PR #23850 完整报告

sgl-project/sglang

Support RunAI loading for quantized checkpoints

合并时间：2026-05-02 11:11

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23850

执行摘要

- 一句话：支持 RunAI 流式加载量化检查点
- 推荐动作：值得精读。该 PR 涉及模型加载架构的重要扩展，特别是流式生成器模式、缓冲区管理策略（_clone_if_runai_streamed_tensor）以及路由设计中的防御性编程，对理解 SGLang 的加载器体系有参考价值。

功能与动机

支持 RunAI 流式加载量化检查点（ModelOpt），同时保持 Kimi 权重加载流式化以避免缓冲区复用问题，并确保 DeepSeek 模型中来自流式缓冲区的张量在缓存前被克隆。

实现拆解

1. 传递量化配置：在 RunaiModelStreamerLoader.load_model 中调用 _get_quantization_config 获取量化配置，并作为 _initialize_model 的参数传入。
2. 路由优化：在 get_model_loader 中引入 model_optloader_allowed 变量，当 load_format=RUNAI_STREAMER 时禁止跳入 ModelOptModelLoader 分支，确保所有 RunAI 加载经过 RunaiModelStreamerLoader，同时避免调用 _is_already_quantized 可能引发的 HFValidationError。
3. Kimi K25 流式加载：重写 load_weights，定义内部生成器 stream_language_weights，即时加载视觉权重并对语言权重使用 yield 惰性产出，最终传递给 self.language_model.load_weights(stream_language_weights())。
4. DeepSeek 缓冲区保护：在 deepseek_weight_loader.py 中添加 _clone_if_runai_streamed_tensor，检查张量是否标记为流式缓冲区（通过 RUNAI_STREAMER_TENSOR_ATTR），若是则克隆；在 do_load_weights 中对缓存的 q_a_proj / kv_a_proj_with_mqa 张量应用此保护。
5. 张量标记：在 runai_safetensors_weights_iterator 中为每个产出的张量设置 _sglang_runai_streamer_tensor 属性，便于下游识别。
6. 测试配套：新增 test_runai_model_streamer_loader.py，包含 4 个测试用例，覆盖量化配置传递、流式张量标记、DeepSeek 克隆行为以及预量化路由正确性。

关键文件：

- test/registered/unit/model_loader/test_runai_model_streamer_loader.py（模块 测试；类别 test；类型 test-coverage；符号 _FakeModel, TestRunaiModelStreamerLoader,

test_passes_quant_config_to_model_init, test_marks_streamer_tensors）：新增完整测试套件，覆盖量化配置传递、流式张量标记、DeepSeek 克隆行为及预量化路由，确保功能正确性。

- python/sglang/srt/models/kimi_k25.py（模块 模型定义；类别 source；类型 data-contract；符号 load_weights, stream_language_weights）：核心变更文件之一，将 load_weights 改为流式生成器模式，是支持 RunAI 加载的关键改动。
- python/sglang/srt/models/deepseek_common/deepseek_weight_loader.py（模块 权重加载器；类别 source；类型 data-contract；符号 _clone_if_runai_streamed_tensor）：新增 _clone_if_runai_streamed_tensor，保护 DeepSeek 模型中来自流式缓冲区的缓存张量，防止数据被后续覆盖。
- python/sglang/srt/model_loader/loader.py（模块 模型加载器；类别 source；类型 core-logic；符号 RunaiModelStreamerLoader.load_model, get_model_loader）：路由逻辑调整和量化配置传递，确保 RunAI 加载绕过 ModelOptModelLoader。
- python/sglang/srt/model_loader/weight_utils.py（模块 权重工具；类别 source；类型 data-contract；符号 RUNAI_STREAMER_TENSOR_ATTR, runai_safetensors_weights_iterator）：添加 RUNAI_STREAMER_TENSOR_ATTR 常量，并在流式迭代器中为每个张量设置该属性，以便下游识别。

关键符号：load_weights, stream_language_weights, _clone_if_runai_streamed_tensor, runai_safetensors_weights_iterator, RunaiModelStreamerLoader.load_model, get_model_loader

