PR #30331 完整报告

sgl-project/sglang

[Fix] Load HunyuanV3 NextN final_layernorm into the draft head's output norm

合并时间：2026-07-13 20:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30331

执行摘要

- 一句话：修复 HunyuanV3 MTP 权重加载错误
- 推荐动作：建议精读：展示了权重加载中 data-contract 变更的典型模式——当 checkpoint 格式与模型代码的命名约定不一致时，如何通过一个条件分支安全修复，并辅以完整的单元测试防止回归。对于 HunyuanV3 用户，此修复能直接提升 MTP 推理吞吐。

功能与动机

Released tencent/Hy3 checkpoints 中 MTP speculative decoding 比 plain decode 更慢。根因是 checkpoint 将 draft layer 的输出 norm 存储为 model.layers.80.final_layernorm.weight，但 HYV3ForCausalLMNextN.load_weights 将其错误映射到不存在的 model.decoder.final_layernorm.weight，导致权重被静默丢弃，draft head 使用默认初始化的输出 norm，每个 draft logit 都被扭曲，接受率崩溃。

实现拆解

1. 在 load_weights 中增加 elif 分支：位于 python/sglang/srt/models/hunyuan_v3_nextn.py:189-192，当 subname 以 final_layernorm 开头时，将 name 设置为 'model.shared_head.norm.weight'，使得该权重被正确加载到 draft head 的输出 norm 参数上。
2. 保留原有 bare-key 分支：对于预览时代的 checkpoint 仍可用 model.shared_head.norm.weight 直接匹配。
3. 新增单元测试文件：test/registered/unit/models/test_hunyuan_v3_nextn_weight_loading.py，使用 _FakeParam 模拟参数加载，测试四个场景：final_layernorm 映射到 shared_head.norm、spec 权重映射到 model 前缀、decoder 层权重映射到 decoder 前缀、embed/lm_head 被跳过。测试在 CPU 上运行，可通过 register_cpu_ci 注册到 CI。

关键文件：

- python/sglang/srt/models/hunyuan_v3_nextn.py（模块 模型加载；类别 source；类型 data-contract；符号 load_weights）：核心修复文件：在 load_weights 中增加 elif subname.startswith('final_layernorm') 分支，将权重正确路由到 model.shared_head.norm.weight。
- test/registered/unit/models/test_hunyuan_v3_nextn_weight_loading.py（模块 测试；类别 test；类型 test-coverage；符号 _FakeParam, init, weight_loader,

TestHunyuanV3NextNWeightLoading）：新增的 CPU 单元测试文件，使用 _FakeParam 模拟参数加载，覆盖四个关键映射类别：final_layernorm 到 shared_head.norm、spec 权重到 model 前缀、decoder 层到 decoder 前缀、embed/lm_head 跳过。

关键符号：load_weights

关键源码片段

python/sglang/srt/models/hunyuan_v3_nextn.py

核心修复文件：在 load_weights 中增加 elif subname.startswith('final_layernorm') 分支，将权重正确路由到 model.shared_head.norm.weight。

```
# python/sglang/srt/models/hunyuan_v3_nextn.py
# 在 load_weights 方法的循环中，处理 nextn 前缀下权重名的分支逻辑

for name, loaded_weight in weights:
    if name.startswith(nextn_prefix):
        subname = name[len(nextn_prefix):]
        if any(subname.startswith(s) for s in spec_weight_names):
            name = f"model.{subname}"
        elif subname.startswith("final_layernorm"):
            # 新增分支：将 released checkpoint 中的 final_layernorm 权重
            # 映射到 draft head 的输出 norm 参数上，
            # 而非不存在的 decoder 参数。
            name = "model.shared_head.norm.weight"
        else:
            name = f"model.decoder.{subname}"
    elif name == "model.shared_head.norm.weight":
        pass
    elif (
        "embed_tokens" in name
        or "shared_head.head" in name
        or "lm_head" in name
    ):
        continue
    else:
        continue
```

