

PR #47361 完整报告

vllm-project/vllm

Delete PagedAttention

合并时间: 2026-07-03 03:31

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/47361>

执行摘要

- 一句话: 删除弃用的 PagedAttention v1/v2 实现
- 推荐动作: 此 PR 是重要的清理动作, 值得开发者关注其背后的决策: vLLM 已完成注意力后端的现代化迁移。阅读 diff 可清晰了解旧代码的结构和如何被替换, 对于理解 vLLM 的注意力架构演进非常有帮助。

功能与动机

PR body 仅写 "It is time.", 表明 PagedAttention 已被新版注意力后端 (如 FlashInfer、V2 Model Runner) 完全替代, 移除旧代码的技术债务时机成熟。

实现拆解

1. 删除 CUDA 内核源文件: 移除 `csrc/libtorch_stable/attention/` 下的 `attention_kernels.cuh`、`paged_attention_v1.cu`、`paged_attention_v2.cu`, 这些是 PagedAttention 的底层 GPU 实现。
2. 移除 Python 绑定和函数入口: 在 `vllm/_custom_ops.py` 中删除 `paged_attention_v1` 和 `paged_attention_v2` 两个 Python 包装函数, 它们直接调用 `torch.ops._C` 下的同名算子。
3. 清理 C++ 头文件声明和绑定: 从 `csrc/libtorch_stable/ops.h` 移除两个函数的声明, 并在 `csrc/libtorch_stable/torch_bindings.cpp` 中删除 `TORCH_LIBRARY` 注册和相关绑定逻辑。
4. 更新 benchmark 和测试: 修改 `benchmarks/kernels/benchmark_paged_attention.py` 和 `tests/kernels/attention/test_attention.py`, 移除 `version` 参数和条件分支, 只保留 ROCm 专用 kernel 的调用, 并限制 head sizes 为 ROCm 支持的 [64, 128]。
5. 调整构建配置: 在 `CMakeLists.txt` 中移除对已删除的 `.cu` 文件的编译。

关键文件:

- `benchmarks/kernels/benchmark_paged_attention.py` (模块 benchmark; 类别 source; 类型 core-logic; 符号 `run_cuda_benchmark`, `run_benchmark`): 核心 benchmark, 大幅简化: 移除版本选择逻辑, 仅保留 ROCm 路径, 函数重命名为 `run_benchmark`。
- `vllm/_custom_ops.py` (模块 Python API; 类别 source; 类型 core-logic; 符号 `paged_attention_v1`, `paged_attention_v2`): 删除了 `paged_attention_v1` 和 `v2` 两个公共 API 函数, 所有调用点需迁移到新版后端。

- tests/kernels/attention/test_attention.py (模块 测试; 类别 test; 类型 test-coverage) : 测试适配: 移除 v1/v2 参数化, 只测试 ROCm 路径, head sizes 收缩到 64/128。
- csrc/libtorch_stable/ops.h (模块 C++ 头文件; 类别 source; 类型 core-logic) : 头文件删除了 paged_attention_v1 和 v2 的 C++ 声明, 避免链接未定义符号。
- csrc/libtorch_stable/attention/attention_kernels.cuh (模块 CUDA 内核; 类别 other; 类型 deletion) : 被整文件删除 (667 行), 包含 paged_attention_kernel 模板和辅助函数。
- csrc/libtorch_stable/attention/paged_attention_v2.cu (模块 CUDA 内核; 类别 other; 类型 deletion) : v2 kernel 启动器被删除, 202 行。
- csrc/libtorch_stable/attention/paged_attention_v1.cu (模块 CUDA 内核; 类别 other; 类型 deletion) : v1 kernel 启动器被删除, 190 行。
- csrc/libtorch_stable/torch_bindings.cpp (模块 C++ 绑定; 类别 source; 类型 core-logic) : 移除 v1/v2 算子的 PyTorch 绑定注册, 避免编译错误。
- CMakeLists.txt (模块 构建脚本; 类别 infra; 类型 configuration) : 构建系统移除对已删除 .cu 文件的引用。

