PR #33561 完整报告

sgl-project/sglang

[Model] Support Ling-3.0-flash (BailingMoeV3)

合并时间：2026-08-27 08:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33561

执行摘要

- 一句话：新增 Ling-3.0-flash 混合 MoE 模型支持，含 DSPARK 投机解码
- 推荐动作：值得精读，尤其建议关注三点设计：metadata_glue_graph 以 "capture 设备算子 + 快照恢复宿主对象 " 消除 bs=1 spec decode 的 host 开销，及其对 host-fed plan 的静默失败防线；KDA safe-gate 如何在共享接口上以 " 不支持则回退 Triton 内核 " 的方式向后兼容；routed scaling 双应用这类跨 helper 交互 bug 的排查方法。对只想使用模型的人，直接参考 PR body 的 benchmark 表与部署 recipe（#34363）。

功能与动机

PR body 明确目标是 "Day-0 support for inclusionAI/Ling-3.0-flash"：在新模型发布当天让 sglang 能直接加载推理。Ling-3.0-flash 是混合架构（KDA 线性注意力 + MLA，512-expert MoE），且官方 checkpoint 同时发布 BF16/FP8/INT4/MXFP4 四种格式，因此 Day-0 意味着模型、投机解码、量化三个维度都要就绪；为此共享的 KDA 接口必须支持 safe-gate，DSPARK 需要为 CUDA Graph 折叠贪心草稿步。

实现拆解

1. 核心模型与配置：新增 python/sglang/srt/models/bailing_moe_v3.py（约 1982 行），实现 BailingMoeV3ForCausalLM、DsV3MLA、BailingMoELinearDecoderLayer。DsV3MLA 继承 DeepseekV2AttentionMLA，通过 _forward_gated / _apply_gated 将 KDA 层的 safe-gate 下界并入注意力输出，gate 以追加 inner_state 元组的形式传给各 forward_core 实现；KimiDeltaAttention 被参数化供 V3 复用。该文件是 Theta fork 约 37 个内部 commit 的累积移植。model_config.py 同步接入 bailing_hybrid，并以 use_kda 区分 V2.5（lightning）与 V3（KDA）。
2. KDA 注意力后端与 safe-gate 传播：python/sglang/srt/layers/attention/linear/kda_backend.py 的 decode / extend 收到 lower_bound 且当前内核不支持 safe-gate 时回退到 Triton 内核，保证旧内核向后兼容；新增 _can_run_fused_chain_verify 门控，为 MTP chain-verify 选择 fused_kda_conv_gating_verify 融合内核（替代 transpose-copy + conv1d + 递推序列），并修复其 conv_state 必须是 stride 1 连续布局的调用约束。
3. 投机解码：bailing_moe_nextn.py 以 _is_bailing_moe_v3_config 让 NEXTN 草稿层在 V3 时选用 BailingMoeV3DecoderLayer 并透传 num_fused_shared_experts；dspark.py 注册 LingDSparkModel，为 VanillaMarkov 新增 sample_block_greedy_fused（单内核完成 bias-dot + add + argmax），run_markov_block 增加 collect_corrected 开关；同时修

复 TARGET_VERIFY 后 KDA/Mamba 状态提交，并在 pool_configurator.py 中修正 DSA 草稿 KV 池按真实 per-token 成本计算以避免 OOM。新增 metadata_glue_graph.py 将每个 replay 的 attention metadata prep 捕获为按 key 复用的 CUDA Graph。

4. 量化与精度修复：覆盖 BF16/FP8/compressed-tensors INT4/native MXFP4，分别路由到 Marlin、Triton WNA16、FlashInfer CUTLASS 等后端；修复 native MXFP4 路径 routed scaling 被应用两次（折入 topk_weights 后 maybe_fuse_routed_scale_and_shared_add 又应用一次）导致的 DSV4-FP4 accept-length 回归；修复 Blackwell INT4 图捕获、MXFP4 CUTLASS 忽略 Bailing expert clamp（gemm1_clamp_limit）等问题。
5. 解析器、测试与 CI：新增 python/sglang/srt/function_call/ling3_detector.py 并在 reasoning_parser.py 注册，支持 ling3 推理与流式工具调用解析。测试覆盖 fused KDA verify 与 unfused 参考的一致性、full GSM8K DSpark 端到端（4x B200 TP4）、shared-experts fusion-gate 布局约束（INT4 混合布局禁融合）、compressed-tensors WNA16 Blackwell 自动选择、parser 流式工具调用等，并注册进 base-b / extra / ROCm CI。

