PR #23837 完整报告

sgl-project/sglang

Add Ling_2_6

合并时间：2026-05-27 14:57

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23837

执行摘要

- 一句话：添加 Ling-2.6 百灵模型支持及推理优化
- 推荐动作：建议重点关注 bailing_moe_linear.py 中的异步 CUDA stream 设计，该模式可推广至其他 MoE 模型；handle_max_mamba_cache 的联合求解公式也值得参考。但需督促补充单元测试，并确认 ispobock 的 double check 已解决。

功能与动机

根据 PR body，此 PR 用于支持百灵模型 Ling-2.6（https://huggingface.co/collections/inclusionAI/ling-26），使用户能够在 sglang 中部署和推理这些新模型。

实现拆解

1. 添加 BailingMoE 模型支持（bailing_moe_linear.py）：修改 BailingMoE 和 BailingMoELinearDecoderLayer 类，接受 alt_stream 参数，在 forward 中使用异步 CUDA stream 并行执行 shared experts 和 routed experts；同时注册新模型名称。
2. 优化 Mamba 缓存内存管理（model_runner_kv_cache_mixin.py）：重构 handle_max_mamba_cache 方法，在 speculative decoding 场景下更精确地计算 intermediate mamba state 内存，基于 max_mamba_cache_size 或 max_running_requests 进行 capped 计算，并引入联合求解公式避免过度分配。
3. 修复 Lightning Attention 后端（lightning_backend.py）：移除不必要的 .cpu() 同步（由 review 指出并修复），添加 mamba extra buffer 的 track mask 处理，避免 GPU-CPU 同步开销；清理未使用的 q_rope/k_rope 变量。
4. 新增 H20 GPU Triton MoE 配置：添加多个 JSON 文件（如 E=256,N=512,device_name=NVIDIA_H20-3e_down.json），为 H20 提供调优后的 MoE kernel 参数。
5. 其他适配：修改 hybrid_linear_attn_backend.py 传递 alt_stream；调整 forward_batch_deepseek_mha_mixin.py 的梯度 checkpoint；为 Bailing 模型设置 mamba_extra_buffer_no_aligned 标志（随后在 review 后移除）；修改 eagle_worker_v2.py 适配 BailingMoE 模型注册。
6. 测试缺失：PR 按 checklist 要求应添加单元测试，但可见文件列表无测试文件。

关键文件：

- python/sglang/srt/models/bailing_moe_linear.py（模块 MoE 层；类别 source；类型 core-logic；符号 BailingMoE, BailingMoELinearDecoderLayer）：核心模型文件，实现 BailingMoE 和 BailingMoELinearDecoderLayer，添加异步 CUDA stream 并行专家计算，是本次功能的主要载体。
- python/sglang/srt/model_executor/model_runner_kv_cache_mixin.py（模块 缓存管理；类别 source；类型 data-contract；符号 handle_max_mamba_cache）：重构 Mamba 缓存内存分配，更精确地计算 speculative decoding 中间状态所需内存，避免浪费或 OOM。
- python/sglang/srt/layers/attention/linear/lightning_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 LightningAttentionBackend, forward_extend, forward_decode）：修复 Lightning Attention 前端性能问题，移除不必要的 GPU-CPU 同步，并添加 mamba extra buffer 的 track mask 处理。
- python/sglang/srt/layers/moe/moe_runner/triton_utils/configs/triton_3_5_1/E=256,N=512,device_name=NVIDIA_H20-3e_down.json（模块 调优配置；类别 config；类型 configuration）：为 H20 GPU 新增 MoE kernel 调优配置，确保在 Ling 2.6 模型（E=256, N=512）上获得最佳性能。

关键符号：BailingMoE.forward, BailingMoELinearDecoderLayer.forward, handle_max_mamba_cache, LightningAttentionBackend.forward_extend

