PR #36186 完整报告

sgl-project/sglang

[Model] Support Nemotron 3.5 Lightning speculative decoding

合并时间：2026-08-26 07:43

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36186

执行摘要

- 一句话：支持 Nemotron 3.5 Lightning DFlash/DSpark 推测解码
- 推荐动作：值得精读。该 PR 展示了如何以低侵入方式将新模型接入现有推测解码框架，尤其是 " 从 checkpoint 配置派生布局行为、避免新增 ServerArgs 开关 " 的设计决策，以及用结构检查保护通用路径的做法。建议关注 dspark_config.get_dspark_sample_from_anchor 的兼容策略和 dflash._logical_linear_weight_shape 的 packed 权重处理。

功能与动机

PR body 明确说明动机：为 NVIDIA Nemotron 3.5 Lightning 添加干净的、最小侵入的推测解码支持，取代 #33554 的实现探索路径，同时保持有用的 ModelOpt 和 Nemotron 模型改动范围聚焦。官方发布的 DFlash/DSpark 草稿 checkpoint 是 W4A16 NVFP4 格式，因此必须先补齐 ModelOpt W4A16 NVFP4 支持；同时需要为避免影响通用 DFlash/DSpark 行为而将 Nemotron 3.5 专属布局逻辑用结构检查保护起来。

实现拆解

1. 补齐 ModelOpt W4A16 NVFP4 量化支持：修改 python/sglang/srt/layers/quantization/modelopt_quant.py，使官方的 NVFP4 draft checkpoint 能被正确加载，这是启用后续 DFlash/DSpark 路径的前提。
2. 改造 DFlash 草稿模型：在 python/sglang/srt/models/dflash.py 中支持 Nemotron 3.5 draft 自带 embedding/LM head 的布局，新增 _logical_linear_weight_shape 处理 packed 权重的逻辑 shape 推断，并引入 sharded_weight_loader 与 ReplicatedLinear；在 python/sglang/srt/speculative/dflash_utils.py 新增 is_nemotron_35_draft_config 结构检查；在 python/sglang/srt/models/nemotron_h.py 新增 set_dflash_layers_to_capture 与 aux hidden states 捕获逻辑，用于 DFlash 的 residual-layer capture；dflash_worker_v2.py 增加 _resolve_dflash_embedding_module 支持 draft 自带 embedding。
3. 改造 DSpark 草稿模型：在 python/sglang/srt/models/dspark.py 新增 Nemotron35VanillaMarkov（可量化的 Markov head）和 build_nemotron_35_markov_head 工厂函数；在 python/sglang/srt/speculative/dspark_components/dspark_draft.py 中新增 select_draft_hidden_without_anchor，并在 no-anchor 路径上避免 contiguous() 开销，同时将 query_token_num 调整为 gamma + 1；在 dspark_config.py 新增 get_dspark_sample_from_anchor 从 checkpoint 顶层字段

读取，缺失时默认 True 保持向后兼容。

4. 解析 Blackwell 注意力后端：修改 python/sglang/srt/arg_groups/speculative_hook.py 与 Nemotron-H overrides，在 SM100 上默认使用 trtllm_mha 作为目标 / 草稿后端，但尊重用户显式指定的 prefill_attention_backend、decode_attention_backend 等选项。
5. 测试与配套：test/registered/unit/test_model_overrides.py 新增 TestDSparkCheckpointConfig 及 Nemotron-H 在不同 SM 架构下 speculative 后端矩阵的覆盖；test/registered/models_e2e/test_nvidia_nemotron_3_nano.py 替换为 NVFP4 Normal/DFlash/DSpark 三个 GSM8K 用例（est_time=500、4-gpu-b200）；cookbook 安装命令更新为从 main 分支安装。

