PR #30741 完整报告

sgl-project/sglang

Prewarm DSV4 MHC post kernel at model load

合并时间：2026-08-04 14:41

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30741

执行摘要

- 一句话：DSV4 MHC post 内核预编译前移，消除首请求 6.9s 延迟
- 推荐动作：值得精读，尤其是对 DeepSeek 模型性能和冷启动优化感兴趣的工程师。这是一个小型但精准的性能修复：通过复用现有 prewarm 基础设施，只改动一个文件就将 MHC post 的编译移出 serving 路径。关注点：one-token 模板如何覆盖动态形状、环境开关的双重门控、以及 barrier 前的 cuda.synchronize() 如何保证 rank 对齐。可作为后续处理其他动态形状 JIT kernel 的参考实现。

功能与动机

PR body 明确指出根因：SGLANG_DSV4_MHC_PREWARM 只编译了 MHC pre 变体，遗漏了独立的动态形状 MHC post kernel，导致冷进程在首个 serving 请求时 JIT 编译，引入约 6.9 秒主机侧延迟并增大多 rank 到达偏斜（multi-rank arrival skew）。将编译移出 serving 路径可消除这一冷启动惩罚，同时让各 rank 在 barrier 处对齐，减少 DeepGEMM 负载不均。

实现拆解

1. 重命名并扩展预编译入口：在 python/sglang/srt/models/deepseek_v4.py 中，将 _prewarm_mhc_pre_kernels 重命名为 _prewarm_mhc_kernels，docstring 和日志文案从“MHC prenorm prewarm”更新为“MHC prewarm”，语义上覆盖 pre 与 post 两个 kernel。
2. 共用 residual 模板：原来 inline 在 prewarm_mhc_pre 调用中的 torch.zeros((1, layer.hc_mult, layer.hidden_size)) 被提取为局部变量 residual，供 pre 与 post 两个调用复用，保证模板 dtype/device 一致。
3. 新增 mhc_post 预编译调用：从 sglang.kernels.ops.layernorm.mhc 导入 mhc_post，在 prewarm 后立即用 one-token 形状（(1, layer.hidden_size)）的 x 和零初始化的 post_layer_mix、comb_res_mix 调用它，触发动态形状 kernel 的 JIT 编译。
4. 保持同步与 barrier 顺序：两次编译调用后仍是 torch.cuda.synchronize()、torch.cuda.empty_cache()、get_tp_group().barrier() 的既有序列，确保编译突发不进入 serving 路径，且 rank 之间对齐。
5. 调用点更新：load_weights 末尾的 self._prewarm_mhc_pre_kernels() 改为 self._prewarm_mhc_kernels()。
6. 配套验证：提交消息提到 compileall 与 pre-commit 检查，以及 GB300 上 1/17/4096 token 执行与 PyTorch 参考对比（4096 token 最大绝对误差 0.03125，平均绝对误差

6e-8）。本 PR 未新增测试文件，依赖现有模型 e2e 测试（如 test_deepseek_v4_flash_fp4_b200.py）验证。

关键文件：

- python/sglang/srt/models/deepseek_v4.py（模块 模型加载；类别 source；类型 data-contract；符号 _prewarm_mhc_pre_kernels, _prewarm_mhc_kernels）：唯一变更文件，承载 DSV4 模型加载路径。将 MHC post kernel 预编译并入现有 prewarm 逻辑，重命名入口函数并保持 barrier 同步顺序，直接决定首请求延迟是否被消除。