关键符号: paged_attention_v1, paged_attention_v2

关键源码片段

benchmarks/kernels/benchmark_paged_attention.py

核心 benchmark, 大幅简化: 移除版本选择逻辑, 仅保留 ROCm 路径, 函数重命名为 run_benchmark。

```
# benchmarks/kernels/benchmark_paged_attention.py
# 变更后: 版本参数被移除, 仅保留 ROCm kernel 调用

NUM_BLOCKS = 128 * 1024
PARTITION_SIZE_ROCM = 256 # 不再使用通用 PARTITION_SIZE

@torch.inference_mode()
def main(
    num_seqs: int, # 移除了 version 参数
    seq_len: int,
    ...
) -> None:
    ...
    # 不再按 version 初始化, 直接分配 ROCm 需要的临时缓冲区
    num_partitions = (max_seq_len + PARTITION_SIZE_ROCM - 1) // PARTITION_SIZE_ROCM
    tmp_output = torch.empty(
        size=(num_seqs, num_query_heads, num_partitions, head_size), ...)
    exp_sums = torch.empty(
        size=(num_seqs, num_query_heads, num_partitions), dtype=torch.float32, ...)
    max_logits = torch.empty_like(exp_sums)

    def run_benchmark(num_iters: int, profile: bool = False) -> float:
        ...
```

```

for _ in range(num_iters):
    # 直接调用 ROCm kernel, 无需版本判断
    ops.paged_attention_rocm(
        output, exp_sums, max_logits, tmp_output,
        query, key_cache, value_cache, num_kv_heads,
        scale, block_tables, seq_lens, None, block_size,
        max_seq_len, alibi_slopes, kv_cache_dtype, k_scale, v_scale)
    ...

if __name__ == "__main__":
    logger.warning(
        "This script benchmarks the ROCm paged attention kernel. "
        "By default this is no longer used in vLLM inference.")
    if not current_platform.is_rocm():
        raise RuntimeError("This benchmark requires the ROCm platform.")

```

vllm/_custom_ops.py

删除了 paged_attention_v1 和 v2 两个公共 API 函数，所有调用点需迁移到新版后端。

```

# vllm/_custom_ops.py
# 变更后：删除了整个 "# page attention ops" 下面的 v1/v2 函数

# 之前存在的 paged_attention_v1 和 paged_attention_v2 被完全移除
# 只保留 paged_attention_rocm (用于 ROCm 平台)

def paged_attention_rocm(
    out: torch.Tensor,
    exp_sum: torch.Tensor,
    max_logits: torch.Tensor,
    tmp_out: torch.Tensor,
    query: torch.Tensor,
    ...
) -> None:
    torch.ops._rocm_C.paged_attention(
        out, exp_sum, max_logits, tmp_out, query,
        key_cache, value_cache, num_kv_heads, scale, ...)

```

tests/kernels/attention/test_attention.py

测试适配：移除 v1/v2 参数化，只测试 ROCm 路径，head sizes 收缩到 64/128。

```

# tests/kernels/attention/test_attention.py
# 变更后：测试仅针对 ROCm paged attention

# 不再定义 PARTITION_SIZE 全局量，只保留 PARTITION_SIZE_ROCM
PARTITION_SIZE_ROCM = 256
# head sizes 缩小到 ROCm 支持的范围
HEAD_SIZES = [64, 128]

@pytest.mark.skipif(

```

```

    not current_platform.is_rocm(), reason="ROCm-only paged attention kernel")
@pytest.mark.parametrize(...)
def test_paged_attention(
    kv_cache_factory,
    # version 参数被移除
    num_seqs: int,
    ...
) -> None:
    ...
    # 直接使用固定分区大小，不再根据 version 初始化临时缓冲区
    num_partitions = (max_seq_len + PARTITION_SIZE_ROCM - 1) // PARTITION_SIZE_ROCM
    ...
    ops.paged_attention_rocm(...)

```

评论区精华

此 PR 没有实质性 review 讨论，仅有 [claude\[bot\]](#) 的自动提醒和两位 maintainer（[WoosukKwon](#)、[yewentao256](#)）的快速批准，说明该清理工作已提前达成共识。

- 暂无高价值评论线程

风险与影响

- 风险：兼容性风险：如果仍有代码或外部依赖调用 `paged_attention_v1/v2` 或直接依赖旧 CUDA 内核，将出现导入错误或运行时缺失。但 PR 表明这些路径已被彻底弃用，且 vLLM V1 引擎和 V2 Model Runner 已不再依赖它们。平台风险：ROCm 路径 `paged_attention_rocm` 保留，但非 ROCm 平台（NVIDIA CUDA）不再使用此 kernel；之前 NVIDIA 用户通过 `v1/v2` 回退的行为被移除，必须依赖新版注意力后端（如 FlashInfer、Triton）。测试覆盖：测试仅限于 ROCm；NVIDIA 及其他平台的注意力测试需要通过其他测试文件覆盖（如 `test_attention_backends.py`）。
- 影响：对用户：无直接影响，因为推理已自动切换到新版注意力后端。用户若强制指定旧版可能失败（但官方已不推荐）。对系统：代码库减少约 1472 行，降低了维护成本，消除了对陈旧 CUDA 代码的依赖。对团队：不再需要维护两套注意力 kernel，精力可集中在新后端的优化上。
- 风险标记：核心路径删除，平台限制变更，测试范围收缩

关联脉络

- PR #43232 Xqa decode kernels: 引入了 FlashInfer XQA 解码路径，逐步淘汰 PagedAttention，为删除做准备。
- PR #44443 [ModelRunner V2] Enable by default for all dense models: 默认启用 V2 Model Runner，不再依赖旧版 PagedAttention。
- PR #46838 [Bugfix][Kernel] Correct FlashInfer CUTLASS MoE tuning token bound: 修复 FlashInfer MoE kernel，进一步巩固新后端的地位。