PR #50534 完整报告

vllm-project/vllm

[XPU] Add tuned Mamba SSU configs for Intel Arc Pro B70

合并时间：2026-08-13 20:36

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50534

执行摘要

- 一句话：B70 新增 Mamba SSU 调优配置，修复 XPU 调优脚本
- 推荐动作：值得精读，重点看三个设计决策：float16 不复制 float32 而是独立调优（用数据说明精度对 kernel 最优解的影响）、以 10% 收益阈值决定是否采纳重调条目（防止 tuner 与模型 ngroups 不一致造成的过拟合）、以及把 torch.cuda.* API 迁移到 current_platform / torch.accelerator 的适配模式。对要在新硬件上做 kernel 调优的贡献者，这是一个完整可复制的流程：工具修复 → 搜索空间设计 → 权衡取舍 → 数值与 E2E 验证。

功能与动机

vLLM 的 selective_state_update 配置目录此前只覆盖 AMD 与 NVIDIA 设备，Intel GPU 上运行的 Mamba/hybrid 模型只能使用启发式启动参数，性能远低于调优配置。作者在 PR body 中说明：float16 必须独立调优而非复制 float32 结果，因为在 bfloat16 state 下 headdim=128,dstate=256 有 9/12 的网格点最优解不同，复用 float32 值会导致 kernel 级中位慢 79.6%；此外，原调优脚本无条件调用 torch.cuda.get_device_capability()，在 XPU 构建上会在调优开始前抛出 AssertionError: Torch not compiled with CUDA enabled，torch.cuda.CUDAGraph() 与 torch.cuda.Event 也会让所有候选被跳过，sweep 无结果。

实现拆解

1. 修复调优脚本的 CUDA 硬编码（benchmarks/kernels/benchmark_selective_state_update.py）：_make_inputs 的 device 默认值从 "cuda" 改为 current_platform.device_type；benchmark_config 中 graph 创建改为平台分支（current_platform.is_cuda_alike() 时用 torch.cuda.CUDAGraph()，否则用 torch.xpu.XPUGraph()），并以 current_platform.graph() 作为捕获上下文；计时事件从 torch.cuda.Event 改为 torch.Event；main 中 torch.cuda.get_device_capability() 改为 current_platform.get_device_capability() 并允许返回 None（打印 n/a）。这样脚本在 CUDA、ROCm、XPU 上都能运行，同时保留 CUDA 原有行为。
2. 确定搜索空间与调优方法论：按 effective_batch 网格（8~131072）调优 (BLOCK_SIZE_M, num_warps)；网格从 8 开始，因为 Mamba2-2.7b 在 concurrency=1 时按 batch 1 x 80 heads 运行，原从 128 开始的网格在该点比启发式慢约 10%；float16 独立调优；headdim=64,dstate=128 保留 incumbent 条目，仅当新调优结果相对启发式有超过 10% 差距时才采纳，因为 tuner 用 ngroups=8 而模型实际跑 ngroups=1，盲目选微

基准赢家在 E2E 上反而损失 4.2%。

3. 新增配置文件：在 vllm/model_executor/layers/mamba/ops/configs/selective_state_update/ 下新增 4 个 JSON，文件名按 (headdim, dstate, device_name, cache_dtype) 编码，运行时由 mamba_ssm 的 get_ssm_config_file_name 自动拾取，产品路径零改动。headdim=64,dstate={16,32,64,256} 虽已调优验证，因无模型可做 E2E 验证，按 review 建议删除，继续使用启发式。
4. 验证配套：数值验证对比 CPU reference 在模型真实 ngroups/nheads 下 8/8 与 7/7 通过，启发式对照组干净；E2E 用 vllm bench serve 在 Falcon-H1-7B 与 Mamba2-1.3b 上按 10 档 concurrency 测量，并在两台 B70 机器复现高增益结果。本 PR 未新增单元测试，验证依赖调优 / 基准流程本身。