关键符号：_prewarm_mhc_kernels, _prewarm_mhc_pre_kernels

关键源码片段

python/sglang/srt/models/deepseek_v4.py

唯一变更文件，承载 DSV4 模型加载路径。将 MHC post kernel 预编译并入现有 prewarm 逻辑，重命名入口函数并保持 barrier 同步顺序，直接决定首请求延迟是否被消除。

```
def _prewarm_mhc_kernels(self) -> None:
    """One-shot MHC JIT prewarm at load time, synced across ranks.

    Runs before any forward so the compile burst stays off the serving
    path; the barrier keeps ranks from proceeding while a peer is still
    compiling. The early returns below must stay rank-uniform.
    """
    if self._mhc_prewarmed_at_load:
        return
    self._mhc_prewarmed_at_load = True

    # NPU 或环境开关未开启时直接跳过，保证 rank 间行为一致
    if _is_npu or not (
        envs.SGLANG_DSV4_MHC_PREWARM.get()
        and envs.SGLANG_OPT_USE_TILELANG_MHC_PRE.get()
    ):
        return

    layer = next(
        (m for m in self.model.layers if isinstance(m, DeepseekV4DecoderLayer)),
        None,
    )
    if layer is None:
        return

    # 局部导入避免模块加载时引入 TileLang 依赖
    from sglang.kernels.ops.layernorm.mhc import mhc_post, prewarm_mhc_pre

    tic = time.perf_counter()

    # 统一模板：one-token residual 同时驱动 pre 与 post 两个动态形状 kernel 的编译
    residual = torch.zeros(
```

```
        (1, layer.hc_mult, layer.hidden_size),
        dtype=torch.bfloat16,
        device=layer.hc_attn_fn.device,
    )
    prewarm_mhc_pre(
        residual=residual,
        fn=layer.hc_attn_fn,
        hc_scale=layer.hc_attn_scale,
        hc_base=layer.hc_attn_base,
        rms_eps=layer.rms_norm_eps,
        hc_pre_eps=layer.hc_eps,
        hc_sinkhorn_eps=layer.hc_eps,
        hc_post_mult_value=_MHC_POST_MULT_VALUE,
        sinkhorn_repeat=layer.hc_sinkhorn_iters,
        n_splits=1,
        n_splits_pre=32,
        norm_weight=layer.input_layernorm.weight.data,
        norm_eps=layer.input_layernorm.variance_epsilon,
    )

    # 新增：用 one-token 模板编译此前遗漏的 mhc_post 动态形状 kernel
    mhc_post(
        x=residual.new_zeros((1, layer.hidden_size)),
        residual=residual,
        post_layer_mix=torch.zeros(
            (1, layer.hc_mult, 1),
            dtype=torch.float32,
            device=residual.device,
        ),
        comb_res_mix=torch.zeros(
            (1, layer.hc_mult, layer.hc_mult),
            dtype=torch.float32,
            device=residual.device,
        ),
    )

    # 编译完成后同步并清缓存，避免瞬时张量影响后续内存池大小估计
    torch.cuda.synchronize()
    compile_secs = time.perf_counter() - tic
    torch.cuda.empty_cache()
    # rank barrier 保证各卡编译完成后才继续加载，减少多 rank 到达偏斜
    get_tp_group().barrier()
    logger.info(
        "DeepSeek V4 MHC prewarm at load: compile %.1fs, rank sync +%.1fs",
        compile_secs,
        time.perf_counter() - tic - compile_secs,
    )
```

评论区精华

Reviewer YAMY1234 指出：该改动大概率不会影响 benchmark 结果（因为 benchmark 通常有 warmup 阶段），但能缓解真实 serving 场景（无 warmup）下的 DeepGEMM 负载不均问题，也有助于避免 warmup 阶段的潜在超时；加入 pre-warmup 阶段是无害的。Fridge003 无评论直接 APPROVED。无未解决的 review 评论。

- 对 benchmark 与真实 serving 的影响评估 (design): 一致认可改动价值，认为加入 pre-warmup 阶段无害，并明确其收益场景为无 warmup 的真实 serving 与 warmup 阶段的超时规避。

风险与影响

- 风险：
  1. 模型加载时间增加：mhc_post 的编译会使 load_weights 显著变长（虽然实际执行时省掉首请求延迟，但运维上可能把加载阶段视为启动时间的一部分）。
  2. 环境门控仍依赖 env 开关：SGLANG_DSV4_MHC_PREWARM 与 SGLANG_OPT_USE_TILELANG_MHC_PRE 必须同时开启才生效；若生产环境未设置则无效果，但不会引入回归。
  3. NPU 路径排除：_is_npu 判断保持提前返回，NPU 上不会执行，影响面限于 CUDA 平台。
  4. 内存瞬时开销：mhc_post 预编译会分配临时 tensor（residual、post_layer_mix、comb_res_mix），但代码在 init_memory_pool() 前执行并显式 empty_cache()，避免污染内存池。
  5. 数值一致性：prewarm 用 one-token 模板编译出的 kernel 需要覆盖 1/17/4096 等不同 token 数，验证显示误差在 BF16 容差内，但极端形状下仍建议关注。 - 影响：影响范围集中在 DeepSeek-V4（DSV4）模型加载路径：对部署在 GB300 等多卡环境、使用 TileLang MHC（多 head 压缩注意力）的 serving 服务，可消除首个请求约 6.9 秒的 JIT 编译延迟，并降低多 rank 到达偏斜，从而缓解 DeepGEMM 负载不均。由于改动被 SGLANG_DSV4_MHC_PREWARM 环境开关门控且仅影响 CUDA 路径，默认行为不变，对现有用户风险很低；对开启该开关的 DSV4 用户属于启动期与运行时之间的延迟转移，整体收益为正。团队影响：为后续同类动态形状 kernel 的 load-time prewarm 提供了可复用的模式。 - 风险标记：模型加载时间增加，依赖环境开关生效，仅覆盖 CUDA 路径，无新增单元测试

关联脉络

- PR #33098 Fix DSpark and DP/EP: 同为 DeepSeek 推理链路修复，涉及 speculative decoding 中 draft 元数据与并行策略，与本 PR 同属 DSV4 部署稳定性优化脉络。
- PR #33448 [DCP] Bound a request by the aggregate KV pool, not one rank's share: 同属 DeepSeek 相关路径（DSV4）的运行时健壮性修复，与本 PR 都关注多 rank 环境下的行为一致性。
- PR #33432 fix(mem_cache): state the MLA KV bound in the DCP index space: 同为 DeepSeek 模型相关的内存与索引空间修复，与本 PR 一样属于 DeepSeek 模型服务稳定性

的持续改进。