_per_token_head_scales(layer.kv_cache_dtype):
        layer._k_scale.copy_(1.0)
        layer._v_scale.copy_(1.0)
        layer._k_scale_float = 1.0
        layer._v_scale_float = 1.0
        del layer.k_scale
        del layer.v_scale
        del layer.q_scale
        del layer.prob_scale
        return

    # 量化 KV cache: 不再有 calculate_kv_scales 的运行时计算分支,
    # 统一从 checkpoint 加载 scale; 缺失时使用默认 1.0。
    if is_quantized_kv_cache(layer.kv_cache_dtype):
        if layer.k_scale > 0.0 and layer.v_scale > 0.0:
            # 优先使用独立的 k_scale 与 v_scale
            k_scale = layer.k_scale.to("cpu").tolist()
            v_scale = layer.v_scale.to("cpu").tolist()
            if current_platform.is_fp8_fnuz():
                k_scale *= 2
                v_scale *= 2
        elif layer.k_scale < 0.0 and layer.v_scale < 0.0:
            # 两者均未加载 (无效负值), 回退默认 1.0

```

```

        k_scale = 1.0
        v_scale = 1.0
    else:
        # 只有一个 kv_scale, 则复制给 k_scale 与 v_scale
        assert layer.k_scale > 0.0
        scale_to_duplicate = max(layer.k_scale, layer.v_scale)
        k_scale = scale_to_duplicate.to("cpu").tolist()
        v_scale = scale_to_duplicate.to("cpu").tolist()
        if current_platform.is_fp8_fnuz():
            k_scale *= 2
            v_scale *= 2

    if not isinstance(k_scale, float) or not isinstance(v_scale, float):
        raise ValueError("Only support per-tensor scaling factor for fp8 KV cache")

    # 同步 device 与 host 上的 scale 副本 (AITER 等后端需要 host 副本)
    layer._k_scale.copy_(k_scale)
    layer._v_scale.copy_(v_scale)
    layer._k_scale_float = k_scale
    layer._v_scale_float = v_scale
    layer._k_scale_cpu.fill_(k_scale)
    layer._v_scale_cpu.fill_(v_scale)
    ...

    # q_scale 与 prob_scale 的处理保持原样, 但不再有
    # layer.calculate_kv_scales = False 这类状态翻转语句。
    ...

```

vllm/config/cache.py

配置源头: 删除 `CacheConfig.calculate_kv_scales` 字段及其 deprecated 警告 validator, 是整个移除工作的配置契约起点。

vllm/config/cache.py (变更后节选)

```

class CacheConfig(BaseModel):
    ...
    # 原 calculate_kv_scales: bool = False 字段已删除。
    # 该字段曾用于开启 FP8 KV cache 的运行时动态 scale 估算,
    # 现已废弃, KV cache scale 只从 checkpoint 加载或默认 1.0。

    kv_cache_dtype_skip_layers: list[str] = field(default_factory=list)
    """Layer patterns to skip KV cache quantization. 接受层索引 (如 '0')
    或 attention 类型名 (如 'sliding_window')。"""
    ...

    # 原 _warn_deprecated_calculate_kv_scales validator 已删除:
    # 该 validator 会在用户设置 --calculate-kv-scales 时打印废弃警告,
    # 由于配置项本身已移除, 警告逻辑也不再需要。

```

```

@field_validator("cache_dtype", mode="after")
@classmethod
def _validate_cache_dtype(cls, cache_dtype: CacheDType) -> CacheDType:
    # cache_dtype 校验逻辑保持不变，仅移除对旧选项的分支依赖
    ...

```

评论区精华

该 PR 的 review 讨论集中在自动化和维护确认层面。[claude\[bot\]](#) 指出这是 fork 提交，自动 review 被禁用，需要维护者手动触发。[mgoin](#) 批准并评论: "Awesome, thanks for picking this up! I'll do a sweep later to make sure it is all gone, but this looks reasonable", 表示认可清理的完整性，并计划后续再做一次全局扫描确认无残留。[yewentao256](#) 同样批准: "LGTM, thanks for the work!". Issue 评论中 [mergify](#) 曾提示 merge 冲突与 pre-commit 失败，后续通过 rebase 和修复（第二个 commit 修正 [run_model](#) 类型注解）解决。

- 自动 review 被禁用 (other): 等待维护者 [mgoin](#) 和 [yewentao256](#) 的批准，最终均 APPROVED。
- 清理完整性确认 (design): PR 已合并，维护者计划在 main 上二次复核；当前源码文件中未见残留引用。
- merge 冲突与 pre-commit 失败 (other): 作者通过 rebase 和修复类型注解（第二个 commit）解决了问题。

风险与影响

- 风险：移除路径整体风险可控，但存在以下要点：
 1. 行为变化风险：calculate_kv_scales=True 的用户（如依赖运行时动态 scale 的 FP8 KV cache 场景）在升级后将静默改用 checkpoint scale 或默认 1.0，可能导致精度变化。PR 未提供迁移警告或兼容期，文档 [quantized_kvcache.md](#) 已同步说明。
 2. CUDA graph 行为变化：gpu_model_runner.py 中删除强制 CUDAGraphMode.NONE 的分支意味着此前依赖此降级的路径（运行时 calc 首轮 forward）不会再出现；但该路径本身已删除，逻辑一致。需确认无其他代码引用 calculate_kv_scales 属性（[mgoin](#) 的 sweep 意图也在此）。
 3. 混合模型（Hybrid Attention/Mamba）：[vllm/model_executor/models/config.py](#) 中针对 GDN/Mamba/SSM 的 calculate_kv_scales 禁用逻辑被删除，但该配置项本身已不存在，因此无实际影响；不过需确认这些模型在 checkpoint 无 scale 时默认 1.0 的行为符合预期。
 4. 测试覆盖缩减：test_fp8_kv_scale_compile 被整体删除，减少了 fp8 + compile 组合的回归覆盖；test_kv_cache_scale_sync_to_host_copies 仍保留 checkpoint 路径的设备到主机同步测试，但运行时计算路径的覆盖随之消失。- 影响：影响范围集中在 FP8 KV cache 量化的配置与运行时路径。对用户而言，--calculate-kv-scales CLI 参数和 Q/K/V_SCALE_CONSTANT 环境变量被移除，依赖它们的脚本需要调整；对系统而言，删除了一个与 CUDA graph 不兼容的动态 scale 计算路径，简化了 KV cache 初始化的状态管理，后续 FP8 KV cache 的行为更可预测（checkpoint 优先、默认 1.0）。对团队而言，减少了配置面和运行时分支，降低了维护成本，但需要关注依赖该选项的模型或私有部署的迁移。- 风险标记：配置项移除影响用户脚本，FP8 KV cache 精度行为变化，测试覆盖

缩减，环境变量删除，需要全局扫描残留引用

关联脉络

- PR #37201（关联变更，未在当前历史列表中）：PR body 明确提及 related change #37201，该 PR 奠定了 checkpoint 优先加载 KV cache scale 的基础，本次是移除运行时估算路径的后续清理。