PR #26347 完整报告

sgl-project/sglang

Support for Zyphra zaya1 model

合并时间：2026-06-10 17:44

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26347

执行摘要

- 一句话：新增 Zyphra ZAYA1 混合模型推理支持
- 推荐动作：该 PR 设计严谨，值得仔细阅读。重点关注 CCA 状态管理策略、MOD+TP 的混合运算顺序、以及 CCA 头并行实现。测试覆盖全面，为后续引入类似混合模型提供了可参考的范本。

功能与动机

Zyphra ZAYA1 是目前尚无 SGLang 上游实现的新型混合模型，用户无法直接使用。本 PR 旨在通过标准 sglang.serve 入口点为 ZAYA1 提供完整的推理支持，包括 CCA 注意力与 MoE 的合理集成。

实现拆解

1. 配置注册：在 python/sglang/srt/configs/zaya.py 中定义 ZayaConfig，继承 PretrainedConfig，注册 model_type="zaya"，支持从 HuggingFace Hub 加载。
2. 模型核心实现：在 python/sglang/srt/models/zaya.py 中实现全部模块：
   - ResidualScaling：fp32 残差缩放，每层独立参数。
   - CCA：深度可分离卷积生成 QKV，带可学习每 K 头温度，通过 MambaPool 管理每请求卷积状态。
   - ZayaAttention：将 CCA 输出接入 RadixAttention。
   - ZayaRouter：3 层 MLP 路由，支持 EDA 和 MOD。
   - ZayaBlock：组合 FusedMoE 与 MOD 混合。
   - ZayaForCausalLM：模型入口，包含词嵌入与 fp32 LM 头。
3. 状态管理：CCA 的卷积状态和 prev_hs 通过 MambaPool 集中管理，每请求每层索引，支持 prefill/decode 边界状态传递。
4. TP 支持：CCA 头并行（Q/K 头拆分）、MOD 预掩码修正（先 mask 专家输出再 all-reduce，避免 skip 路径被 tp_size 乘算）、RowParallelLinear 输出投影。
5. 测试与文档：CPU 单元测试覆盖 CCA 数值、MOD+TP 运算、配置字段；端到端测试验证服务与 MMLU 子集；文档详细说明架构与状态缓存策略。

关键文件：

- python/sglang/srt/models/zaya.py（模块 模型实现；类别 source；类型 core-logic；符号 ResidualScaling, CCA, ZayaAttention, ZayaBlock）：核心模型实现，包含 CCA、MoE、MOD、ResidualScaling 等全部模块，新增 1658 行。
- python/sglang/srt/configs/zaya.py（模块 模型配置；类别 source；类型 configuration；符号 ZayaConfig, register_zaya_config）：ZAYA1 模型配置类，注册 model_type，管理参数默认值与 HF 集成。
- test/registered/unit/models/test_zaya_cca.py（模块 CCA 测试；类别 test；类型 test-coverage；符号 _ensure_dist_initialized, _MockReqToTokenPool, TestZayaCCA）：CCA 模块数值与状态缓存正确性测试，含 TP 模拟与批量回归保护。
- test/registered/unit/models/test_zaya_mod_tp.py（模块 MOD-TP 测试；类别 test；类型 test-coverage；符号 _reference_blend, _real_tp_blend, _buggy_old_tp_blend, TestZayaMODUnderTP）：MOD 混合在 TP>1 下的正确性验证，包含旧 bug 的反向回归。
- test/registered/models/test_zaya.py（模块 端到端测试；类别 test；类型 test-coverage；符号 TestZayaServer, _zaya_enabled）：端到端服务测试，启动真实服务器验证生成与 MMLU 子集。
- test/registered/unit/configs/test_zaya_config.py（模块 配置测试；类别 test；类型 test-coverage；符号 TestZayaConfig）：ZayaConfig 单元测试，验证默认值、rope_parameters 派生、注册等。

关键符号：ResidualScaling.init, ResidualScaling.forward, CCA.init, CCA.forward, CCA._forward_extend, CCA._forward_decode, CCA._get_pool_state, ZayaRouter.forward, ZayaBlock.forward, ZayaForCausalLM.forward, mod_premask_experts, mod_blend

