

PR #38176 完整报告

sgl-project/sglang

keeping router GEMM in fp32 for deterministic inference (DeepSeek V3/V4)

合并时间: 2026-09-15 16:16

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/38176>

执行摘要

- 一句话: DeepSeek V3/V4 确定性推理下 router GEMM 保持 fp32, 避免专家选择翻转
- 推荐动作: 值得精读, 尤其适合关注确定性推理、MoE 路由数值稳定性和自研 kernel 接口演进的工程师。重点看三处设计决策: 一是把 dtype 语义交给猴补丁后的 torch.mm 而非硬 import kernel, 换取解耦但引入隐式契约; 二是 matmul_persistent 中「输出缓冲决定 fp32 store 分支可达性」这一根因, 以及 triton 调优参数仍按输入 dtype 选取的取舍; 三是 _mm_dtype_compat 去掉 .to(out_dtype) 后对全局调用点的影响面。建议后续补一个针对 router logits dtype 与 top-k 选择一致性的单测, 并显式校验 deterministic 模式与 batch-invariant 补丁的绑定关系。

功能与动机

PR body 明确说明这是 issue #34758 (确定性推理) 下的 wip: "under deterministic inference the DeepSeek router GEMM falls back to F.linear, which stores buffers in bf16"。而 MoEGate.forward 的其他路径 (dsv3_router_gemm/linear_bf16_fp32、prefill-CP) 都已经返回 fp32; 这些 logits 会进入 noaux_tc 的 top-k (256/384 专家) 并叠加 fp32 修正 bias, "using bf16's 8-bit mantissas here can change which experts are selected when scores sit within one ULP"。作者同时指出根因: 确定性 matmul 早已有 fp32 累加与 fp32 store 分支, 但 "it was unreachable because the output buffer was always allocated as the input dtype", 而 batch invariance 只关心归约顺序、不关心输出 dtype。

实现拆解

1. 改写 MoEGate 的确定性分派入口 (python/sglang/srt/models/deepseek_v2.py)

原实现只有一行 `return F.linear(hidden_states, self.weight, None)`, 输出落回 bf16。现改为两支: CUDA/HIP 走 `torch.mm(hidden_states, self.weight.t(), out_dtype=torch.float32)`, 其他后端走 `F.linear(hidden_states.float(), self.weight.float(), None)`。这样确定性模式下 router logits 的 dtype 契约与 tiny_gemm_bf16、linear_bf16_fp32 等分支统一为 fp32, 并且在任何 token 数下都不再触发 `max_router_gemm_tokens` 的 kernel 切换; 非确定性路径逐字未改。

2. 把 out_dtype 打进批次不变量 matmul 内核 (python/sglang/srt/batch_invariant_ops/batch_invariant_ops.py)

matmul_persistent 新增 out_dtype: torch.dtype | None = None, 内部 out_dtype = out_dtype or a.dtype 保证既有调用点零行为变化; _matmul_persistent_triton 与 _matmul_persistent_deepgemm 都新增 out_dtype 形参, torch.empty 改按 out_dtype 分配。关键细节是 triton 侧调优参数仍取 `**configs[a.dtype]`, 因为加载与累加的数据类型没有变, 变的只有输出缓冲的存储宽度, 从而让 kernel 内既有的 fp32 store 分支变得可达。

3. 修掉输出转换中的二次舍入, 并约束兜底路径

_mm_dtype_compat 由 matmul_persistent(...).to(out_dtype) 改为 matmul_persistent(..., out_dtype=out_dtype), 避免 fp32→bf16→fp32 的双重舍入; 这同时是 torch.mm(..., out_dtype=...) 在猴补丁下可直接工作的接口前提。einsum 兜底 (_ENABLE_MM_FALLBACK_VARIANT) 新增 out_dtype == a.dtype 门控, 防止没有 fp32 累加语义的路径冒充 fp32 输出, 否则回落到 persistent triton kernel。

4. 测试与 CI 配套

本 PR 未新增测试、配置或文档: 改动仅 2 个源码文件, 依赖既有的 test/registered/attention/test_deepseek_v3_deterministic.py 与 test/registered/unit/batch_invariant_ops/test_batch_invariant_ops.py (issue 评论显示这两个用例 rerun 后在 1-gpu-h100 上通过)。PR Test (Extra) 与 AMD ROCm 10 曾失败, 最终靠多次 /rerun-failed-ci 收尾。提交序列共 5 个 commit: 初版 → 依 review 补 "adding fp32 to non cuda backends" → 两次 merge main (由 b8zhong 操作) → 一次 MoEGate 分支的 ruff-format 修复。

关键文件:

