PR #26929 完整报告

sgl-project/sglang

Add stochastic rounding for FP16 Mamba SSM cache

合并时间：2026-06-29 16:47

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26929

执行摘要

- 一句话：为 Mamba SSM 缓存添加随机舍入支持
- 推荐动作：值得关注的设计决策是随机舍入在两个后端中的统一抽象——通过 ssu_dispatch.py 的工厂函数将 SR 参数注入后端构造，使得上层调度器无需关心具体精度策略。同时，PR 中 Triton 内联 PTX 的用法为未来类似低精度运算提供了参考模式。但必须强调，PTX 操作数顺序的 bug 尚未确认是否已在合并前修复（代码 head 仍为错误顺序），建议尽快审查并修复该问题。

功能与动机

FP16 Mamba SSM 缓存在写入时默认使用简单截断，这会导致精度损失，尤其在长序列推理中误差累积明显。随机舍入是一种无偏的舍入策略，能够在统计意义上保持数值期望，降低量化噪声。本 PR 为 Mamba 模型部署提供了更灵活的精度控制选项，并由 #28695 在 SM100 上进一步验证。

实现拆解

1. CLI 参数扩展（python/sglang/srt/server_args.py）：新增 *enable_mamba_cache_stochastic_rounding*（bool，默认关闭）和 *mamba_cache_philox_rounds*（int，默认 0）两个参数，位于 Mamba 缓存配置节。
2. 配置校验增强（python/sglang/srt/server_args.py 中 _handle_mamba_backend 方法）：在校验后端可用性之前增加了对 Philox 轮数非负性检查，以及随机舍入启用后的三个前置条件——SSM dtype 必须为 float16、平台必须是 NVIDIA CUDA、若后端为 Triton 则必须是 SM100+。对于 FlashInfer 后端，在导入错误信息中追加随机舍入相关的提示。
3. Triton 内核改造（python/sglang/srt/layers/attention/mamba/ops/mamba_ssm.py）：新增 Triton JIT 函数 *convert_rs_fp16x2*，通过 PTX 指令 *cvt.rs.f16x2.f32* 实现一个 FP32 向量的随机舍入并打包为 FP16x2。在原先的 _selective_scan_update_kernel 中增加 USE_RS_ROUNDING 和 PHILOX_ROUNDS 模板参数，当启用时使用 Philox 随机数生成器和 *convert_rs_fp16x2* 执行舍入写入而非直接类型转换。
4. FlashInfer后端适配（python/sglang/srt/layers/attention/mamba/ops/ssu_dispatch.py）：TritonSSUBackend 和 FlashInferSSUBackend 的 __init__ 均新增 *enable_stochastic_rounding* 和 *cache_philox_rounds* 参数并存储为实例属性。调用 FlashInfer 的 *selective_state_update* 时，在启用随机舍入时生成一个随机的 *rand_seed* 并传递 philox_rounds（默认为 10）。

5. 测试与文档配套：在 test_server_args.py 中新增 TestMambaCacheStochasticRounding 类，覆盖非法配置的拒绝逻辑；在 test_mamba_ssm.py 中新增 test_selective_state_update_stochastic_rounding，参数化 philox_rounds、has_z、dstate、dim，验证随机舍入输出与参考实现一致。文档侧在部署脚本（nemotron3-ultra-deployment.jsx）和 server_arguments.mdx、Nemotron3-Ultra.mdx 中更新了参数说明。

关键文件：

- python/sglang/srt/server_args.py（模块 参数配置；类别 source；类型 core-logic）：核心参数定义与启动校验，新增两条 CLI 参数并在 _handle_mamba_backend 中增加了随机舍入的完整前置条件检查，是功能入口和防护屏障。
- test/registered/unit/server_args/test_server_args.py（模块 参数测试；类别 test；类型 test-coverage；符号 TestMambaCacheStochasticRounding, test_rejects_fp32_ssm_cache, test_rejects_non_cuda, test_rejects_triton_without_sm100）：新增 TestMambaCacheStochasticRounding 测试类，覆盖了三种非法配置场景，确保启动校验正确。
- test/registered/layers/mamba/test_mamba_ssm.py（模块 SSM 测试；类别 test；类型 test-coverage；符号 test_selective_state_update_stochastic_rounding）：新增 test_selective_state_update_stochastic_rounding 集成测试，参数化验证随机舍入下的 Triton 内核输出正确性。
- python/sglang/srt/layers/attention/mamba/ops/mamba_ssm.py（模块 Triton 内核；类别 infra；类型 infrastructure；符号 convert_rs_fp16x2）：Triton 内核中新增 convert_rs_fp16x2 内联 PTX 函数和随机舍入写入路径，是随机舍入算法的核心实现。
- python/sglang/srt/layers/attention/mamba/ops/ssu_dispatch.py（模块 后端调度；类别 infra；类型 infrastructure；符号 init）：后端分发层适配，两个 SSU 后端的 __init__ 接收 SR 参数并在调用时传递，同时 FlashInfer 后端生成随机种子。
- docs_new/src/snippets/autoregressive/nemotron3-ultra-deployment.jsx（模块 部署脚本；类别 source；类型 core-logic）：部署代码片段中添加了随机舍入配置选项，方便用户通过交互界面启用。
- docs_new/docs/advanced_features/server_arguments.mdx（模块 文档；类别 other；类型 core-logic）：服务器参数文档中添加了新参数的说明。
- docs_new/cookbook/autoregressive/NVIDIA/Nemotron3-Ultra.mdx（模块 文档；类别 other；类型 core-logic）：Cookbook 文档更新，显示随机舍入相关启动命令。