关键源码片段

python/sglang/srt/models/zaya.py

核心模型实现，包含 CCA、MoE、MOD、ResidualScaling 等全部模块，新增 1658 行。

```
class ResidualScaling(nn.Module):
    """Affine fp32 scaling applied to the residual / hidden_states streams.

    Layer 0 has no incoming residual stream, so its checkpoint omits
    ``residual_scale`` / ``residual_bias`` and ``has_residual`` stays False.
    """

    def __init__(self, config: ZayaConfig, layer_n: int) -> None:
        super().__init__()
        self.hidden_size = config.hidden_size
        self.has_residual = layer_n != 0
        # Params are always fp32 regardless of model dtype
        self.hidden_states_scale = nn.Parameter(torch.ones(self.hidden_size))
        self.hidden_states_bias = nn.Parameter(torch.zeros(self.hidden_size))
        if self.has_residual:
            self.residual_scale = nn.Parameter(torch.ones(self.hidden_size))
            self.residual_bias = nn.Parameter(torch.zeros(self.hidden_size))
```

```
        else:
            # Layer 0: register buffers so checkpoint loading doesn't crash
            self.register_buffer("residual_scale", None)
            self.register_buffer("residual_bias", None)

    def forward(self, hidden_states: torch.Tensor,
                residual: Optional[torch.Tensor] = None):
        # Apply scale/bias to the hidden states stream
        hidden_states = hidden_states.float() * self.hidden_states_scale + self.hidden_states_bias
        if residual is not None:
            residual = residual.float() * self.residual_scale + self.residual_bias
        return hidden_states, residual
```

评论区精华

- MOD TP 正确性：alexnails 指出 MOD 混合中 mod_out 是复制张量，all-reduce 后会被乘以 tp_size，导致 TP>1 时 skip 路径错误。ChengYao 修正为先对每 rank 专家输出应用 mask 再 reduce，最后叠加 skip path，并添加单元测试（test_zaya_mod_tp.py）。
- RMSNorm fp32 效率：alexnails 发现 fp32 残差流导致每层 RMSNorm 进入 eager 模式，产生多次 kernel launch。ChengYao 通过修改 residual 传入 dtype 或融合操作修复。
- CCA 卷积批处理：alexnails 建议将无 prefix 请求的卷积操作合并为一次调用。ChengYao 实现了 all_fresh 路径，将所有请求打包成单个卷积输入。
- mamba_indices 同步优化：alexnails 指出每层调用 item() 产生同步开销。ChengYao 改为在循环前一次 tolist()，下标访问。
- CCA TP 头并行：alexnails 提议将 CCA 的 grouped-mean 和卷积头拆分。ChengYao 在 PR 内实现，确保 head count 可整除。
  - MOD + TP 混合运算正确性 (correctness): ChengYao 修复为先对每 rank 专家输出应用 mask 再 reduce，最后叠加 skip path，并添加单元测试验证 tp_size∈{2,4,8}。
  - CCA 卷积状态同步与批处理优化 (performance): ChengYao 实现 all_fresh 路径打包卷积；mamba_indices 改为循环前一次 tolist()，下标访问。
  - RMSNorm fp32 residua 融合开销 (performance): ChengYao 通过让 residual 保持 fp32 并修改 Norm 调用，减少不必要的类型转换。
  - CCA 头并行 TP 支持 (performance): ChengYao 在 PR 内实现 CCA head-parallel TP，约束 head counts 必须可整除。

风险与影响

- 风险：核心风险在于 CCA 状态管理与标准 SGLang 后端（CUDA Graph、RadixAttention）的兼容性：PR 修复了与 prefix cache 的冲突，但任何新的状态管理类（MambaPool）都可能与其他优化交互（如 mixed-chunk 调度）。MOD TP 的数学修正虽然经过单元测试，但真实多 Rank 环境仍需验证数值稳定性。性能方面，CCA 的串行卷积在长序列下可能成为瓶颈，虽有批处理优化但预填充 +prefix 场景仍逐请求处理。
- 影响：用户可直接通过 --model-path Zyphra/ZAYA1-base 加载 ZAYA1 模型，NVIDIA 和 AMD GPU 均受支持。服务端内存消耗因 CCA 状态池增加约 60 MB（80 层×80 并发），

可忽略不计。对现有模型无影响，为纯新增功能。

- 风险标记：CCA 状态管理与 CUDA Graph 兼容性，MOD TP 数学修正需严格验证，性能敏感路径（RMSNorm、卷积批处理）

关联脉络

- 暂无明显关联 PR