关键文件：

- python/sglang/srt/models/dflash.py（模块 推测解码；类别 source；类型 data-contract；符号 _logical_linear_weight_shape, init, get_input_embeddings）：DFlash 草稿模型核心改造：支持 Nemotron 3.5 自带 embedding/LM head、packed 投影权重与 residual 层捕获，新增 _logical_linear_weight_shape 处理量化权重逻辑形状。
- python/sglang/srt/models/dspark.py（模块 推测解码；类别 source；类型 data-contract；符号 Nemotron35VanillaMarkov, init, project_bias, build_nemotron_35_markov_head）：DSpark 草稿模型核心改造：新增可量化的 Nemotron35VanillaMarkov 与 build_nemotron_35_markov_head，并让 sample_from_anchor 从 draft config 派生。
- python/sglang/srt/speculative/dspark_components/dspark_draft.py（模块 推测解码；类别 source；类型 core-logic；符号 select_draft_hidden_without_anchor）：DSpark 采样路径核心改造：新增 select_draft_hidden_without_anchor 移除 query anchor，并按 sample_from_anchor 条件调整 block shape 与 embedding 来源，避免 legacy 路径 contiguous() 开销。
- python/sglang/srt/models/nemotron_h.py（模块 模型层；类别 source；类型 data-contract；符号 set_dflash_layers_to_capture）：新增 set_dflash_layers_to_capture 与 aux hidden states 捕获逻辑，为 DFlash 提供 residual-layer capture 能力。
- python/sglang/srt/speculative/dspark_components/dspark_config.py（模块 推测解码；类别 source；类型 core-logic；符号 get_dspark_sample_from_anchor, read_draft_checkpoint_gamma, read_draft_checkpoint_config）：新增 get_dspark_sample_from_anchor，从 checkpoint 顶层字段读取布局标志，缺失时默认 True，避免依赖模型类型推断。
- python/sglang/srt/speculative/dflash_utils.py（模块 推测解码；类别 source；类型 core-logic；符号 is_nemotron_35_draft_config）：新增 is_nemotron_35_draft_config 结构检查，用于保护 Nemotron 3.5 专属布局逻辑。
- python/sglang/srt/layers/quantization/modelopt_quant.py（模块 量化层；类别 source；类型 data-contract）：补充 ModelOpt W4A16 NVFP4 量化路径，Draft checkpoint 可被正确加载。
- test/registered/unit/test_model_overrides.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestDSparkCheckpointConfig,

test_sample_from_anchor_is_read_from_checkpoint_config,test_nemotron_h_speculation_uses_arch_specific_attention_on_blackwell,test_nemotron_h_sm100_speculative_draft_backend_matrix）：覆盖新增的 Blackwell注意力后端解析矩阵、DSpark 配置读取以及 ServerArgs 白名单变化。

- test/registered/models_e2e/test_nvidia_nemotron_3_nano.py（模块 集成测试；类别 test；类型 test-coverage；符号 TestNvidiaNemotron3Nano30BFP8, _Nemotron35LightningServer, setUpClass, tearDownClass）：注册 E2E 从 Nemotron Nano 替换为 Nemotron 3.5 Lightning NVFP4 的 Normal/DFlash/DSpark 三用例，作为外部 draft 路径的主要验证。

关键符号：_logical_linear_weight_shape, Nemotron35VanillaMarkov, build_nemotron_35_markov_head, get_dspark_sample_from_anchor, select_draft_hidden_without_anchor, set_dflash_layers_to_capture, is_nemotron_35_draft_config, _resolve_dflash_embedding_module

关键源码片段

python/sglang/srt/models/dflash.py

DFlash 草稿模型核心改造：支持 Nemotron 3.5 自带 embedding/LM head、packed 投影权重与 residual 层捕获，新增 _logical_linear_weight_shape 处理量化权重逻辑形状。

```
def _logical_linear_weight_shape(
    param: torch.Tensor,
    loaded_weight: torch.Tensor,
    *,
    output_features: int,
) -> Tuple[int, ...]:
    """返回 checkpoint 中线性层权重在逻辑元素上的形状。"""
    loaded_shape = tuple(loaded_weight.shape)
    pack_factor = getattr(param, "pack_factor", None)
    # 普通权重直接使用加载形状；只有 packed 参数需要展开逻辑形状。
    if pack_factor is None or loaded_shape != tuple(param.shape):
        return loaded_shape

    logical_numel = int(loaded_weight.numel() * pack_factor)
    # 按输出特征数推断逻辑二维形状，否则退回一维。
    if logical_numel % output_features == 0:
        return (output_features, logical_numel // output_features)
    return (logical_numel,)
```