test/registered/unit/models/test_hunyuan_v3_nextn_weight_loading.py

新增的 CPU 单元测试文件，使用 _FakeParam 模拟参数加载，覆盖四个关键映射类别：final_layernorm 到 shared_head.norm、spec 权重到 model 前缀、decoder 层到 decoder 前缀、embed/lm_head 跳过。

```
# test/registered/unit/models/test_hunyuan_v3_nextn_weight_loading.py
# 使用 _FakeParam 模拟模型参数，验证权重加载映射是否正确

class _FakeParam:
    def __init__(self):
        self.loaded = None
```

```
    def weight_loader(self, param, loaded_weight, *args, **kwargs):
        self.loaded = (param, loaded_weight, args, kwargs)

class TestHunyuanV3NextNWeightLoading(unittest.TestCase):
    def _make_minimal_model(self, named_parameters=()):
        model = object.__new__(HYV3ForCausalLMNextN)
        model.config = SimpleNamespace(num_hidden_layers=80, num_experts=2)
        model.named_parameters = lambda: iter(named_parameters)
        return model

    def test_final_layernorm_loads_into_shared_head_norm(self):
        # 核心回归测试：验证 final_layernorm 被正确映射到 shared_head.norm
        param = _FakeParam()
        model = self._make_minimal_model([("model.shared_head.norm.weight", param)])
        loaded_weight = torch.ones(1)
        model.load_weights([("model.layers.80.final_layernorm.weight", loaded_weight)])
        self.assertEqual(param.loaded, (param, loaded_weight, (), {}))

    def test_spec_weights_map_to_model_prefix(self):
        # 验证 spec 权重（enorm/hnorm/eh_proj）映射到 model 前缀
        params = {
            "model.enorm.weight": _FakeParam(),
            "model.hnorm.weight": _FakeParam(),
            "model.eh_proj.weight": _FakeParam(),
        }
        model = self._make_minimal_model(list(params.items()))
        weights = [
            ("model.layers.80.enorm.weight", torch.ones(1)),
            ("model.layers.80.hnorm.weight", torch.ones(1)),
            ("model.layers.80.eh_proj.weight", torch.ones(1)),
        ]
        model.load_weights(weights)
        for name, param in params.items():
            self.assertIsNotNone(param.loaded, f"{name} was not loaded")

    def test_decoder_layer_weight_maps_to_decoder_prefix(self):
        # 验证 decoder 层权重映射到 decoder 前缀
        param = _FakeParam()
        model = self._make_minimal_model(
            [("model.decoder.input_layernorm.weight", param)]
        )
        loaded_weight = torch.ones(1)
        model.load_weights([("model.layers.80.input_layernorm.weight", loaded_weight)])
        self.assertEqual(param.loaded, (param, loaded_weight, (), {}))

    def test_embed_tokens_and_lm_head_are_skipped(self):
        # 验证 embed_tokens 和 lm_head 被正确跳过
        params = {
            "model.embed_tokens.weight": _FakeParam(),
```

```
        "lm_head.weight": _FakeParam(),
    }
    model = self._make_minimal_model(list(params.items()))
    weights = [
        ("model.embed_tokens.weight", torch.ones(1)),
        ("lm_head.weight", torch.ones(1)),
    ]
    model.load_weights(weights)
    for name, param in params.items():
        self.assertIsNone(param.loaded, f"{name} should have been skipped")
```

评论区精华

无审查评论。PR body 和 commit message 已详细说明根因和修复方案。

- 暂无高价值评论线程

风险与影响

- 风险：变更仅增加一个 elif 分支，路径覆盖：final_layernorm 被显式路由到 shared_head.norm；其他 spec 权重保持原有映射；decoder 层权重仍映射到 model.decoder 前缀。新增的单元测试验证了所有关键路径，回归风险极低。
- 影响：影响范围：仅影响 HunyuanV3 MTP 权重加载逻辑，不影响其他模型或非 MTP 模式。影响程度：修复后 MTP 解码 tok/s 从 54–56 提升至 87–89（与 plain decode 持平），并额外获得推测解码的延迟优势，对使用 tencent/Hy3 检查点的用户是显著的性能提升。
- 风险标记：暂无

关联脉络

- PR #29909 [Bugfix][NPU] Fix Hunyuan3 model where MoE's routing_scaling_ratio is missing on NPU: 同为 Hunyuan3 模型相关的 bugfix，显示该模型近期有多项修复。