PR #22868 完整报告

sgl-project/sglang

[Apple Silicon] Add custom Metal RoPE kernel with fused KV cache store

合并时间：2026-05-29 15:09

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22868

执行摘要

- 一句话：为 Apple Silicon 添加融合 KV 缓存的 Metal RoPE 内核
- 推荐动作：建议重点关注 MlxAOTKernelRegistry 可扩展设计，作为后端自定义内核的注册典范。同时关注 partial RoPE 兼容性处理和环境变量开关的默认配置。值得参考 3D 线程网格优化和零拷贝 buffer 捐赠策略。

功能与动机

基于 Apple Device Support 路线图（#19137），减少 Metal 命令缓冲区开销（每个内核派发约1-4msGPU空闲时间），通过融合操作降低派发次数以改善解码延迟。详见issue#22114。

实现拆解

1Metal内核源文件(sgl-kernel/csrc/metal/rope_pool_fused.metal)：编写三个函数rope_q、rope_k_pool、v_to_pool，采用 3D 线程网格 (dim, token, head) 消除 div/mod 热路径，通过函数常量特化形状参数，内联三角计算，支持 heads-per-thread 摊销。

2. C++ Primitive 集成(sgl-kernel/csrc/metal/rope_pool_fused.cpp)：继承 mlx::core::Primitive 定义 RopePoolFused 类，eval_gpu 中加载预编译 .metallib，构建三种不同流水线，通过 copy_shared_buffer 实现 KV pool 的零拷贝捐赠。

3. Python 包装(sgl-kernel/python/sgl_kernel/metal.py)：增加 rope_pool_fused 函数，做完整输入验证后调用 C++ 扩展，不强制 mx.eval，保持惰性图。

4. AOT 内核注册中心(python/sglang/srt/hardware_backend/mlx/aot.py)：定义 MlxAOTKernelRegistry、MlxAOTKernelSpec 等数据类，提供 build_kernel_set 方法。当前注册了名为 "rope" 的规格，启用条件为环境变量 SGLANG_MLX_USE_CUSTOM_ROPE=true 且模型 RoPE 兼容（rope_dim == head_dim，非传统 RoPE）。

5. 模型运行器集成(python/sglang/srt/hardware_backend/mlx/model_runner.py)：在初始化时调用 _build_aot_kernels 构建内核集。BatchedDecodeContext 增加 aot 字段（MlxAOTKernelContext），from_decode 类方法构造时注入。

6. 注意力包装器调用(python/sglang/srt/hardware_backend/mlx/kv_cache/attention_wrapper.py)：在 _batched_decode 中判断 ctx.aot.rope，若可用则调用自定义 _rope_custom_aot 一次性完成 Q/K 旋转和 KV pool 写入，跳过 MLX 内置 RoPE 和逐请求循环。

7. 环境变量与文档：python/sglang/srt/environ.py 添加 SGLANG_MLX_USE_CUSTOM_ROPE（默认关闭），更新 apple_metal.mdx 文档，删除旧的占位文件。

关键文件：

- python/sglang/srt/hardware_backend/mlx/aot.py（模块 内核注册；类别 source；类型 dependency-wiring；符号 _load_metal_rope_pool_fused, MlxAOTRoPEKernel, enabled, MlxAOTKernelBuildInputs）：首次引入 MLX AOT 内核注册中心，定义内核 specification 和 registry，是未来所有 AOT 内核的集合点。
- sgl-kernel/csrc/metal/rope_pool_fused.cpp（模块 融合内核；类别 source；类型 dependency-wiring；符号 RopePoolFused, register_library_impl, pick_tg, pick_heads_per_thread）：C++ 入口，实现 RopePoolFused primitive，包含 register_library、3D thread-group 选择、heads-per-thread 选择、零拷贝 pool 捐赠。
- python/sglang/srt/hardware_backend/mlx/kv_cache/attention_wrapper.py（模块 注意力包装；类别 source；类型 core-logic；符号 from_decode, _rope_custom_aot）：核心集成点，在 batched decode 中使用自定义 RoPE 内核替代 MLX 内置 RoPE，融合 KV pool 写入。
- python/sglang/srt/hardware_backend/mlx/model_runner.py（模块 模型运行器；类别 source；类型 data-contract；符号 _build_aot_kernels）：模型运行器入口，负责在初始化时构建 AOT 内核集，并在 decode_batch_start 中传递到上下文。
- sgl-kernel/python/sgl_kernel/metal.py（模块 内核绑定；类别 source；类型 core-logic；符号 rope_pool_fused）：Python 到 C++ 的桥梁，实现 rope_pool_fused 函数的输入验证并调用底层 primitive。

关键符号：_load_metal_rope_pool_fused, MlxAOTKernelRegistry.build_kernel_set, _build_rope_kernel, RopePoolFused.eval_gpu, rope_pool_fused, _rope_custom_aot, _build_aot_kernels, BatchedDecodeContext.from_decode