关键源码片段

python/sglang/srt/models/bailing_moe_linear.py

核心模型文件，实现 BailingMoE 和 BailingMoELinearDecoderLayer，添加异步 CUDA stream 并行专家计算，是本次功能的主要载体。

```
def forward(self, hidden_states) -> torch.Tensor:
    num_tokens, hidden_size = hidden_states.shape
    hidden_states = hidden_states.view(-1, hidden_size)

    # CUDA Graph capture 模式下启用异步流并行
    if (
        self.alt_stream is not None
        and self.num_shared_experts > 0
        and hidden_states.shape[0] > 0
        and get_is_capture_mode()
    ):
        with torch.no_grad():
            current_stream = torch.cuda.current_stream()
            self.alt_stream.wait_stream(current_stream)
            # 主流：执行 shared experts（计算量较小）
            shared_output = self.shared_experts(hidden_states)
            # 另一流：执行 gate、topk 和 routed experts（计算量较大）
            with torch.cuda.stream(self.alt_stream):
                router_logits = self.gate(hidden_states)
                topk_output = self.topk(hidden_states, router_logits)
                final_hidden_states = self.experts(hidden_states, topk_output)
            current_stream.wait_stream(self.alt_stream)
```

```
            final_hidden_states = final_hidden_states + shared_output
    else:
        # 回退到顺序执行
        if self.num_shared_experts > 0:
            shared_output = self.shared_experts(hidden_states)
        router_logits = self.gate(hidden_states)
        topk_output = self.topk(hidden_states, router_logits)
        final_hidden_states = self.experts(hidden_states, topk_output)
        if self.num_shared_experts > 0:
            final_hidden_states = final_hidden_states + shared_output

    # 跨张量并行 all-reduce（省略具体实现）
    if self.tp_size > 1 and not should_skip_post_experts_all_reduce(...):
        ...
    return final_hidden_states
```

python/sglang/srt/model_executor/model_runner_kv_cache_mixin.py

重构 Mamba 缓存内存分配，更精确地计算 speculative decoding 中间状态所需内存，避免浪费或 OOM。

```
def handle_max_mamba_cache(self: ModelRunner, total_rest_memory):
    config = self.mambaish_config
    server_args = self.server_args
    assert config is not None

    has_spec_dec = not self.spec_algorithm.is_none()
    if has_spec_dec:
        assert server_args.speculative_num_draft_tokens is not None
        assert server_args.max_running_requests is not None

    if server_args.max_mamba_cache_size is not None:
        # 用户明确设置 max_mamba_cache_size
        server_args.max_mamba_cache_size //= (server_args.dp_size if server_args.enable_dp_
        attention else 1)
        if has_spec_dec:
            # 用 capped_reqs 限制中间状态数量，避免超出
            ratio = self._calculate_mamba_ratio()
            capped_reqs = min(
                server_args.max_running_requests // (self.dp_size if server_args.enable_dp_
                attention else 1),
                server_args.max_mamba_cache_size // ratio,
            )
            intermediate_size = (
                config.mamba2_cache_params.mamba_cache_per_req * capped_reqs * server_args.
                speculative_num_draft_tokens
            )
            total_rest_memory -= intermediate_size / (1 << 30)
    elif server_args.disable_radix_cache and server_args.max_running_requests is not None:
        ... # 类似 capped 逻辑
```

```
    else:
        # Ratio 分支：联合求解 main + intermediate 状态
        per_req = config.mamba2_cache_params.mamba_cache_per_req
        mamba_budget = total_rest_memory * server_args.mamba_full_memory_ratio / (1 + server_
        args.mamba_full_memory_ratio)
        mamba_budget_bytes = mamba_budget * (1 << 30)
        if has_spec_dec:
            ratio = self._calculate_mamba_ratio()
            D = server_args.speculative_num_draft_tokens
            # 联立方程：main + intermediate = mamba_budget_bytes
            server_args.max_mamba_cache_size = int(mamba_budget_bytes // (per_req * (1 + D /
            ratio)))
            capped_reqs = min(
                server_args.max_running_requests // (self.dp_size if server_args.enable_dp_
                attention else 1),
                server_args.max_mamba_cache_size // ratio,
            )
            intermediate_size = per_req * capped_reqs * D
            total_rest_memory -= intermediate_size / (1 << 30)
        else:
            server_args.max_mamba_cache_size = int(mamba_budget_bytes // per_req)
    ...
```