python/sglang/srt/models/dspark.py

DSpark 草稿模型核心改造：新增可量化的 Nemotron35VanillaMarkov 与 build_nemotron_35_markov_head，并让 sample_from_anchor 从 draft config 派生。

```
class Nemotron35VanillaMarkov(VanillaMarkov):
    """仅用于 Nemotron 3.5 DSpark 的 checkpoint 量化 Markov head。"""

    def __init__(
```

```
        self,
        *,
        vocab_size: int,
        markov_rank: int,
        quant_config,
        prefix: str,
    ) -> None:
        nn.Module.__init__(self)
        self.vocab_size = int(vocab_size)
        self.markov_rank = int(markov_rank)
        if self.markov_rank <= 0:
            raise ValueError(
                "Nemotron35VanillaMarkov requires markov_rank > 0, "
                f"got {self.markov_rank}."
            )
        # 将低秩映射拆成 Embedding 与可量化 Linear，加载官方 W4A16 权重。
        self.markov_w1 = nn.Embedding(self.vocab_size, self.markov_rank)
        self.markov_w2 = ReplicatedLinear(
            self.markov_rank,
            self.vocab_size,
            bias=False,
            quant_config=quant_config,
            prefix=f"{prefix}.markov_w2" if prefix else "markov_w2",
        )

    def project_bias(self, latent_states: torch.Tensor) -> torch.Tensor:
        # 与通用 VanillaMarkov 的 bias 投影保持一致，但走 quantized linear。
        bias, _ = self.markov_w2(latent_states)
        return bias


def build_nemotron_35_markov_head(config, quant_config, prefix: str) -> nn.Module:
    markov_head_type = str(getattr(config, "markov_head_type", "vanilla")).lower()
    if markov_head_type != "vanilla":
        raise ValueError(
            "Nemotron 3.5 DSpark requires markov_head_type='vanilla', "
            f"got {markov_head_type!r}."
        )
    markov_prefix = f"{prefix}.markov_head" if prefix else "markov_head"
    return Nemotron35VanillaMarkov(
        vocab_size=int(config.vocab_size),
        markov_rank=int(config.markov_rank),
        quant_config=quant_config,
        prefix=markov_prefix,
    )
```

python/sglang/srt/speculative/dspark_components/dspark_draft.py

DSpark 采样路径核心改造：新增 select_draft_hidden_without_anchor 移除 query anchor，并按 sample_from_anchor 条件调整 block shape 与 embedding 来源，避免 legacy 路径 contiguous() 开销。

```
def select_draft_hidden_without_anchor(
    hidden_states: torch.Tensor,
    *,
    bs: int,
    gamma: int,
) -> tuple[torch.Tensor, torch.Tensor]:
    """从 draft 前向结果中剔除 query anchor 行，返回 (draft_hidden, draft_hidden_3d)。"""
    query_token_num = gamma + 1
    expected_rows = bs * query_token_num
    # 前置校验，避免后续 view 失败时难以定位问题。
    if hidden_states.shape[0] != expected_rows:
        raise RuntimeError(
            f"DSpark draft returned {hidden_states.shape[0]} hidden rows, "
            f"expected {expected_rows}."
        )
    hidden_by_query = hidden_states.view(bs, query_token_num, *hidden_states.shape[1:])
    selected = hidden_by_query[:, 1:].contiguous()
    return (
        selected.view(bs * gamma, *hidden_states.shape[1:]),
        selected.view(bs, gamma, -1),
    )
```