关键源码片段

python/sglang/srt/hardware_backend/mlx/aot.py

首次引入 MLX AOT 内核注册中心，定义内核 specification 和 registry，是未来所有 AOT 内核的集合点。

```
# 文件：python/sglang/srt/hardware_backend/mlx/aot.py
# MlxAOTKernelRegistry 是 MLX 后端的 AOT 内核注册中心。

from dataclasses import dataclass, field
from typing import Any, Callable, Optional

class MlxAOTKernelRegistry:
    # 为可选 MLX AOT 内核提供注册能力。
    # 每个 spec 拥有 MlxAOTKernelSet 上的一个内核字段。
    def __init__(self, specs: tuple[MlxAOTKernelSpec, ...]) -> None:
        self._specs = specs
```

```
    def build_kernel_set(
        self,
        *,
        sample_attn: Any,
        n_kv_heads: int,
        head_dim: int,
    ) -> MlxAOTKernelSet:
        # 遍历所有注册 spec ， 构建启用的内核集。
        inputs = MlxAOTKernelBuildInputs(
            sample_attn=sample_attn,
            n_kv_heads=n_kv_heads,
            head_dim=head_dim,
        )
        kernel_set = MlxAOTKernelSet()
        selected = []
        for spec in self._specs:
            if not spec.is_enabled():
                # 内核未启用则跳过
                continue
            kernel = spec.build(inputs)
            # 只有真正启用的内核才记录（如 rope_kernel.enabled 为 True）
            if getattr(kernel, "enabled", False):
                if not hasattr(kernel_set, spec.kernel_attr):
                    raise ValueError(
                        f"AOT kernel {spec.name} targets unknown kernel-set "
                        f"attribute {spec.kernel_attr}"
                    )
                setattr(kernel_set, spec.kernel_attr, kernel)
                selected.append(spec.name)
        kernel_set.selected_kernel_names = tuple(selected)
        if kernel_set.selected_kernel_names:
            logger.info("MLX AOT kernels selected: %s", ", ".join(kernel_set.selected_kernel_names))
        return kernel_set
```

评论区精华

review 中几个关键讨论：

- 正确性：gemini-code-assist[bot] 指出当 rope_dim < head_dim 时内核未拷贝非旋转部分导致 KV 缓存损坏。alexnails 确认并建议添加断言。最终在 _build_rope_kernel 中增加条件判断，不满足时回退到 MLX 内置 RoPE。
- 性能权衡：yeahdongcn 基于基准数据（Qwen-like 形状 0.64x-0.86x 速度），建议默认关闭并提供环境变量 SGLANG_MLX_USE_CUSTOM_ROPE。adityavaid 采纳。
- 架构设计：yeahdongcn 建议使用显式 MlxAOTKernelContext 抽象而非在 BatchedDecodeContext 中直接添加字段。后续实现为 @dataclass MlxAOTKernelContext 与 from_decode 类方法。

- Partial RoPE 导致 KV 缓存损坏 (correctness): 在 _build_rope_kernel 中增加 rope_dim != head_dim 时返回空内核，触发回退到 MLX 内置 RoPE。
- 默认关闭 AOT 内核 (performance): 引入 SGLANG_MLX_USE_CUSTOM_ROPE，默认 false。
- AOT 内核注册抽象设计 (design): 采用 @dataclass MlxAOTKernelContext 和 from_decode 类方法。

风险与影响

- 风险：
  1. 性能回退：Qwen-like 小 KV head 模型上 AOT 内核比 MLX 内置 RoPE 慢（如 BS=512 时 0.64x），需用户按需启用。
  2. partial RoPE 兼容性：通过显式检查 rope_dim == head_dim 触发回退，保障 Gemma 等部分旋转模型正确性。
  3. 测试覆盖不足：仅有手动 E2E 测试，缺少单元测试和 CI 集成。未来 AOT 内核需建立测试套件。
  4. 构建依赖：要求 Xcode Command Line Tools 和 metallib，在 macOS 上是合理要求但可能影响开发体验。 - 影响：影响范围限定于 Apple Silicon（M 系列）用户，首次引入自定义 Metal 内核，为未来更多 AOT 内核奠定设计模式。性能提升仅在特定形状（LLaMA-like 大 batch）显著，小 KV head 形状有回退。默认关闭，用户需显式启用。团队需维护 AOT 内核注册机制和构建脚本。 - 风险标记：缺少单元测试覆盖，小 KV head 模型性能回退，依赖 Metal 工具链，Partial RoPE 兼容性需确保回退

关联脉络

- PR #19137 [Roadmap] Apple Device Support (2026 Q1): 此 PR 是该路线图的一部分，实现自定义 Metal 内核以支持 Apple Silicon。
- PR #22114 [Apple Silicon] Enable overlap scheduling: 该 issue 调查了 Metal 命令缓冲区开销，直接促成了此 PR 的优化方向。