PR #48223 完整报告

vllm-project/vllm

[Perf][ROCm] Dual-stream decode with hipgraphs

合并时间：2026-08-12 09:32

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48223

执行摘要

- 一句话：ROCm 启用双流 shared experts 解码，DP 下 TPOT 提升数个点
- 推荐动作：值得精读。这是跨平台双流调度的一次关键重构：事件化同步（enqueue/wait）替代 wait_stream 是保证 hipgraph 安全的核心设计，_should_enable_stream_overlap_heuristic 用 benchmark 数据驱动平台差异开关的做法也值得借鉴。建议重点阅读 shared_experts.py 的 maybe_forward_async/wait 与 moe_runner.py 的 _forward_impl 编排改动，并关注后续是否有测试补充。

功能与动机

Issue #48111 指出 SharedExpertsOrder.MULTI_STREAM_OVERLAPPED 双流解码此前仅支持 CUDA，而 ATOM 推理引擎在 ROCm 上使用双流解码可使 MoE 层耗时减少约 20%（DPA 部署、18 并发）。PR 提到相关 PR#38665 的一行式修复（放宽为 CUDA-like）不足以让该特性在 HIP graphs 下正确工作，因此需要重新设计跨流同步机制。作者还观察到仅放宽判定会导致 shared expert 在 routed experts 完成后才于 aux stream 启动，形成无重叠的顺序执行，这正是需要把启动时机提前的原因。

实现拆解

1. 放宽平台判定并加入 DP 启发式：在 vllm/model_executor/layers/fused_moe/runner/shared_experts.py 中把 _determine_shared_experts_order 的平台检查从 current_platform.is_cuda() 改为 is_cuda_alike()，并新增 _should_enable_stream_overlap_heuristic 属性。该启发式在非 ROCm 平台恒为 True；在 ROCm 上仅当 dp_size > 1 时返回 True，这是因为作者对 TP8EP 的 benchmark 显示 TP 下双流除低并发外均劣于顺序执行，而 fused shared experts（FSE）反而更优。
2. 重构跨流同步 API：删除 maybe_sync_shared_experts_stream 与 _run_in_aux_stream，改为 maybe_forward_async + wait 两段式。构造时为每个 DBO ubatch id 维护一对 torch.cuda.Event（_input_ready_event/_output_ready_event）。maybe_forward_async 在主流记录输入就绪事件、在 aux stream 等待该事件后立即执行 shared experts，并记录输出就绪事件；wait 让主流只等待输出事件而非整流同步。这一事件化设计既允许与 routed experts 计算重叠，也保证 hipgraph/cudagraph 可以安全捕获。
3. 调整 MoE runner 编排顺序：在 moe_runner.py 的 _forward_impl 中，于 routed expert dispatch（含 gate 计算、_maybe_dispatch）之前调用 maybe_forward_async 提前启动 shared experts；_apply_quant_method 新增 shared_experts_overlapping: bool

= False 参数，在 fused MoE 内核完成后调用 self._shared_experts.wait() 统一收敛，并移除原先在 _apply_quant_method 内基于 SharedExpertsOrder.MULTI_STREAM_OVERLAPPED 的二次启动逻辑。forward 同步分支因此简化为直接执行 self._layer(...)。

4. 配置与互斥约束：VLLM_DISABLE_SHARED_EXPERTS_STREAM=1 仍可整体关闭双流；与 VLLM_ROCM_USE_AITER_FUSION_SHARED_EXPERTS 互斥。本次 PR 未附带任何测试文件变更，验证完全依赖 MI300/B200 上的手工 benchmark 与 trace 截图。

关键文件：

- vllm/model_executor/layers/fused_moe/runner/shared_experts.py（模块 共享专家；类别 source；类型 core-logic；符号 _should_enable_stream_overlap_heuristic, maybe_forward_async, wait, _determine_shared_experts_order）：核心改动文件：新增 _should_enable_stream_overlap_heuristic 平台差异开关，将流同步从 wait_stream 重构为事件对机制，新增 maybe_forward_async/wait API，是双流解码能真正重叠且兼容 hipgraph 的关键。
- vllm/model_executor/layers/fused_moe/runner/moe_runner.py（模块 MoE 调度；类别 source；类型 core-logic；符号 _apply_quant_method, _forward_impl）：编排层改动：_forward_impl 把 shared experts 启动提前到 dispatch 之前，_apply_quant_method 新增 shared_experts_overlapping 参数并在 fused MoE 计算后统一 wait()，移除旧的二次启动逻辑。