关键文件：

- python/sglang/srt/models/bailing_moe_v3.py（模块 模型实现；类别 source；类型 core-logic；符号 DsV3MLA, _forward_gated, _apply_gated, resolve_nextn_layer_id）：核心新增模型文件（约 1982 行），实现 BailingMoeV3ForCausalLM、DsV3MLA gated attention、KDA safe-gate 传递与 512-expert MoE，是本次 Day-0 支持的主体。
- python/sglang/srt/model_executor/runner/metadata_glue_graph.py（模块 图捕获；类别 source；类型 core-logic；符号 MetadataGlueGraph, run, reset, _leaves）：新增通用机制：把 spec decode 每 replay 的 attention metadata prep 捕获为按 key 复用的 CUDA Graph，消除 host 侧 op soup 开销，是本 PR 对框架层最有通用价值的优化。
- python/sglang/srt/models/bailing_moe_nextn.py（模块 模型实现；类别 source；类型 data-contract；符号 _is_bailing_moe_v3_config, BailingMoEModelNextN, shared_experts_fusion_disable_reason, weight_direct_load）：NEXTN（MTP）草稿模型的关键分支：用 use_kda 区分 Ling-V3 与 V2.5，并为 V3 选择对应的 KDA + gated-MLA 解码层与融合共享专家透传。
- python/sglang/srt/models/dspark.py（模块 投机解码；类别 source；类型 core-logic；符号 sample_block_greedy_fused, run_markov_block, LingDSparkModel）：DSpark 投机解码支持的核心：注册 LingDSparkModel、新增 sample_block_greedy_fused 融合贪心采样，并让 run_markov_block 支持跳过 corrected_logits 收集以适配图折叠。
- python/sglang/srt/layers/attention/linear/kda_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 _can_run_fused_chain_verify, decode, extend）：KDA 注意力后端：decode/extend 在 lower_bound 存在时回退 Triton 内核实现 safe-gate，并新增 fused chain-verify 快速路径选择逻辑。
- python/sglang/srt/layers/attention/flashinfer_mla_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 _build_fast_verify_plan_kwargs）：扩展 fast verify plan 构造，支持 Ling-V3 混合 KDA + MLA 目标的验证计划，是本 PR 对现有 MLA 后端最重要的修改。

- python/sglang/srt/parser/reasoning_parser.py（模块 解析器；类别 source；类型 core-logic；符号 Ling3Detector, detect_and_parse）：注册 Ling3Detector 推理 / 工具调用解析器，是 Day-0 可用性的一部分（思维链与工具调用格式解析）。
- test/registered/kernels/test_fused_kda_conv_recurrent_verify.py（模块 内核测试；类别 test；类型 test-coverage；符号 test_matches_unfused_reference, _run_reference, _run_fused, _compare_case）：新增 fused KDA conv+recurrent verify 与 unfused 参考的逐例对比测试，覆盖多种 batch/head/ 负槽位 / 下界场景，守护 MTP chain-verify 正确性。

关键符号：BailingMoeV3ForCausalLM.forward, DsV3MLA._forward_gated, DsV3MLA._apply_gated, BailingMoeV3ForCausalLM.determine_num_fused_shared_experts, BailingMoeV3ForCausalLM.resolve_nextn_layer_id, BailingMoeV3ForCausalLM.rewrite_nextn_weight_name, BailingMoEModelNextN.init, VanillaMarkov.sample_block_greedy_fused, MetadataGlueGraph.run, KDAAttnBackend._can_run_fused_chain_verify, Ling3Detector.detect_and_parse