关键符号：convert_rs_fp16x2, selective_state_update, _handle_mamba_backend, TritonSSUBackend.init, FlashInferSSUBackend.init, initialize_mamba_selective_state_update_backend

关键源码片段

python/sglang/srt/server_args.py

核心参数定义与启动校验，新增两条 CLI 参数并在 _handle_mamba_backend 中增加了随机舍入的完整前置条件检查，是功能入口和防护屏障。

```
# python/sglang/srt/server_args.py — 新增参数定义（位于 Mamba 缓存配置节）

# 随机舍入开关，默认关闭
enable_mamba_cache_stochastic_rounding: A[
    bool,
    "启用 FP16 Mamba SSM 缓存写入的随机舍入。"
    " 需要 --mamba-ssm-dtype float16 和 CUDA。"
    " 当 --mamba-backend triton 时需要 SM100。"
] = False

# Philox 随机数生成轮数，0 表示使用后端默认值
mamba_cache_philox_rounds: A[
    int,
    "随机舍入的 Philox 轮数。"
    " Triton 默认使用 Triton 内置值，FlashInfer 默认 10。"
] = 0


def _handle_mamba_backend(self):
    # 首先校验 Philox 轮数非负
    if self.mamba_cache_philox_rounds < 0:
        raise ValueError("--mamba-cache-philox-rounds must be non-negative.")

    # 随机舍入前置条件检查
    if self.enable_mamba_cache_stochastic_rounding:
        # 1. 必须为 FP16 SSM 缓存
        if self.mamba_ssm_dtype != "float16":
            raise ValueError(
                "随机舍入需要 --mamba-ssm-dtype float16，"
                f"当前为 {self.mamba_ssm_dtype!r}。"
            )
        # 2. 仅支持 NVIDIA CUDA
        if not is_cuda():
            raise ValueError(
                "随机舍入仅支持 NVIDIA CUDA 平台。"
            )
        # 3. Triton 后端需要 SM100（支持 cvt.rs.f16x2.f32）
        if self.mamba_backend == "triton" and not is_sm100_supported():
            raise ValueError(
                "Triton 随机舍入需要 SM100 + CUDA >= 12.8。"
                "在 H100 上请改用 --mamba-backend flashinfer。"
            )

    # 以下为原有 FlashInfer 可用性校验，错误信息中追加随机舍入提示
    if self.mamba_backend == "flashinfer":
        flashinfer_error = "FlashInfer mamba 模块不可用，请检查安装。"
        if self.enable_mamba_cache_stochastic_rounding:
            flashinfer_error += (
                " FlashInfer 随机舍入需要 FlashInfer Mamba 和 FP16。"
```

```
        )
    # ... 原有导入并检查可用性的逻辑 ...
```

