PR #27471 完整报告

sgl-project/sglang

add dflash gemma4 support

合并时间：2026-06-18 07:39

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27471

执行摘要

- 一句话：为 Gemma4 添加 DFlash 推测解码支持
- 推荐动作：值得精读该 PR，尤其是 dflash_worker_v2.py 中的采样逻辑改动：如何优雅地处理不同类型 LM head（有 / 无 shard_indices），以及测试脚本如何通过 /server_info 校验 spec 配置和 acceptance length。建议后续类似功能的实现可以遵循这种模式。

功能与动机

从 PR #23000 分拆而来，为 Gemma 4 模型系列提供 DFlash 推测解码能力。详见 PR body。关联的依赖 PR：#27469, #27737。

实现拆解

实现拆解：

1. 添加层捕获接口：在 python/sglang/srt/models/gemma4_causal.py 和 python/sglang/srt/models/gemma4_mm.py 中新增 set_dflash_layers_to_capture(self, layer_ids) 方法。该方法校验 layer_ids 非空，设置 capture_aux_hidden_states = True，并将层索引偏移 +1 后赋值给 model.layers_to_capture。这样 DFlash worker 在前期只需调用模型该方法即可注册待捕获的中间层。
2. 增强贪婪采样鲁棒性：在 python/sglang/srt/speculative/dflash_worker_v2.py 的 _greedy_sample_from_vocab_parallel_head 方法中，重构了缺少 shard_indices 属性的 LM head（如 Gemma4 视觉模型的 tied LM head）的采样路径。原先会直接抛出异常，现在先检查 shard_indices 是否存在；若不存在，则对每个 chunk 执行简单的 matmul(hs, weight.T) + argmax，避免了不必要的 TP 同步。同时将 _cast_hs 辅助函数提取到外层，减少重复。
3. 新增集成测试：新增 test/registered/spec/test_gemma4_dflash_31b_extra.py，注册 CI stage "extra-a" 使用 2-GPU-large runner。测试启动 Gemma4 31B 目标模型和 DFlash draft 模型（z-lab/gemma-4-31B-it-DFlash），验证服务器配置（speculative_algorithm, draft_attention_backend 等）正确，运行 GSM8K 评测并断言 accuracy 和 average speculative acceptance length 均达到阈值（0.75 和 5.4）。

关键文件：

- test/registered/spec/test_gemma4_dflash_31b_extra.py（模块 测试套件；类别 test；类型 test-coverage；符号 get_server_info, get_avg_spec_accept_length,

TestGemma4DFlash31B, _common_server_args）：新增 31B DFlash 端到端测试，验证 GSM8K 准确率和 spec accept length 指标

- python/sglang/srt/speculative/dflash_worker_v2.py（模块 推测解码；类别 source；类型 core-logic；符号 _cast_hs）：核心 DFlash worker 采样逻辑修改，支持无 shard_indices 的 LM head
- python/sglang/srt/models/gemma4_causal.py（模块 文本模型；类别 source；类型 data-contract；符号 set_dflash_layers_to_capture）：新增 set_dflash_layers_to_capture 方法，配置 DFlash 辅助层捕获
- python/sglang/srt/models/gemma4_mm.py（模块 多模态模型；类别 source；类型 data-contract；符号 set_dflash_layers_to_capture）：新增 set_dflash_layers_to_capture 方法，为多模态模型启用 DFlash 层捕获

关键符号：_greedy_sample_from_vocab_parallel_head, set_dflash_layers_to_capture

关键源码片段

test/registered/spec/test_gemma4_dflash_31b_extra.py

新增 31B DFlash 端到端测试，验证 GSM8K 准确率和 spec accept length 指标

```
class TestGemma4DFlash31B(CustomTestCase):
    base_url = DEFAULT_URL_FOR_TEST

    @classmethod
    def _server_args(cls) -> list[str]:
        # 构造 DFlash 服务器所需参数
        return [
            "--speculative-algorithm", "DFLASH",
            "--speculative-draft-model-path", DRAFT_PATH,
            "--speculative-num-draft-tokens", str(SPECULATIVE_NUM_DRAFT_TOKENS),
            "--speculative-draft-attention-backend", DRAFT_ATTENTION_BACKEND,
        ] + cls._common_server_args()

    @classmethod
    def _gsm8k_args(cls) -> SimpleNamespace:
        return SimpleNamespace(
            base_url=cls.base_url,
            model=TARGET_PATH,
            eval_name="gsm8k",
            api="completion",
            max_tokens=512,
            num_examples=GSM8K_NUM_EXAMPLES,
            num_threads=GSM8K_NUM_THREADS,
            num_shots=5,
        )

    def test_gsm8k_dflash(self) -> None:
        process = None
```

```
        try:
            # 启动服务器（目标模型 + DFlash draft 模型）
            process = popen_launch_server(
                TARGET_PATH,
                self.base_url,
                timeout=SERVER_LAUNCH_TIMEOUT,
                other_args=self._server_args(),
            )
            # 验证服务器配置
            server_info = get_server_info(self.base_url)
            self.assertEqual(server_info.get("speculative_algorithm"), "DFLASH")
            # 运行 GSM8K 评测
            metrics = run_eval(self._gsm8k_args())
            dflash_score = float(metrics["score"])
            avg_accept = get_avg_spec_accept_length(self.base_url)
        finally:
            if process is not None:
                self._stop_process(process)
        # 断言 score 和 acceptance length 达到阈值
        self.assertGreaterEqual(dflash_score, GSM8K_SCORE_THRESHOLD)
        self.assertGreaterEqual(avg_accept, ACCEPT_LENGTH_THRESHOLD)
```