python/sglang/srt/layers/attention/linear/lightning_backend.py

修复 Lightning Attention 前端性能问题，移除不必要的 GPU-CPU 同步，并添加 mamba extra buffer 的 track mask 处理。

```
  def forward_extend(self, q, k, v, layer, forward_batch, save_kv_cache=True, **kwargs):
      layer_id = layer.layer_id if layer else kwargs["layer_id"]
      metadata = self.forward_metadata
      if self.kv_cache_dtype_str != "auto" and layer.k_scale is not None:
          q = q.to(self.kv_cache_dtype)
      cache_indices = self.forward_metadata.mamba_cache_indices
      mamba_cache_params = self.req_to_token_pool.mamba2_layer_cache(layer_id)
      ssm_states = mamba_cache_params.temporal
      if self.linear_backend == "minimax":
          o = self._prefill_and_mix_infer(...)
      elif self.linear_backend == "seg_la":
          ... # seg_la forward
      else:
          raise ValueError(...)

      # 保存 mamba cache 到 extra buffer（仅在非 target verify 时）
      if not forward_batch.forward_mode.is_target_verify() and forward_batch.mamba_track_mask
      is not None:
          # 注意：此处已移除 .cpu() 同步，直接在设备端操作
          mamba_track_mask = forward_batch.mamba_track_mask
          mamba_track_indices = forward_batch.mamba_track_indices
          dst_masked = mamba_track_indices[mamba_track_mask]
```

```
        src_masked = metadata.mamba_cache_indices[mamba_track_mask]
        ssm_states[dst_masked] = ssm_states[src_masked]
    return o.view(-1, layer.tp_q_head_num * layer.v_head_dim)
```

评论区精华

-.cpu() 同步问题：gemini-code-assist[bot] 指出 lightning_backend.py 中 mamba_track_mask.cpu() 会引发 GPU-CPU 同步，严重性能瓶颈；ispobock 也质疑引入 CPU 同步的原因。ant-yy 随后移除，并确认“无需 .cpu() 调用”。

- 参数必要性：yizhang2077 质疑 --mamba-extra-buffer-no-aligned 参数的必要性，yuan-luo 也认为应由模型自动推断而非用户设置。ant-yy 解释 Ling 2.6 的 Lightning Attention 不支持 aligned branching，但最终接受了反馈并移除了该参数。
- 缓存分配修改：ispobock 要求对 handle_max_mamba_cache 的修改进行 double check，ant-yy ping yizhang2077 但未获得明确回复。
  - lightning_backend.py 中的 .cpu() 同步问题 (performance): ant-yy 移除了 .cpu() 调用，改为直接在设备端操作，避免了同步开销。
  - model_runner_kv_cache_mixin.py 修改需 double check (correctness): ant-yy 回复了 ping，但 yizhang2077 未发表最终意见；PR 此后合并状态表明修改被接受。
  - --mamba-extra-buffer-no-aligned 参数必要性 (design): ant-yy 解释该模型不支持 aligned branching，但随后移除该参数，采纳了建议。

风险与影响

- 风险：
  - 核心路径变更：bailing_moe_linear.py 改动较大，新增异步流逻辑，若 alt_stream 未正确设置或同步错误，可能导致 CUDA 数据竞争或崩溃。
  - 缺少测试覆盖：未发现对应单元测试文件，回归风险较高，特别是动态流并行和内存分配新逻辑。
  - 配置新增：JSON 配置文件仅针对 H20 GPU，不影响其他硬件，但若名称冲突可能导致误用。
  - 参数移除：--mamba-extra-buffer-no-aligned 参数在 review 后移除，但相关代码（如 mamba_track_mask 路径）仍在，需要确保未遗留依赖。
  - 兼容性：修改了 eagle_worker_v2.py，可能影响其他 speculative decoding 模型。
- 影响：
  - 用户：可部署 Ling-2.6 系列模型（如 Ling-2.6-1T、Ling-2.6-Flash），享受异步流带来的推理加速；H20 用户获得调优的 MoE kernel 配置。
  - 系统：增加了新模型注册和调度逻辑，对 scheduler 和内存管理有细微改动（如 forward_batch_deepseek_mha_mixin.py），可能影响其他 hybrid 模型。
  - 团队：后续需要维护新模型配置，并响应可能出现的回归问题。
  - 风险标记：缺少测试覆盖，核心路径变更，review 遗留疑问

关联脉络

- PR #26397 Reland "[perf][spec decoding] Skip full-vocab softmax in EAGLE draft when topk == 1 (#26235)": 同样修改了 eagle_worker_v2.py，涉及 speculative decoding worker 适配，与本 PR 有文件重叠。