PR #34647 完整报告

sgl-project/sglang

[AMD] Enable 12-head MLA aiter fp8 Gluon decode (batched bh16bn128).

合并时间：2026-09-01 14:47

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34647

执行摘要

- 一句话：为 AMD GPU 启用 Kimi-K3 12 头 MLA FP8 Gluon 解码，显著提升长上下文吞吐。
- 推荐动作：该 PR 值得精读，尤其是以下设计决策： 1) 运行时探针与优雅降级模式：通过 MlaGluonCapability 数据类和分层检查（环境变量、导入、API 存在性），实现了对硬件和软件依赖的动态适应，是处理可选硬件加速功能的优秀范例。 2) 零填充拓扑的泛化：将原来的硬编码填充逻辑抽象为 head_pad_mode（"repeat"/"zero"/"none"），为未来支持其他低头数模型（如 h4, h8）的优化提供了扩展点。 3) 最小化变更边界：尽管涉及核心解码路径，但通过清晰的条件判断和回退机制，确保了对非目标配置的零影响。

功能与动机

Kimi-K3 在 TP8 下每个 GPU 暴露 12 个 MLA 头。aiter 快速解码 ASM 内核要求头数能被 16 整除，因此 FP8 服务不得不将 12 头零填充到 16，通过更慢的内核运行，在长上下文解码 CUDA 图场景下性能尤为不佳。aiter 已具备一个 Gluon 内核（bh16bn128 regime），支持原生 12 头掩码 MFMA 指令和 FP8 KV。本 PR 的目标是将 SGLang 的解码路径与此 Gluon 内核连接起来。参见 PR body: "The fast aiter persist decode ASM wants head counts divisible by 16 and has no native fp8 12-head path, so fp8 serving had to zero-pad to 16 and go through slower kernels — especially painful at long context under decode CUDA graph."

实现拆解

1. 新增运行时探针与包装器模块 (aiter_mla_gluon.py)：
   - 关键符号：MlaGluonCapability, probe_mla_gluon_capability, mla_gluon_decode, prefer_mla_gluon_decode。
   - 具体变更：创建新文件，提供运行时环境检测（Triton 版本、cga_layout API）、aiter mla_gluon 函数导入和 Gluon 解码调用封装。
   - 原因与影响：将硬件内核的可用性探查与业务逻辑解耦，为 aiter_backend.py 提供清晰的决策接口。若探针失败，解码路径自动降级到零填充模式，保证了向后兼容性。
2. 修改注意力后端 (aiter_backend.py) 以集成 Gluon 解码路由：
   - 关键符号：_zero_pad_mla_q_heads, _resolve_fp8_kv_scale_float, _resolve_mla_gluon_min_kv_seq_len, _forward_mla_decode。

- 具体变更： a. 在初始化阶段，为 num_head=12 设置 head_pad_mode="zero"，并在 FP8 KV 缓存时禁用传统的 PS 内核并调用 log_mla_gluon_capability()。 b. 重构 _mla_decode_fwd_with_head_pad 以支持 "repeat" 和 "zero" 两种填充模式。 c. 新增 _forward_mla_decode 方法，在解码入口点检查 prefer_mla_gluon_decode 条件（ head_pad_mode="zero"，num_head=12，fp8_dtype），并尝试调用 mla_gluon_decode。若调用成功则返回结果，否则回退到传统的 _mla_decode_fwd_with_head_pad。 d. 新增辅助函数 _resolve_fp8_kv_scale_float 和 _resolve_mla_gluon_min_kv_seq_len 以正确处理 FP8 缩放因子和 CUDA 图捕获时的序列长度。
- 原因与影响：这是功能接入的核心。将 Gluon 路径作为优先的快速路径，在核心解码逻辑中实现了基于运行时状态的动态路由。

3. 调整 MLA 前向方法 (forward_mla_rocm.py) 以兼容 Gluon 内核：
   - 关键符号：_fused_rope_cat_and_cache。
   - 具体变更：在 FP8 KV 缓存且使用 aiter 后端时，保持查询张量 Q 为 bf16（而非 FP8），以满足 Gluon bh16bn128 regime 的要求。同时改进了 _skip_rope_for_aiter_fused_mla 的注释以明确 NoPE 模型（如 Kimi-K3）的行为。
   - 原因与影响：这是一个关键的数据契约调整。Gluon 内核要求 FP8 KV 模式下的 Q 输入必须是 bf16，此变更确保了 prefill 阶段的 Q 数据类型正确，避免了运行时错误。
4. 添加环境变量开关 (environ.py)：
   - 具体变更：新增 SGLANG_AITER_MLA_GLUON = EnvBool(True) 环境变量，允许用户全局禁用 Gluon 解码路径，用于性能对比或紧急回退。
   - 原因与影响：提供了运维层面的控制权，是生产环境部署的重要安全阀。
5. 补充单元测试 (test_mla_gluon_h12_fp8.py)：
   - 关键符号：TestMlaGluonCapability, TestMlaGluonDecodeFallback。
   - 具体变更：新增纯 CPU 的单元测试，通过 mock 覆盖 aiter_mla_gluon 模块的探针逻辑（环境开关、导入、Triton API 检查）和 aiter_backend 的解码回退逻辑。
   - 原因与影响：确保在无 aiter/GPU 环境下，核心路由和降级逻辑的正确性，提高了代码的可维护性和 CI 覆盖率。