评论区精华

Review 核心围绕 " 如何让 Nemotron 3.5 专属行为更少侵入通用路径 " 展开：

- 作者自评指出不要新增 ServerArgs 字段，建议提供独立函数 get_dspark_sample_from_anchor(draft_hf_config)，每次从模型配置读取——最终移除该字段。
- 进一步指出 sample_from_anchor 不应按模型类型（is_nemotron_35_draft_config）推断，而应直接读 DSpark checkpoint 的 config.json 顶层字段；最终实现以 False 为 checkpoint 显式值、缺失时默认 True 的兼容方案。
- 在 dspark_draft.py 中，作者自评强调 no-anchor 路径才会调用选择函数，避免在 legacy anchor 路径引入 contiguous() 延迟，并要求将外部 helper 命名为 select_draft_hidden_without_anchor。
- 测试类改动也经历自评修正：E2E 基础模型从 BF16 换成生产 NVFP4 checkpoint，est_time 调整为 500 秒。
- E2E 基础模型改用 NVFP4 checkpoint (testing): 已切换为 nvidia/NVIDIA-Nemotron-3.5-Lightning-30B-A3B-NVFP4。
- 从 ServerArgs 移除 speculative_dspark_sample_from_anchor (design): 已移除字段，DSpark 模型和 worker 从已加载的 draft config 派生行为。

- sample_from_anchor 应直接读 checkpoint 配置而非按模型类型推断 (design): get_dspark_sample_from_anchor 直接读顶层 sample_from_anchor 字段，缺失时默认 True 以保持向后兼容。
- no-anchor 路径避免无谓 contiguous 延迟 (performance): 选择现在有条件下调用，legacy anchor 路径复用原始张量，不触发 contiguous。
- 外部 helper 命名 select_draft_hidden_without_anchor (design): 已重命名。
- Nemotron 特定 Markov head 选择也应从配置派生 (design): 已改为从 draft-config 派生的同一条件选择 build_nemotron_35_markov_head。
- E2E est_time 调整到约 500 秒 (testing): 已更新 register_cuda_ci(est_time=500)。

风险与影响

- 风险:
  1. 核心推测解码路径变更：dspark_draft.py 的 propose/_run_forward 改动了 draft block 的 shape 语义（query_token_num = gamma + 1），虽然 Nemotron 3.5 分支有结构检查保护，但任何 shape 推导错误都可能影响所有 DSpark 用户。
  2. 量化权重加载风险：dflash.py 的 _logical_linear_weight_shape 依赖 pack_factor 与 numel 推断逻辑形状，如果推断分支出错会静默加载错位权重，需依赖 E2E 精度兜底。
  3. Blackwell 后端默认变更：Nemotron-H overrides 在 SM100 上将默认 attention 后端从 flashinfer 改为 trtllm_mha，可能影响其他使用 Nemotron-H override 的配置；单元测试已覆盖显式用户选择不被覆盖的场景，但仍属行为变化。
  4. CI 耗时与依赖：新增 E2E 用例 est_time=500、4-gpu-b200，且依赖 HuggingFace 权重下载，存在网络不稳定性风险。
  5. 兼容性默认值：get_dspark_sample_from_anchor 在 checkpoint 缺失字段时默认 True，保持旧行为，但可能掩盖配置错误。 - 影响：对用户，B200/GB300 用户可以以生产 NVFP4 权重运行 Nemotron 3.5 Lightning 的 Normal/DFlash/DSpark 推测解码，GSM8K 精度保持 95% 以上，DFlash/DSpark 平均接受长度健康（DFlash 约 4.2/6，DSpark 约 3.7/4）。对系统，DFlash/DSpark 通用路径新增 Nemotron 3.5 分支，但通过结构检查保证非 Nemotron 行为不变；新增 ModelOpt W4A16 NVFP4 加载路径，扩大了量化支持面。对团队，需要维护新增的 draft 布局与 Blackwell 后端解析逻辑，并承担更长 CI 测试的排队成本。 - 风险标记：核心推测解码路径变更，量化权重加载改动，Blackwell 后端默认变更，新增长耗时 E2E 测试，checkpoint 配置兼容默认值

关联脉络

- PR #33554 Superseded Nemotron 3.5 speculative decoding approach: PR body 明确说明本 PR 取代 #33554 的实现探索路径，保留其中有用的 ModelOpt 与 Nemotron 模型改动。