关键源码片段

python/sglang/srt/models/kimi_k25.py

核心变更文件之一，将 load_weights 改为流式生成器模式，是支持 RunAI 加载的关键改动。

```
def load_weights(self, weights: Iterable[Tuple[str, torch.Tensor]]):
    """流式加载权重，即时加载视觉权重并惰性产出语言权重。

    RunAI 的迭代器会复用后端缓冲区，因此必须避免将张量收集到列表后再使用。
    """
    mapper = getattr(self, "hf_to_sglang_mapper", None)
    if mapper is not None:
        weights = mapper.apply(weights)

    vision_params = (
        None
        if self.config.language_only
        else dict(self.named_parameters(remove_duplicate=False))
    )

    def stream_language_weights():
        for name, loaded_weight in weights:
            if "vision_tower" in name or "mm_projector" in name:
                if vision_params is None:
```

```
                continue
            vname = (
                name.replace(r"wqkv.", r"attn.qkv_proj.")
                .replace(r"wo.", r"attn.proj.")
                .replace("mm_projector.proj.0", "mm_projector.linear_1")
                .replace("mm_projector.proj.2", "mm_projector.linear_2")
            )
            if vname not in vision_params:
                raise ValueError(f"Weight {vname} not found in params_dict")
            param = vision_params[vname]
            weight_loader = getattr(param, "weight_loader", default_weight_loader)
            weight_loader(param, loaded_weight) # 立即加载视觉权重
            continue
        yield name.replace("language_model.", ""), loaded_weight

if self.language_model is not None:
    self.language_model.load_weights(stream_language_weights())
else:
    # encoder-only: 消耗生成器以触发视觉权重加载
    for _ in stream_language_weights():
        pass
```

python/sglang/srt/models/deepseek_common/deepseek_weight_loader.py

新增 _clone_if_runai_streamed_tensor，保护 DeepSeek 模型中来自流式缓冲区的缓存张量，防止数据被后续覆盖。

```
def _clone_if_runai_streamed_tensor(tensor: torch.Tensor) -> torch.Tensor:
    """如果张量来自 RunAI 流式缓冲区则克隆，避免数据被覆盖。"""
    if getattr(tensor, RUNAI_STREAMER_TENSOR_ATTR, False):
        return tensor.clone().detach()
    return tensor

# 在 do_load_weights 中使用：
if fuse_qkv_a_proj and ("q_a_proj" in name or "kv_a_proj_with_mqa" in name):
    cached_a_proj[name] = _clone_if_runai_streamed_tensor(loaded_weight)
    # ... 后续融合逻辑 ...
```

python/sglang/srt/model_loader/loader.py

路由逻辑调整和量化配置传递，确保 RunAI 加载绕过 ModelOptModelLoader。

```
class RunaiModelStreamerLoader(DefaultModelLoader):
    def load_model(self, *, model_config, device_config):
        # ... 设备设置 ...
        quant_config = _get_quantization_config(model_config, self.load_config)
        with set_default_torch_dtype(model_config.dtype):
            with target_device:
                model = _initialize_model(model_config, self.load_config, quant_config)
            DefaultModelLoader.load_weights_and_postprocess(
                model, self._get_all_weights(model_config, model), target_device
```

```
            )
        return model.eval()

def get_model_loader(load_config, model_config=None):
    # 确保 RUNAI_STREAMER 始终避开 ModelOptModelLoader
    model_optloader_allowed = (
        model_config and load_config.load_format != LoadFormat.RUNAI_STREAMER
    )
    if model_optloader_allowed and (
        (hasattr(model_config, "modelopt_quant") and model_config.modelopt_quant)
        or model_config.quantization in [...]
    ):
        return ModelOptModelLoader(load_config)
    # ... 其他分支，最后 fallthrough 到 RunaiModelStreamerLoader
```

评论区精华

作者在 review 中对 _MODELOPT_QUANTIZATIONS 集合定义的可读性提出疑问（"Can this be less annoying to read somehow"），随后在后续 commit 中重构为更简洁的 model_optloader_allowed 变量，避免了额外常量集合。此评论仅涉及代码风格，无功能性争议。

- 代码可读性 - _MODELOPT_QUANTIZATIONS 集合定义 (style): 在后续 commit 中重构为 model_optloader_allowed 变量，避免了额外常量集合，使代码更简洁。

风险与影响

- 风险：
  1. 流式生成器模式要求迭代器只能消费一次，若其他代码多次遍历 stream_language_weights 可能导致数据丢失或重复加载。
  2. RunAI 流式加载路径增加了模型初始化的复杂度，可能影响加载性能，但对非 RunAI 用户无影响（通过路由保护）。
  3. 新标记属性 _sglang_runai_streamer_tensor 可能与第三方代码产生意外交互，但仅在 RunAI 流式路径中设置。
  4. Kimi 和 DeepSeek 的加载改动可能影响其他非 RunAI 的加载场景（如 HuggingFace 直接加载），但通过路由和条件克隆保护了原有逻辑。 - 影响：正面影响：支持 RunAI 用户直接流式加载量化检查点，无需本地复制全部权重，显著减少加载时间和存储开销。负面影响：无（非 RunAI 用户不受影响，路由确保兼容性）。影响范围主要针对使用 RunAI 流式加载的用户和配置（LoadFormat.RUNAI_STREAMER）。 - 风险标记：模型加载路径变更，流式生成器单次消费，缓冲区管理依赖标记属性

关联脉络

- 暂无明显关联 PR