python/sglang/srt/layers/attention/mamba/ops/mamba_ssm.py

Triton 内核中新增 convert_rs_fp16x2 内联 PTX 函数和随机舍入写入路径，是随机舍入算法的核心实现。

```
# python/sglang/srt/layers/attention/mamba/ops/mamba_ssm.py — 随机舍入 Triton 内核

@triton.jit
def convert_rs_fp16x2(x: tl.tensor, rand: tl.tensor) -> tl.tensor:
    """使用 PTX cvt.rs.f16x2.f32 指令，将两个 FP32 元素随机舍入为 FP16x2 向量。
    注意：操作数顺序 $1 和 $2 分别对应 x 的两个元素，$3 为随机数种子。
    当前实现中 $1 和 $2 顺序与 Triton 的 pack=2 约定可能不匹配，存在潜在风险。
    """
    y = tl.inline_asm_elementwise(
        asm="""{
            cvt.rs.f16x2.f32 $0, $2, $1, $3;
        }",
        constraints="=r,r,r,r,r",
        args=(x, rand),
        dtype=tl.float16,
        is_pure=True,
        pack=2,
    )
    return y


@triton.heuristics({"HAS_DT_BIAS": lambda args: args["dt_bias_ptr"] is not None})
@triton.heuristics({"HAS_D": lambda args: args["D_ptr"] is not None})
@triton.heuristics({"HAS_Z": lambda args: args["z_ptr"] is not None})
# ... 其他装饰器 ...
def _selective_scan_update_kernel(..., USE_RS_ROUNDING: tl.constexpr, PHILOX_ROUNDS: tl.
constexpr):
    # ... 前面计算 state 的逻辑 ...
    if not DISABLE_STATE_UPDATE:
        if USE_RS_ROUNDING:
            rand_seed = tl.load(rand_seed_ptr)
            # 计算随机数偏移量（与线程索引对应）
            if HAS_STATE_BATCH_INDICES:
                rand_offsets = state_batch_idx * stride_state_batch + pid_h * stride_state_head
            else:
                rand_offsets = pid_b * stride_state_batch + pid_h * stride_state_head
            rand_offsets += offs_m[:, None] * stride_state_dim + offs_n[None, :] * stride_state_
            dstate
            if PHILOX_ROUNDS > 0:
                rand = tl.randint(rand_seed, rand_offsets, PHILOX_ROUNDS)
            else:
                rand = tl.randint(rand_seed, rand_offsets)
```

```
            state_to_store = convert_rs_fp16x2(state, rand)
            tl.static_assert(state_to_store.dtype == tl.float16, "state must be fp16")
            tl.static_assert(state_ptrs.dtype.element_ty == tl.float16, "仅支持 FP16 状态写入")
        else:
            state_to_store = state.to(state_ptrs.dtype.element_ty)
        tl.store(state_ptrs, state_to_store, mask=mask)
```

评论区精华

唯一的 review 评论来自 gemini-code-assist[bot]，指出 Triton JIT 函数 convert_rs_fp16x2 中的 PTX 指令操作数顺序存在严重正确性问题：cvt.rs.f16x2.f32 $0, $2, $1, $3; 交换了 $2 和 $1 导致相邻的 FP16 元素被错误放置，从而无声地损坏 Mamba SSM 缓存状态。评论建议修正为 $0, $1, $2, $3。然而该评论未被解决，PR 在得到 b8zhong 和 yuan-luo 的批准后合并，该潜在 bug 可能遗留在代码中。

- PTX 指令操作数顺序导致状态损坏 (correctness): 评论标记为 critical，但未得到修复或回应，PR 已合并。该问题可能仍存在。

风险与影响

- 风险：
  1. PTX 指令操作数错误：convert_rs_fp16x2 中操作数 $2 和 $1 顺序颠倒，可能导致状态缓存中相邻 FP16 元素被交换，引起推断精度下降甚至发散。虽然后续测试可能掩盖（test_selective_state_update_stochastic_rounding 使用宽容限 5e-3/1e-1），但该问题在真实场景可能显现。
  2. Triton 后端硬件限制：Triton 路径要求 SM100（Blackwell）架构，在旧 GPU 上启用随机舍入会直接报错，不存在静默退化风险。
  3. 性能开销：随机舍入引入 Philox 随机数生成和额外 PTX 指令，会增大写入延迟，但仅作用于 SSM 缓存写入（由 mamba_track_interval 控制间隔），非关键路径。
  4. FlashInfer 兼容性：FlashInfer 路径仅在 FlashInfer 的 mamba 模块可用时生效，启用随机舍入前已有明确校验。 - 影响：对用户而言，该功能默认关闭，不影响现有工作负载。开启后可提升 FP16 Mamba 缓存写入的数值质量，适用于对精度敏感的推理场景。对系统而言，仅增加两条 CLI 参数和少量启动验证逻辑，无运行时额外开销（未启用时完全不变）。测试覆盖了参数校验和随机舍入的核心路径，但功能测试依赖于 SM100 硬件，在 CI 中可能被跳过。文档更新帮助用户理解配置。 - 风险标记：PTX 指令顺序 bug，Triton 路径仅 SM100，Philox 随机数性能开销

关联脉络

- PR #28695 Stochastic rounding for Mamba SM100: 该 PR 的随机舍入方法是 #28695 在 SM100 上的后续实现，且由 yuan-luo 在 issue 评论中明确关联。