- python/sglang/srt/models/deepseek_v2.py (模块 专家路由; 类别 source; 类型 core-logic; 符号 MoEGate.forward, MoEGate): MoEGate.forward 的确定性推理分支从 F.linear(bf16 输出) 改为 CUDA/HIP 上的 torch.mm(out_dtype=torch.float32) 以及非 CUDA 后端的显式 fp32 cast, 是本 PR 的行为入口; router logits 的 dtype 直接决定 256/384 专家 top-k 的结果。
- python/sglang/srt/batch_invariant_ops/batch_invariant_ops.py (模块 批不变算子; 类别 infra; 类型 infrastructure; 符号 matmul_persistent, _matmul_persistent_triton, _matmul_persistent_deepgemm, _mm_dtype_compat): 批次不变量 matmul 的核心实现: 为 matmul_persistent 及其 triton/deepgemm 后端引入 out_dtype, 使 fp32 输出缓冲可达; _mm_dtype_compat 去掉 compute-then-cast, 是让 monkeypatch 后的 torch.mm 支持 out_dtype 的关键, 影响所有确定性 matmul 调用点。

关键符号: MoEGate.forward, matmul_persistent, _matmul_persistent_triton, _matmul_persistent_deepgemm, _mm_dtype_compat

关键源码片段

python/sglang/srt/models/deepseek_v2.py

MoEGate.forward 的确定性推理分支从 F.linear(bf16 输出) 改为 CUDA/HIP 上的 torch.mm(out_dtype=torch.float32) 以及非 CUDA 后端的显式 fp32 cast, 是本 PR 的行为入口; router logits 的 dtype 直接决定 256/384 专家 top-k 的结果。

```

class MoEGate(nn.Module):
    # __init__ 中已根据 config 与 weight dtype 决定 tiny_router_gemm_max_tokens 等属性
    def forward(
        self,
        hidden_states,
        gemm_output_zero_allocator: BumpAllocator = None,
        forward_batch: ForwardBatch = None,
    ):
        # 权重本身就是 fp32 (未量化 / expert_pack 场景) 时, 直接做 fp32 线性层
        if self.weight.dtype == torch.float32:
            return F.linear(hidden_states.float(), self.weight)

        if use_intel_amx_backend(self):
            return torch.ops.sgl_kernel.weight_packed_linear(
                hidden_states,
                self.weight,
                None, # bias
                True, # is_vnni
            )

        # 确定性推理分支: 本 PR 的核心改动点, 之前这里只有一行
        # `return F.linear(hidden_states, self.weight, None)` , 输出会落回 bf16
        if get_exec().deterministic.enable_deterministic_inference:
            if _is_cuda or _is_hip:
                # torch.mm 已被 batch_invariant_ops 猴补丁为 matmul_persistent,
                # 因此这里的 out_dtype 会一路透传到 persistent kernel:
                # 输出缓冲直接按 fp32 分配, 既没有 bf16 中间结果, 也不需要
                # compute-then-cast, 自然避免了 fp32 -> bf16 -> fp32 的二次舍入。
                # 同时对任意 token 数都走同一条路径, 不再受 max_router_gemm_tokens 影响。
                return torch.mm(hidden_states, self.weight.t(), out_dtype=torch.float32)
            # 非 CUDA / HIP 后端 (NPU、CPU 等) 不支持 out_dtype,
            # 退化为把两侧输入都 cast 成 fp32 再算, 代价是两份额外 fp32 张量
            return F.linear(hidden_states.float(), self.weight.float(), None)

        # 非确定性路径保持原样: 小 batch 走 tiny_gemm_bf16, 其余按后端分派;
        # 这些分支本来就已经返回 fp32 logits, 与确定性分支的输出契约一致
        if hidden_states.shape[0] <= self.tiny_router_gemm_max_tokens:
            logits = tiny_gemm_bf16(
                hidden_states,
                self.weight,
                out_dtype=torch.float32,
                max_m=self.tiny_router_gemm_max_tokens,
            )
        elif _use_aiter:
            logits = aiter_dsv3_router_gemm(hidden_states, self.weight)
        elif not _is_cuda:
            logits = F.linear(hidden_states, self.weight, None)
        else:
            # cuBLAS bf16 x bf16 -> fp32 GEMM (torch.mm 的 out_dtype 仅 CUDA 可用)

```

```
from sglang.kernels.ops.attention.dsv4 import linear_bf16_fp32

logits = linear_bf16_fp32(hidden_states, self.weight)

return logits
```

评论区精华

讨论集中在两处，且都被作者采纳：

1. b8zhong 建议不要直接 import kernel，而是复用猴补丁入口："Is it possible to call: `return torch.mm(hidden_states, self.weight.t(), out_dtype=torch.float32)` And it could be monkeypatched directly right?"（指向当时还在显式 `from sglang.srt.batch_invariant_ops import matmul_persistent` 的版本）。作者回复 "ouuu good point"，最终 head 版本确为 `torch.mm(...)`，去掉了模型文件对 `batch_invariant_ops` 的硬依赖。
2. b8zhong 对非 CUDA 回退行提出 "Can we change it to FP32 (through casts)"，作者回 "good catch"，落地为 `F.linear(hidden_states.float(), self.weight.float(), None)`，对应 commit "adding fp32 to non cuda backends"。