关键文件：

- python/sglang/srt/layers/attention/aiter_mla_gluon.py（模块 Gluon MLA 包装器；类别 source；类型 dependency-wiring；符号 MlaGluonCapability, missing_for_ready, _triton_version, _triton_cga_layout_ok）：新增的 Gluon MLA 解码探针与调用封装模块，是功能的核心引入点。
- python/sglang/srt/layers/attention/aiter_backend.py（模块 Aiter 注意力后端；类别 source；类型 dependency-wiring；符号 _zero_pad_mla_q_heads, _resolve_fp8_kv_scale_float, _resolve_mla_gluon_min_kv_seq_len, _forward_mla_decode）：修改后的注意力后端，集成了 Gluon 解码路由逻辑，是功能的主要接入点。
- test/registered/attention/test_mla_gluon_h12_fp8.py（模块 Gluon 解码测试；类别 test；类型 test-coverage；符号 TestMlaGluonCapability, setUp, tearDown,

test_env_disable_not_ready）：新增的单元测试，验证探针逻辑和解码回退机制，保障核心路由代码质量。

- python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla_rocm.py（模块 MLA ROCm 前向；类别 source；类型 data-contract）：修改 MLA 前向计算，在 FP8 KV 模式下为 Gluon 内核保持 Q 为 bf16，调整数据契约。
- python/sglang/srt/environ.py（模块 环境配置；类别 config；类型 configuration）：新增环境变量 SGLANG_AITER_MLA_GLUON，为功能提供全局开关。

关键符号：probe_mla_gluon_capability, log_mla_gluon_capability, mla_gluon_available, mla_gluon_decode, prefer_mla_gluon_decode, reset_mla_gluon_state_for_test, _forward_mla_decode, _resolve_fp8_kv_scale_float, _resolve_mla_gluon_min_kv_seq_len, _zero_pad_mla_q_heads

关键源码片段

python/sglang/srt/layers/attention/aiter_mla_gluon.py

新增的 Gluon MLA 解码探针与调用封装模块，是功能的核心引入点。

```
# Gluon 能力探针：检查环境变量、aiter 导入和 Triton cga_layout API
@dataclass(frozen=True)
class MlaGluonCapability:
    """Runtime probe of aiter/Triton Gluon prerequisites for h12 + FP8 decode."""
    enabled_by_env: bool
    import_ok: bool
    triton_version: str
    triton_cga_layout_ok: bool
    ready: bool
    summary: str

    def missing_for_ready(self) -> list[str]:
        missing = []
        if not self.enabled_by_env:
            missing.append("SGLANG_AITER_MLA_GLUON=0") # 环境变量禁用
        if not self.import_ok:
            missing.append("aiter.ops.triton.gluon.mla_gluon import") # aiter 未安装或版本旧
        if not self.triton_cga_layout_ok:
            missing.append(
                f"Triton Gluon cga_layout (have {self.triton_version or 'unknown'}, need >= 3.7)"
            ) # Triton 版本过低或 API 缺失
        return missing

def probe_mla_gluon_capability(*, force_refresh: bool = False) -> MlaGluonCapability:
    """缓存并返回当前环境对 Gluon MLA 解码的支持状态。"""
    global _capability_cache
    if _capability_cache is not None and not force_refresh:
        return _capability_cache
    enabled = _mla_gluon_enabled()
    triton_ver = _triton_version()
```

```
    import_ok = mla_gluon_available() if enabled else False
    cga_ok = _triton_cga_layout_ok()
    ready = enabled and import_ok and cga_ok
    # ... 构建 summary 和缓存 _capability_cache ...
    return _capability_cache
```