关键文件：

- benchmarks/kernels/benchmark_selective_state_update.py（模块 基准工具；类别 source；类型 platform-adaptation；符号 _make_inputs, benchmark_config, main）：调优工具的平台适配核心：移除 torch.cuda 硬编码，引入 current_platform 与 XPUGraph，使脚本可在 XPU-only 构建运行，是本 PR 能够产出调优配置的前提。
- vllm/model_executor/layers/mamba/ops/configs/selective_state_update/headdim=64,dstate=128,device_name=Intel(R)_Arc(TM)_Pro_B70_Graphics,cache_dtype=float16.json（模块 SSU 配置；类别 infra；类型 configuration）：Mamba2 默认路径（float16 缓存）的调优配置，网格从 effective_batch=8 开始，直接体现 " 小 batch 下启发式更优 " 的修复成果。
- vllm/model_executor/layers/mamba/ops/configs/selective_state_update/headdim=64,dstate=128,device_name=Intel(R)_Arc(TM)_Pro_B70_Graphics,cache_dtype=float32.json（模块 SSU 配置；类别 infra；类型 configuration）：Mamba2 显式 float32 缓存路径的配置，若干档位（如 8192/16384 用 8/1）与 float16 文件不同，体现 dtype 对最优解的影响。
- vllm/model_executor/layers/mamba/ops/configs/selective_state_update/headdim=128,dstate=256,device_name=Intel(R)_Arc(TM)_Pro_B70_Graphics,cache_dtype=float16.json（模块 SSU 配置；类别 infra；类型 configuration）：FalconH1 默认路径（float16）的调优配置，也是 kernel 峰值收益最大的 shape（最高 4.47x），E2E TPOT 提升最高达 1.455x。
- vllm/model_executor/layers/mamba/ops/configs/selective_state_update/headdim=128,dstate=256,device_name=Intel(R)_Arc(TM)_Pro_B70_Graphics,cache_dtype=float32.json（模块 SSU 配置；类别 infra；类型 configuration）：FalconH1 显式 float32 路径的配置，多个档位使用 BLOCK_SIZE_M=8 且 num_warps=2，与 float16 文件差异明显，印证独立调优的必要性。

关键符号：_make_inputs, benchmark_config, main

关键源码片段

benchmarks/kernels/benchmark_selective_state_update.py

调优工具的平台适配核心：移除 torch.cuda 硬编码，引入 current_platform 与 XPUGraph，使脚本可在 XPU-only 构建运行，是本 PR 能够产出调优配置的前提。

```
def benchmark_config(
    batch: int,
    nheads: int,
    dim: int,
    dstate: int,
    ngroups: int,
    block_size_m: int,
    num_warps_val: int,
    dtype: torch.dtype,
    state_dtype: torch.dtype | None = None,
    num_iters: int = 100,
    num_warmup: int = 20,
    graph_batch_size: int = 10,
) -> float | None:
    """Time one (BLOCK_SIZE_M, num_warps) config for selective_state_update.

    返回微秒耗时，失败时返回 None。通过 graph capture 反复执行
    graph_batch_size 次 kernel，隔离 Python 调度与 kwarg 解析开销。
    """
    state, x, dt, A, B, C, D, dt_bias, out = _make_inputs(
        batch, nheads, dim, dstate, ngroups, dtype, state_dtype=state_dtype
    )

    def _call_kernel() -> None:
        selective_state_update(
            state, x, dt, A, B, C, D=D, z=None, dt_bias=dt_bias,
            dt_softplus=True, out=out,
        )

    try:
        with override_ssm_config((block_size_m, num_warps_val)):
            # Eager 模式 warmup：触发 Triton JIT 与 autotune，预热缓存
            for _ in range(num_warmup):
                _call_kernel()
            torch.accelerator.synchronize()

            # CUDA 需要 side-stream 捕获，而 XPU 上 torch.cuda.CUDAGraph()
            # 不可用，因此按平台创建 graph 对象，再统一用 current_platform.graph()
            # 作为捕获上下文，规避 torch.accelerator.Graph() 在 CUDA 未验证的风险
            graph = (
                torch.cuda.CUDAGraph()
                if current_platform.is_cuda_alike()
                else torch.xpu.XPUGraph()
            )
            with current_platform.graph(graph):
                for _ in range(graph_batch_size):
```

```
                _call_kernel()
            torch.accelerator.synchronize()

            # 预热 graph 回放，让运行时稳定
            for _ in range(5):
                graph.replay()
            torch.accelerator.synchronize()

            # torch.cuda.Event 在 XPU 构建不存在，torch.Event 是跨平台别名
            start = torch.Event(enable_timing=True)
            end = torch.Event(enable_timing=True)
            latencies: list[float] = []
            for _ in range(num_iters):
                start.record()
                graph.replay()
                end.record()
                end.synchronize()
                latencies.append(start.elapsed_time(end))
            graph.reset()
        # elapsed_time 返回 ms；每次回放跑 graph_batch_size 个 kernel，
        # 因此除以 (num_iters * graph_batch_size) 并换算为 us
        return sum(latencies) / (num_iters * graph_batch_size) * 1000
    except Exception as e:
        if "OutOfResources" not in str(e):
            print(
                f"    Warning: config M={block_size_m},w={num_warps_val} "
                f"raised {type(e).__name__}: {e}"
            )
        return None


# main() 中的设备能力探测：XPU 上 get_device_capability() 返回 None，
# 原 torch.cuda.get_device_capability() 会直接抛 AssertionError
cap = current_platform.get_device_capability()
is_blackwell = cap is not None and cap[0] >= 10 # 仅 CUDA 设备可能为 Blackwell
cap_str = f"sm_{cap[0]}{cap[1]}" if cap is not None else "n/a"
print(f"Device : {device_name}  ({cap_str})")
```