BBuf 最终 APPROVED；无遗留未解决的技术分歧。

- 用猴补丁后的 `torch.mm` 取代直接 import `matmul_persistent` (design): 采纳。最终 head 版本改为调用 `torch.mm`，由 `batch_invariant_ops` 的猴补丁把它路由到 `matmul_persistent`，模型层不再硬依赖内核模块；代价是隐式要求补丁已安装。
- 非 CUDA/HIP 分支也要 cast 到 fp32 (correctness): 采纳，落地为 `F.linear(hidden_states.float(), self.weight.float(), None)`，对应 commit "adding fp32 to non cuda backends"，使所有后端在确定性模式下都输出 fp32。
- CI 反复 rerun 才通过，确定性用例覆盖依赖既有测试 (testing): 合入前依赖既有确定性 / 批次不变量集成测试兜底，未补充针对 router logits dtype 的定向用例；CI 最终转绿。

风险与影响

- 风险：
 1. 依赖猴补丁的隐含契约：最终实现调用 `torch.mm(..., out_dtype=torch.float32)`，只有 `torch.mm` 被 `batch_invariant_ops` 替换为 `_mm_dtype_compat` 时才合法。若出现「开启确定性推理但未安装批次不变量补丁」的组合（新增后端、独立配置路径、单测直接构造 MoEGate），该调用会落到原生 `torch.mm`；代码注释本身写着 `torch.mm` 的 `out_dtype` 仅 CUDA 可用，因此在非 CUDA 上更可能直接抛 `TypeError` 而不是静默降级。
 2. 共享内核输出契约变更：`_mm_dtype_compat` 从「按输入 dtype 算完再 `.to(out_dtype)`」变为「直接以 `out_dtype` 分配输出」，凡是带 `out_dtype` 的 `matmul` 调用点（不止 router GEMM）输出位模式都会变化。虽然方向上是更精确，但此前建立的确定性基线 / 重放结果可能出现差异，需要全量确定性用例复核。

3. 非 CUDA/HIP 回退的内存与拷贝成本: `F.linear(hidden_states.float(), self.weight.float(), None)` 会额外物化 `hidden_states` 与 `256/384xhidden_size` 的 router 权重两份 fp32 张量, 在 NPU/CPU 等后端只在确定性模式下可接受。
4. 兜底语义被静默改变: `einsum` 变体在 `out_dtype != a.dtype` 时被跳过 (即使 `_ENABLE_MM_FALLBACK_VARIANT` 打开), 行为正确但调试时容易误判。
5. 缺少针对性回归测试: 本 PR 无测试变更, ULP 级专家翻转问题没有专门的 `dtype/top-k` 一致性单测, 风险靠既有 `deterministic` 集成测试兜底; CI 的 Extra 与 AMD ROCm 工作流出现过失败需 rerun。- 影响: 用户侧: 开启 `deterministic inference` 的 DeepSeek V3/V4 部署, MoE 路由的专家选择在分数接近 ULP 时不再被 bf16 尾数翻转, 可复现性更接近理论承诺; 代价是 router logits 缓冲区从 bf16 变 fp32 (256/384 专家的 logits 显存与写带宽约翻倍, 但相对 KV cache 与专家权重属小量级)。系统侧: `matmul_persistent` 是确定性推理下被补丁的 `torch.mm / matmul` 等的统一后端, 本次新增的 `out_dtype` 参数改变了所有此类调用的输出分配语义, 属于共享基础设施层的契约调整, 会影响后续所有依赖 batch invariance 的模型与内核。团队侧: 确立了「输出 dtype 必须与累加 dtype 对齐, 禁止 compute-then-cast」的约定, 也为后续把确定性覆盖扩展到更多后端 / 模型提供了可复用的参数通道。- 风险标记: 依赖猴补丁语义, 共享内核输出契约变更, 缺少针对性测试, 非 CUDA 路径额外 fp32 拷贝

关联脉络

- PR #39366 fix: stop shadowing the DSpark shared-experts fusion guard: 同属 DeepSeek 系列模型的 MoE 前向路径修复: 一个是被重复定义的方法把融合守卫「shadow」掉, 一个是确定性分支把 dtype 悄悄降级, 都是模型 forward 里不易察觉的默认行为偏差。
- PR #39223 Fix MegaMoE buffer allocation and caching for effective SM budgets: 同样属于 MoE 内核缓冲区 / 分配语义的正确性修复 (buffer 分配与实际预算不一致), 与本 PR 修改 persistent matmul 输出缓冲分配方式的思路同源, 可作为内核契约类改动的对照参考。
- PR #39534 [Fix] Aggregate all IPC weight update responses: 被标注 consistency 的静默正确性修复, 与确定性 / 批次不变量这条主线 (deterministic 标签) 同属一致性保障工作, 可作为仓库近期一致性修复脉络的一部分。