python/sglang/srt/layers/attention/aiter_backend.py

修改后的注意力后端，集成了 Gluon 解码路由逻辑，是功能的主要接入点。

```
def _forward_mla_decode(self, q, layer, forward_batch, k_descale):
    """核心解码路由：优先尝试 Gluon，失败则回退到传统路径。"""
    k_buffer = self.token_to_kv_pool.get_key_buffer(layer.layer_id)
    q_mla = q.view(-1, layer.tp_q_head_num, layer.qk_head_dim)
    max_q_len = self.forward_metadata.max_q_len or 1

    # 决策点：检查是否应启用 Gluon 解码
    if (
        prefer_mla_gluon_decode(
            head_pad_mode=getattr(self, "head_pad_mode", "none"),
            num_head=getattr(self, "num_head", layer.tp_q_head_num),
            kv_cache_dtype=self.kv_cache_dtype,
        )
        and max_q_len == 1 # 仅解码阶段
    ):
        kv_scale = self._resolve_fp8_kv_scale_float(layer, k_descale)
        min_kv_seq_len = self._resolve_mla_gluon_min_kv_seq_len(forward_batch)
        gluon_out = mla_gluon_decode( # 调用包装好的 Gluon 内核
            q=q_mla, k_buffer=k_buffer, layer=layer,
            kv_indices=..., kv_indptr=..., seq_lens=...,
            sm_scale=layer.scaling, kv_scale=kv_scale, min_kv_seq_len=min_kv_seq_len,
        )
        if gluon_out is not None: # Gluon 调用成功
            return gluon_out
    # 回退到传统的 mla_decode_fwd + 头填充路径
    return self._mla_decode_fwd_with_head_pad(q_mla, k_buffer.view(-1, 1, 1, layer.qk_head_
    dim), layer, ...)
```

评论区精华

PR 讨论相对简洁。审核者 HaiShaw 仅给出了 "LGTM" 的批准。PR 作者在收到评论（非审核评论）后，前往了 Issue #21302 补充了 aiter 依赖跟踪信息，表明项目有集中的依赖管理流程。代码本身展示了清晰的防御性编程风格和对硬件兼容性的周密考虑，这可能是获得快速批准的原因。

- aiter 依赖跟踪与集成 (other): 明确了外部硬件库依赖的集散点，便于项目层面统一管理升级和兼容性检查。

风险与影响

- 风险：
  1. 运行时依赖风险：功能严重依赖未在代码仓库中固定版本的外部 aiter（需要 PR #4480 和 #4555 合并后的版本）和 Triton（>= 3.7）。探针机制（ probe_mla_gluon_capability）可优雅降级，但用户可能需要手动升级容器镜像才能获得性能收益。
  2. CUDA 图兼容性：解码路径引入了对 min_kv_seq_len 的新处理逻辑（ _resolve_mla_gluon_min_kv_seq_len）。在 CUDA 图捕获期间，seq_lens 为空，函数返回 max_context_len。需要确保此逻辑与 CUDA 图的执行模型完全一致，否则可能导致 KV 分割预算错误。关联 PR #4555 旨在修复此类问题，但两者需共同测试。
  3. 模型特定路径：此优化严格限定于 num_head=12 且 kv_cache_dtype=fp8_e4m3 的情况（Kimi-K3 TP8）。对于其他模型或配置，路径不受影响，但增加了后端代码的复杂性。 - 影响：用户影响：直接影响在 AMD MI355X (gfx950) GPU 上使用 --kv-cache-dtype fp8_e4m3 和 --attention-backend aiter 运行 Kimi-K3 模型的用户。长上下文（ ISL>50K）解码性能获得 94%-168% 的巨大提升，8k1k 基准测试 TTT 也有 4-10% 的改善。对于使用其他模型或 FP16/BF16 KV 缓存的用户，行为无变化。系统影响：核心注意力解码路径 (AiterAttnBackend) 增加了动态路由和状态管理。aiter_mla_gluon 模块作为新的职责点被引入，其初始化时的日志输出有助于运维监控。团队影响：需要维护新的硬件特定优化路径和对应的测试用例。对 aiter 上游的依赖关系更加紧密。
- 风险标记：运行时依赖版本约束，CUDA 图兼容性，硬件特定路径

关联脉络

- PR #37307 fix(unified-memory): forward the KV-index translator through every wrapper backend: 同属 KV 缓存与注意力后端一致性修复线。本 PR 优化了特定模型（ Kimi-K3）的解码路径，而 #37307 修复了 wrapper 后端的通用问题（未转发 KV 索引翻译器），两者都影响 MLA 注意力后端的正确性和性能。
- PR #37339 [Fix] Use real ReqKvInfo in unit-test req mocks: 测试基础设施维护。本 PR 新增的测试使用 mock，而 #37339 修复了其他测试中因 mock 不真实导致的 kv 信息不一致问题，共同提升了 KV 缓存相关测试的保真度。
- PR #37299 refactor(hicache): simplify decode offload state bookkeeping: 同属解码路径优化。本 PR 优化了 Kimi-K3 的 MLA 解码内核，而 #37299 重构了异步解码状态管理（ hicache offload），两者都在提升解码阶段的效率和稳定性。