关键符号：maybe_forward_async, wait, _should_enable_stream_overlap_heuristic, _apply_quant_method, _forward_impl, _determine_shared_experts_order

关键源码片段

vllm/model_executor/layers/fused_moe/runner/shared_experts.py

核心改动文件：新增 _should_enable_stream_overlap_heuristic 平台差异开关，将流同步从 wait_stream 重构为事件对机制，新增 maybe_forward_async/wait API，是双流解码能真正重叠且兼容 hipgraph 的关键。

```
# vllm/model_executor/layers/fused_moe/runner/shared_experts.py
# 核心：把“先同步、再上 aux stream 执行”改成“先异步 enqueue、事后 wait”。

def maybe_forward_async(self, shared_experts_input: torch.Tensor) -> bool:
    """Enqueue shared experts on the aux stream without waiting for them.

    Returns true if the shared experts were enqueued, false otherwise. Call
    `wait` to wait for the shared experts to finish if this returns true.
    """
    # 只有多流重叠模式被选中时才真正异步提交，否则返回 False，
    # 调用方会走同步路径。
    if (
        self._determine_shared_experts_order(shared_experts_input)
        != SharedExpertsOrder.MULTI_STREAM_OVERLAPPED
    ):
        return False
```

```
        assert self._stream is not None
        idx = self._output_idx
        assert self._output[idx] is None
        # 在主流上记录"输入已就绪"事件，供 aux stream 等待，
        # 避免在 shared experts 启动前主流的输入尚未写完。
        self._input_ready_event[idx].record(current_stream())
        with torch.cuda.stream(self._stream):
            # aux stream 等待输入就绪后立即启动 shared experts，
            # 从而与后续 routed experts 的 dispatch/ 计算真正重叠。
            self._input_ready_event[idx].wait(self._stream)
            self._output[idx] = self._layer(shared_experts_input)
            # 记录"输出已就绪"事件，主流只需等这个事件即可，
            # 无需整流 wait_stream，保证 cudagraph/hipgraph 可捕获。
            self._output_ready_event[idx].record(self._stream)
        return True

    def wait(self) -> None:
        """Block the main stream until `maybe_forward_async` output is ready."""
        assert self._stream is not None
        # 让主流在 routed experts 计算完成后才等待 shared experts 结果，
        # 两个计算流在此期间完全并行。
        self._output_ready_event[self._output_idx].wait(current_stream())
```

评论区精华

1. 单节点收益存疑（tjtanaa）：tjtanaa 在 issue 评论中质疑 DBO（dual-batch overlap）主要对 wide EP 部署有效，要求补充低并发数据，"It seems smaller concurrencies might have large degradation, and I am concern with the effects on these low latency use case"。作者补充 1/2/4/8/32 并发数据后，结论是低并发下 TPOT 仍提升约 4%，疑虑消除。
2. TP 回退成为开关依据：作者提交 TP8EP 实测：TP 下 8 并发起双流开始劣于顺序执行（-1.38%），FSE 优于两者，因此最终决定 ROCm 仅在 DP > 1 时启用双流，避免对 TP 用户造成回归。maeehart 评论认可："Since we cannot use FSE with Mori a2a kernels, this makes a lot of sense to enable. Otherwise, FSE is the most performant option."
3. 启发式范围之争（mpashkovskii vs simondanielsson）：review 中 mpashkovskii 认为 _should_enable_stream_overlap_heuristic 仅在 DPA decode 生效过于保守，"parallel shared experts are also beneficial for other setups"。作者回应 benchmark 显示 DP 受益而 TP 受损，并指出 NVIDIA 平台始终默认启用；该讨论最终以保留 DP guard 合并收尾。
4. NVIDIA 兼容性确认（SageMoore）：SageMoore 要求确认该 code path 对 NVIDIA 各后端无影响，作者在 B200 上验证并贴出 nsys trace，确认本分支与 main 的图结构和流分布一致。
5. CG profiling 影响（Rohan138）：Rohan138 提醒需确认是否受 CUDA graph profiling 影响（引用 PR#48526）；作者 rebase 后重跑 CI（Buildkite #83133）最终通过。