评论区精华

review 中最核心的讨论有两点：一是 jikunshang 在 PR 评论中要求删除没有模型覆盖的配置（"headdim=64,dstate={16,32,64,256} I float32 only I no model on hand ... we'd better remove unnecessary config."），作者将其采纳，最终只保留 FalconH1 与 Mamba2 实际使用的两种 shape；二是 jikunshang 在 diff hunk 上针对最初的 torch.accelerator.Graph() 写法提问 "is it safe here?"，因为该 API 在仓库内没有 CUDA 使用先例，最终 head 版本改为显式平台分支（CUDA 走 CUDAGraph()、XPU 走 XPUGraph()）并统一使用 current_platform.graph() 上下文，既保留了 CUDA 的 side-stream 捕获语义，又支持 XPU。之后 yma11 审核通过（"LGTM. @jikunshang please take a look."），jikunshang 最终

APPROVED。

- 删除无模型覆盖的调优配置 (design): 作者接受，最终 PR 仅保留 FalconH1 与 Mamba2 实际覆盖的两种 shape 共 4 个配置文件，未覆盖 shape 继续使用启发式。
- torch.accelerator.Graph() 在 CUDA 上的安全性 (design): 改为显式平台分支：CUDA 用 CUDAGraph()、XPU 用 XPUGraph()，并以 current_platform.graph() 作为捕获上下文，既保留 CUDA side-stream 语义又支持 XPU。

风险与影响

- 风险：
  1. 覆盖范围有限：四个配置文件均以精确设备名 Intel(R) Arc(TM) Pro B70 Graphics 命名，其他 Intel 显卡（如 Arc A 系列、B580 等）仍回退启发式，不享受本轮收益，但也不会变差。
  2. float16 配置同时服务 bfloat16 模型：_SSM_CACHE_DTYPE_MAP 将 bfloat16 映射到 float16 文件，本 PR 已针对 bf16 state 验证，但未来若引入新的 bf16 专属优化需重新审视该映射。
  3. benchmark 脚本的平台分支改动影响 CUDA 路径：current_platform.graph() 与 torch.cuda.graph() 的语义一致性依赖仓库平台抽象的正确实现，GPU CI 会覆盖该路径，但本 PR 没有为脚本本身添加单元测试。
  4. 配置依赖 Triton 版本：JSON 内记录 triton_version=3.7.1，Triton 升级后这些参数可能不再是全局最优，需要重新调优。
  5. 轻微工程问题：JSON 文件缺少 trailing newline，且无 schema 校验，不影响解析但容易被后续工具链误报。 - 影响：对用户而言，在 B70 上运行 FalconH1 系列 hybrid 模型与 Mamba2 系列模型时，decode 延迟与吞吐显著改善（显式 float32 下 TPOT 最高提升 1.455x / 1.316x，默认 float16 路径也有 2.6%~5.7% 收益），且未覆盖场景继续走启发式，行为无损。对系统而言，mamba_ssm 运行时按文件名自动拾取配置，无需修改产品代码，属于 zero-touch 接入，其他平台不受影响。对团队而言，修复后的 benchmark 脚本成为可复用的跨平台调优工具（与 benchmark_moe.py 的方法一致），后续为其他 Intel 设备调优可直接复用同一流程，也为社区贡献了对 CUDA/ROCm 更友好的基准基础设施。 - 风险标记：单设备精确匹配，覆盖范围有限，配置无自动化测试覆盖，benchmark 平台分支影响 CUDA 路径，配置依赖 Triton 3.7.1

关联脉络

- PR #50513 [XPU] update UMD to 26.27: 同一 Intel GPU（XPU）平台支持线，由同一位维护者 jikunshang 合入，本 PR 是该设备生态建设在 Mamba kernel 调优上的延续。
- PR #52092 [CPU] Ship triton-cpu wheel and fix several hardcoded pin_memory=True: 同为去除 torch.cuda 硬编码、推广平台无关 API（current_platform / torch.accelerator）的平台适配工作，与本 PR 的脚本修复思路一致。
- PR #51624 [Hardware][Power] Unqualized MoE Backend for Power (VSX): 同为非 NVIDIA/AMD 平台的 kernel 性能支持，体现了仓库对多硬件调优的通用方法论，与本 PR 互为补充。