python/sglang/srt/speculative/dflash_worker_v2.py

核心 DFlash worker 采样逻辑修改，支持无 shard_indices 的 LM head

```
def _greedy_sample_from_vocab_parallel_head(
    self,
    *,
    hidden_states: torch.Tensor,
    lm_head,
    chunk_size: int = 256,
) -> torch.Tensor:
    # 空输入处理
    if hidden_states.numel() == 0:
        return torch.empty((0,), dtype=torch.long, device=hidden_states.device)

    weight = lm_head.weight # [local_vocab_padded, hidden]
    weight_dtype = weight.dtype
    num_tokens = int(hidden_states.shape[0])
    out_tokens = torch.empty(
        (num_tokens,), dtype=torch.long, device=hidden_states.device
    )

    def _cast_hs(x: torch.Tensor) -> torch.Tensor:
        # 若 hidden_states 精度与 weight 不一致则转换
        return x if x.dtype == weight_dtype else x.to(weight_dtype)

    # 当 LM head 没有 shard_indices（非 TP 或 tied head）时，
    # 使用简单 matmul 采样，避免不必要的 TP 同步
```

```
    if not hasattr(lm_head, "shard_indices"):
        for start in range(0, num_tokens, int(chunk_size)):
            end = min(num_tokens, start + int(chunk_size))
            hs = _cast_hs(hidden_states[start:end])
            logits = torch.matmul(hs, weight.T)
            out_tokens[start:end] = torch.argmax(logits, dim=-1).to(torch.long)
        return out_tokens

    # 以下为原有 TP 同步路径（有 shard_indices 时执行）
    shard = lm_head.shard_indices
    tp_group = get_tp_group()
    tp_size = int(tp_group.world_size)
    # ... ( 后续 TP 同步逻辑保持不变 )
```

python/sglang/srt/models/gemma4_causal.py

新增 set_dflash_layers_to_capture 方法，配置 DFlash 辅助层捕获

```
def set_dflash_layers_to_capture(self, layer_ids: list[int]):
    # DFlash 要求显式指定要捕获的层 IDs
    if layer_ids is None:
        raise ValueError(
            "DFLASH requires explicit layer_ids for aux hidden capture."
        )
    # 开启辅助 hidden states 捕获，forward 时会返回额外信息
    self.capture_aux_hidden_states = True
    # 注意：layer_ids 需要偏移 +1，因为模型内部层索引可能从 1 开始
    self.model.layers_to_capture = [val + 1 for val in layer_ids]
```

评论区精华

Review 中主要讨论如下：

- extra-test 需求：kphant-sgl 要求添加额外的集成测试（extra-test），作者随后补充了 31B DFlash 测试。
- 依赖与冲突：作者指出该测试依赖 #27469 和 #27737 两个 PR 的配合。分支在合并过程中多次与 main 冲突，经 kpham-sgl 协助解决并清理了已废弃的 SGLANG_ENABLE_SPEC_V2 环境变量覆盖。
  - 添加额外测试（extra test） (testing): 已添加 test_gemma4_dflash_31b_extra.py 作为 extra-a 测试。

风险与影响

- 风险：
  - 变更范围集中：仅影响 Gemma4 模型，且仅在 DFlash 推测解码启用时激活。
  - 新采样路径正确性：dflash_worker_v2 新增的无 shard_indices 路径虽简单，但需要确认 Gemma4 视觉模型 tiled head 的行为符合预期；当前视觉模型非 TP 场景可以使用此路径。

    - 测试覆盖：仅覆盖 31B 模型，26B-A4B 模型未测试；但 31B 是主要验证场景。
    - 外部依赖：若 #27469 和 #27737 未合并，测试可能失败；但最后测试通过表明依赖已就绪。
    - 性能风险：新增的 fallback 路径使用简单的 matmul 循环，对于小 batch 没有明显开销。
- 影响：
    - 用户影响：Gemma4 用户现在可选择使用 DFlash 加速推理；需指定 speculative-algorithm DFLASH 和对应的 draft 模型。
    - 系统影响：Gemma4 模型注册无需额外改动，但模型类增加了新方法，可能被未来 DFlash 扩展复用。
    - 团队影响：DFlash 支持扩展到 Gemma4 系列，为后续添加其他模型提供了参考模式。
    - 风险标记：新采样路径仅测试 31B，依赖外部 PR，仅 DFlash 路径变更

关联脉络

- PR #23000 dflash support (original PR): 该 PR 从 #23000 分拆。
- PR #27469 unknown (dependency): 依赖该 PR 才能通过测试。
- PR #27737 unknown (dependency): 依赖该 PR 才能通过测试。