- DBO 单节点收益与低并发性能疑虑 (performance): 作者补测后显示低并发 TPOT 仍提升约 4%（1k/1k 与 8k/1k），疑虑消除。

- TP 下双流回退与开关决策 (performance): 保留 dp_size > 1 的启发式开关，TP 默认回退到顺序执行。
- _should_enable_stream_overlap_heuristic 是否过于保守 (design): 维护者接受 DP guard 设计，评论线程以作者说明结束。
- NVIDIA 硬件行为一致性 (question): 作者在 B200 上验证并贴出 trace，确认与 main 行为一致。
- CUDA graph profiling 影响 (correctness): 作者 rebase 后重跑 CI（Buildkite #83133），最终通过并合并。

风险与影响

- 风险：
  1. TP 配置回归已规避但有条件：通过 dp_size > 1 启发式关闭 ROCm TP 场景，但该开关依赖 MoE 并行配置读取时机，若配置在实例化后才变化（如弹性 EP），_should_enable_stream_overlap_heuristic 可能读到旧值。
  2. 图捕获正确性依赖事件语义：maybe_forward_async 的 _input_ready_event/_output_ready_event 在 CUDA graph 捕获期间会被固定，若与 DBO ubatch id 交错（_output_idx 随 dbo_current_ubatch_id() 变化）可能出现事件错配；B200 验证覆盖 NVIDIA，但 ROCm hipgraph 不同捕获路径未见专项测试。
  3. 无测试覆盖：两个核心运行文件变更均无对应单元测试，回归只能靠手工 benchmark 与 trace 发现（后续 PR 需补）。
  4. 高并发与精度轻微波动：1k/1k 在 16 并发时 TPOT 回退 -6.77%；GSM8k 从 nightly 的 0.9484 降到 0.9409，虽可能是噪声，但需要在正式发布前确认不是 shared experts 结果序变化导致的数值差异。
  5. 与 FSE/AITER 的互斥：仅靠文档说明互斥，若用户同时开启 VLLM_ROCM_USE_AITER_FUSION_SHARED_EXPERTS 与双流，行为未定义，缺少显式运行时告警。 - 影响：影响面集中在 fused MoE 的 shared experts 调度路径，属于所有带 shared experts（如 DeepSeek-V3 等 MoE 模型）的解码热路径。ROCm + DPA 用户获得约 3% - 5% 的 TPOT 改善（低并发），高并发最高 +11%；TP 用户行为不变（默认关闭）。NVIDIA 路径只是将同步方式从 wait_stream 换成事件等待，trace 确认无行为变化。对团队而言，该 PR 统一了 CUDA/ROCm 的跨流调度机制，为后续 wide EP/PD 场景的 DBO 演进打下基础，但缺少测试配套是后续维护的主要负担。 - 风险标记：核心解码路径变更，缺少测试覆盖，TP 下默认关闭（bench 支撑），跨平台行为差异，无新增测试文件

关联脉络

- PR #49758 [ROCm][MoE] Fix expert_map vs AITER expert_mask for non-AITER experts under EP: 同为 ROCm 下 fused MoE 执行路径的正确性修复，涉及 routed_experts.py 与 AITER 内核的交互，与本 PR 的 shared experts 调度同属 fused_moe 模块。

- PR #50268 [Hardware][AMD] Enable fused bf16→fp32 router GEMM on ROCm: ROCm MoE 性能优化线（路由 GEMM 融合），与本 PR 改善 MoE 层解码延迟的目标一致。
- PR #50654 [ROCm][Perf] Kimi-K3 Fused kernel for KDA decode: ROCm 解码路径的专用融合内核优化，同属 ROCm 解码性能演进脉络。