PR #25847 完整报告

sgl-project/sglang

[diffusion] Cache fp32 layernorm params

合并时间：2026-05-25 22:56

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25847

执行摘要

- 一句话：缓存 diffusion 中 FP32LayerNorm 参数转换
- 推荐动作：值得关注缓存失效设计，测试覆盖全面。如使用扩散模型，建议合并。

功能与动机

Wan 扩散路径每次前向都调用 self.weight.float().to(device)，推理时参数不变，可缓存。从 FastVideo PR #1245 移植。

实现拆解

1. 在 FP32LayerNorm 类中添加 _cached_fp32_param 方法，构建 key（data_ptr、_version、device 等）缓存转换后的 fp32 张量，存储到 __dict__中避免污染 state_dict。
2. 修改 forward 方法，调用 _cached_fp32_param 取代直接转换，grad 模式下回退原路径。
3. 新增 test/unit/test_fp32_layernorm.py，包含 4 个单元测试覆盖正确性、缓存复用、参数更新失效、grad 模式行为。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/layernorm.py（模块 层归一化；类别 source；类型 core-logic；符号 _cached_fp32_param, forward）：核心变更，添加 _cached_fp32_param 缓存逻辑，修改 forward 使用缓存。
- python/sglang/multimodal_gen/test/unit/test_fp32_layernorm.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_fp32_layernorm_cache_matches_reference, test_fp32_layernorm_cache_reuses_converted_params, test_fp32_layernorm_cache_invalidates_on_param_update, test_fp32_layernorm_grad_mode_preserves_autograd_path）：新增单元测试，覆盖缓存正确性、复用、失效、grad 模式。

关键符号：_cached_fp32_param, forward

关键源码片段

python/sglang/multimodal_gen/runtime/layers/layernorm.py

核心变更，添加 _cached_fp32_param 缓存逻辑，修改 forward 使用缓存。

```
  def _cached_fp32_param(
```

```
        self, attr: str, param: torch.Tensor | None, device: torch.device
    ) -> torch.Tensor | None:
        # 参数为 None 时直接返回
        if param is None:
            return None

        # 保持 autograd 语义不变：若 grad 启用，直接转换（不缓存）
        if torch.is_grad_enabled():
            return param.float().to(device=device)

        # 构建 key：包含 data_ptr、_version、来源设备、目标设备、dtype
        key = (
            param.data_ptr(),
            param._version,
            param.device,
            device,
            param.dtype,
        )
        # 从实例的 __dict__ 中按 attr 名称查询缓存
        cache = self.__dict__.get(attr)
        if cache is not None and cache[0] == key:
            return cache[1]

        # 否则做转换并缓存
        fp32_param = param.detach().to(device=device, dtype=torch.float32)
        self.__dict__[attr] = (key, fp32_param)
        return fp32_param


    def forward(self, inputs: torch.Tensor) -> torch.Tensor:
        origin_dtype = inputs.dtype
        device = inputs.device
        weight = self._cached_fp32_param('_weight_fp32_cache', self.weight, device)
        bias = self._cached_fp32_param('_bias_fp32_cache', self.bias, device)
        return F.layer_norm(
            inputs.float(),
            self.normalized_shape,
            weight,
            bias,
            self.eps,
        ).to(origin_dtype)
```

python/sglang/multimodal_gen/test/unit/test_fp32_layernorm.py

新增单元测试，覆盖缓存正确性、复用、失效、grad 模式。

```
    @pytest.mark.skipif(not torch.cuda.is_available(), reason='CUDA required')
    def test_fp32_layernorm_cache_reuses_converted_params():
        norm = FP32LayerNorm(16, eps=1e-5).cuda().to(torch.bfloat16)
        inputs = torch.randn(4, 16, device='cuda', dtype=torch.bfloat16)
```

```
    with torch.no_grad():
        norm(inputs)
        weight_cache = norm.__dict__['_weight_fp32_cache']
        bias_cache = norm.__dict__['_bias_fp32_cache']

        norm(inputs)

    # 第二次调用应复用同一 tensor 对象
    assert norm.__dict__['_weight_fp32_cache'][1] is weight_cache[1]
    assert norm.__dict__['_bias_fp32_cache'][1] is bias_cache[1]
    # 缓存应不出现在 state_dict 中
    assert '_weight_fp32_cache' not in norm.state_dict()
    assert '_bias_fp32_cache' not in norm.state_dict()


@pytest.mark.skipif(not torch.cuda.is_available(), reason='CUDA required')
def test_fp32_layernorm_grad_mode_preserves_autograd_path():
    norm = FP32LayerNorm(16, eps=1e-5).cuda().to(torch.bfloat16)
    inputs = torch.randn(4, 16, device='cuda', dtype=torch.bfloat16, requires_grad=True)

    output = norm(inputs).float().sum()
    output.backward()

    # 应能正确反向传播
    assert inputs.grad is not None
    # 且缓存不被填充
    assert '_weight_fp32_cache' not in norm.__dict__
    assert '_bias_fp32_cache' not in norm.__dict__
```

评论区精华

无实质性讨论，直接合并。

- 暂无高价值评论线程

风险与影响

- 风险：缓存依赖 data_ptr 和 version，若参数被就地修改但版本未变（不常见），可能导致错误结果。测试覆盖了 add 操作，grad 模式完全跳过缓存，风险较低。
- 影响：仅影响扩散模型推理路径，训练无影响。性能提升微小（e2e 0.1%），但代码更清晰，为类似模式提供参考。
- 风险标记：缓存依赖参数指针和版本，测试覆盖完整，grad 模式回退

关联脉络

- 暂无明显关联 PR