关键源码片段

python/sglang/srt/model_executor/runner/metadata_glue_graph.py

新增通用机制：把 spec decode 每 replay 的 attention metadata prep 捕获为按 key 复用的 CUDA Graph，消除 host 侧 op soup 开销，是本 PR 对框架层最有通用价值的优化。

```
# MetadataGlueGraph：把 spec decode 的 metadata prep 捕获成 CUDA Graph
# decode_cuda_graph_runner.load_batch 在每个 replay 都会跑一遍
# init_forward_metadata_out_graph，bs=1 spec decode 下这是几十个小算子，
# host 调度开销主导了阶段间缝隙；按 replay key 捕获一次即可把每步开销
# 压到一次图发射。关键约束：Python 侧分支在 key 内必须恒定。

class MetadataGlueGraph:
    NUM_WARMUP = 2

    def __init__(self, device):
        self.device = device
        self.disabled = False
        self._states: Dict[Any, dict] = {}
        self._capture_stream = None

    def reset(self):
        # 丢弃已捕获图：runner 重捕时静态缓冲区与后端状态可能已重建
        self._states.clear()

    @staticmethod
    def _leaves(attn_backend) -> List[Any]:
        # 需覆盖主后端及其挂载的后端列表（例如 MLA + KDA 混合验证）
        backends = [attn_backend]
        if attn_backend.attn_backend_list is not None:
            backends.extend(attn_backend.attn_backend_list)
```

```
        return backends

    def run(self, attn_backend, fb_view, key) -> None:
        st = self._states.get(key)
        if st is None:
            st = {"warmups": 0, "graph": None, "meta": None}
            self._states[key] = st

        if st["graph"] is not None:
            # 快照的 forward_metadata 对象每次 replay 前重新挂回后端,
            # 图只重放刷新这些对象所指向 tensor 的设备算子
            for backend, metadata in st["meta"]:
                backend.forward_metadata = metadata
            st["graph"].replay()
            return

        if st["warmups"] < self.NUM_WARMUP:
            st["warmups"] += 1
            attn_backend.init_forward_metadata_out_graph(fb_view)
            return

        if self._capture_stream is None:
            self._capture_stream = torch.cuda.Stream()
        graph = torch.cuda.CUDAGraph()
        try:
            # 捕获只记录不执行，之后需要 replay 一次完成本次 prep
            with torch.cuda.graph(graph, stream=self._capture_stream):
                attn_backend.init_forward_metadata_out_graph(fb_view)
        except Exception:
            logger.warning(
                "Metadata glue-graph capture failed for key %s; falling back "
                "to eager metadata prep permanently.",
                key,
                exc_info=True,
            )
            self.disabled = True
            attn_backend.init_forward_metadata_out_graph(fb_view)
            return

        # 捕获 Python 副作用（各后端 forward_metadata 对象）以便 replay 前恢复
        st["meta"] = [(b, b.forward_metadata) for b in self._leaves(attn_backend)]
        st["graph"] = graph
        graph.replay()
```

python/sglang/srt/models/bailing_moe_nextn.py

NEXTN（MTP）草稿模型的关键分支：用 use_kda 区分 Ling-V3 与 V2.5，并为 V3 选择对应的 KDA + gated-MLA 解码层与融合共享专家透传。

```
# bailing_moe_nextn.py: _is_bailing_moe_v3_config 用 use_kda 区分两代
```

```
# 混合架构 checkpoint；NEXTN 草稿层据此选择 V2.5 的 lightning 解码层还是
# V3 的 KDA + gated-MLA 解码层，并把融合共享专家数透传给草稿层。

def _is_bailing_moe_v3_config(config: PretrainedConfig) -> bool:
    # use_kda 由 BailingHybridConfig 根据是否存在 short conv 设置，
    # 恰好是两代模型的分界点
    return config.model_type == "bailing_hybrid" and config.use_kda


class BailingMoEModelNextN(nn.Module):
    def __init__(
        self,
        config: PretrainedConfig,
        quant_config: Optional[QuantizationConfig] = None,
        prefix: str = "",
        num_fused_shared_experts: int = 0,
    ) -> None:
        ...
        self.is_hybrid = (
            hasattr(config, "model_type") and config.model_type == "bailing_hybrid"
        )
        if self.is_hybrid:
            config.attention_type = 1
            decoder_layer_cls = BailingMoELinearDecoderLayer # V2.5 默认路径
            decoder_kwargs = {
                "quant_config": quant_config,
                "layer_id": 0,
                "is_nextn": True,
                "prefix": add_prefix(f"layers.{config.num_hidden_layers}", prefix),
            }
            if _is_bailing_moe_v3_config(config):
                # Ling-V3 换用 V3 的 KDA + gated-MLA 解码层，
                # 并透传融合共享专家数，保持与主模型 fusion-gate 一致
                decoder_layer_cls = BailingMoeV3DecoderLayer
                decoder_kwargs["num_fused_shared_experts"] = num_fused_shared_experts
            self.decoder = decoder_layer_cls(config, **decoder_kwargs)
        else:
            self.decoder = BailingMoEBlock(
                config,
                0,
                quant_config=quant_config,
                prefix=add_prefix("decoder", prefix),
            )
```

python/sglang/srt/models/dspark.py

DSpark 投机解码支持的核心：注册 LingDSparkModel、新增 sample_block_greedy_fused 融合贪心采样，并让 run_markov_block 支持跳过 corrected_logits 收集以适配图折叠。

```
# dspark.py：VanillaMarkov.sample_block_greedy_fused 用单个内核
```

```
# （MarkovGreedyStep）完成 bias-dot + add + argmax，替代 GEMV + add +
# 两遍 argmax，且不物化全词表 bias / step logits，是 DSPARK 支持
# CUDA Graph 图折叠的关键。

def sample_block_greedy_fused(
    self,
    base_logits: torch.Tensor,
    *,
    first_prev_tokens: torch.Tensor,
) -> Optional[torch.Tensor]:
    if not base_logits.is_cuda:
        # 非 CUDA 环境回退到普通 sample_block
        return None
    batch_size, proposal_len = base_logits.shape[:2]
    if proposal_len == 0:
        return torch.empty(
            batch_size, 0, dtype=torch.long, device=base_logits.device
        )
    sampled_tokens = []
    prev_tokens = first_prev_tokens.long()
    for step_idx in range(proposal_len):
        prev_embeds = self.get_prev_embeddings(prev_tokens)
        # 融合内核直接根据 base_logits、prev embedding 与 w2 权重产出贪心 token
        prev_tokens = MarkovGreedyStep.execute(
            base_logits=base_logits[:, step_idx, :],
            prev_embeds=prev_embeds,
            w2_weight=self.markov_w2.weight,
        )
        sampled_tokens.append(prev_tokens)
    return torch.stack(sampled_tokens, dim=1)
```

评论区精华

Fridge003 在 review 中给出 DISMISSED 结论："This PR is really messy. We need some cleaning before merge"，这是最大争议点。yuan-luo 回应解释了 mess 的结构性根源：Ling-3.0-flash 是混合 KDA/MLA MoE，KDA safe-gate 必须穿过公共接口，DSPARK 图折叠影响框架层，四种量化 checkpoint 共享 MoE/quant 改动，核心部分无法独立验证。JustinTong0323 补充量化论证：约 60% 是自包含新模型代码，共享改动约 75% 是 gated 新增或语义保持重构，并列出 6 处行为变更与证据表；同时定位了 DSV4-FP4 accept-length 回归的根因（MXFP4 路径 routed scaling 双应用），在 ee800a63 修复。

- PR 是否应拆分再合并 (design): 维持单 PR 合并，但作者承认代码库 messy；后续通过 #36584 等修复持续打磨。
- DSV4-FP4 spec accept-length 回归根因 (correctness): ee800a63 修复：helper 在已折叠折入 topk 时不再重复应用，回归验证通过。
- MXFP4 CUTLASS 路径丢弃 expert clamp (correctness): 已合入 PR 头部，23 行 / 2 文件修复。

- metadata glue graph 的静默失败契约 (design): 以保守禁用换取安全，设计被合入方接受。
- Ling3 parser 默认值与 thinking-on 语义对齐 (question): 已修复并合入。

风险与影响

- 风险： 1) 共享路径回归：6 处改变现有模型行为的改动集中在 MoE 量化与融合路径，DeepSeek 系模型在回归半径内，DSV4-FP4 已在 PR 内实际触发 accept-length 回归。 2) metadata_glue_graph 静默失败：只捕获设备算子，host 侧写 plan 的 DFlash family 若被 glue 会 accept length 塌缩且无报错，只能靠调用方强制禁用。 3) 长分支合并：85 个 commit 多次 merge main，冲突集中在 fused_moe_triton_kernels、kda_triton、flashinfer_mla_backend、compressed_tensors；后期仍在修 bring-up 问题，多后端 x 多量化组合验证深度有限。 4) 部署约束：DSPARK 路径依赖 decode CUDA Graph，文档明确所有验证未使用 --disable-cuda-graph。
  - 影响：用户获得 Ling-3.0-flash Day-0 推理，H200/B200 上 BF16/FP8/INT4/MXFP4 全格式可用，DSPARK 在 4x B200 TP4 full GSM8K 96.66%、0.00% error rate。系统层面，KDA safe-gate、fused chain-verify、metadata glue graph、融合贪心 Markov、router-gate matvec 单发射等被设计为通用机制，惠及后续混合线性注意力模型与宽专家 MoE。团队层面，单 PR 与拆分之争暴露了大型模型移植的组织问题，合入后仍需 #36584 等后继修复持续打磨，维护成本偏高。
  - 风险标记：共享路径行为变更，量化双重缩放回归，glue graph 静默失败隐患，长分支合并冲突，DSPARK 依赖 CUDA Graph

关联脉络

- PR #36584 Fix BailingMoeV3 reading enable_dp_lm_head off live topology instead of config: 本 PR 合入后对 BailingMoeV3 的后续 bugfix：修正 enable_dp_lm_head 从 live topology 而非 config 读取，说明模型支持仍在迭代打磨。
- PR #36309 [AMD][Bugfix] Skip invalid fused MoE reduction for direct top-1 output: 与本 PR 同处 fused MoE / shared experts 共享路径，均为 MoE 运行时的正确性修复。
- PR #33871 [Performance] Reduce idle DP work in breakable prefill CUDA graphs: 同属 KDA / 混合线性注意力架构的调度与 attention 优化线，与本 PR 的 KDA backend 改动相互印证。