PR #26083 完整报告

sgl-project/sglang

Implement online nvfp4 quantization

合并时间：2026-06-10 15:26

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26083

执行摘要

- 一句话：为 Blackwell GPU 实现在线 NVFP4 MoE 量化
- 推荐动作：值得精读此 PR，了解如何将加载时量化和运行时激活缩放结合。推荐关注 NvFp4OnlineConfig 的设计（复用 ModelOpt 布局）、环境变量控制排除层、以及通过 temp_set_env 临时覆盖量化底层数学的实现模式。

功能与动机

FlashInfer TRTLLM MoE 现在支持运行时每 token 激活缩放（#22918），因此 SGLang 不再需要校准的静态激活 FP32 缩放。添加 nvfp4_online 接口允许在加载时从 BF16/FP16/FP8 检查点量化 MoE 权重，避免预量化检查点的需求。

实现拆解

1. 新增量化配置类：在 nvfp4_online.py 中创建 NvFp4OnlineConfig，继承自 ModelOptQuantConfig，标记 is_nvfp4_online=True，并支持通过环境变量 SGLANG_FP4_IGNORED_LAYERS 排除特定层。
2. 注册量化方法：在 server_args.py 中将 nvfp4_online 加入合法量化列表，并在 _handle_moe_kernel_config 中验证硬件为 Blackwell、MoE 后端为 FlashInfer TRTLLM，自动设置后端并禁用共享专家融合。
3. 模型加载适配：在 loader.py 的 load_weights_and_postprocess 中，当检测到 is_nvfp4_online 时，临时设置 TRTLLM_DISABLE_FP4_QUANT_FAST_MATH=1 以确保精确转换，加载后同步 CUDA 并清空缓存。
4. MoE 方法扩展：在 modelopt_quant.py 中复用 ModelOptFp4Config 的参数布局，扩展 ModelOptNvFp4FusedMoEMethod 以支持在线权重加载：对 BF16/FP16 直接量化，对 FP8 检查点先反量化再量化；运行时通过 use_per_token_activation=True 让 FlashInfer 计算每 token 激活缩放。
5. 测试与文档：在 test_flashinfer_trtllm_gen_moe_backend.py 中新增测试基类，使用 Qwen3-Next 模型运行 GSM8K 评估。同时更新文档（quantization.mdx、environment_variables.mdx、server_arguments.mdx）。

关键文件：

- python/sglang/srt/layers/quantization/nvfp4_online.py（模块 量化层；类别 source；类型 core-logic；符号 NvFp4OnlineConfig, _normalize_ignored_layers, init, get_name）：

核心新增文件，实现 NVFP4 在线量化配置和 MoE 量化方法。

- test/registered/backends/test_flashinfer_trtllm_gen_moe_backend.py（模块 测试；类别 test；类型 test-coverage；符号 FlashinferTrtllmGenMoeBackendNvFp4OnlineBase, setUpClass, tearDownClass, test_gsm8k）：新增 NVFP4 在线量化集成测试，验证 GSM8K 准确率。
- python/sglang/srt/layers/quantization/modelopt_quant.py（模块 量化层；类别 source；类型 data-contract；符号 ModelOptFp4Config, create_weights, get_online_weight_loader）：修改现有 MoE 量化方法，支持在线 NVFP4 权重加载和层排除。
- python/sglang/srt/model_loader/loader.py（模块 模型加载；类别 source；类型 core-logic；符号 load_weights_and_postprocess）：修改模型加载流程，为 nvfp4_online 设置临时环境变量并清理缓存。
- python/sglang/srt/server_args.py（模块 服务配置；类别 source；类型 configuration；符号 _handle_moe_kernel_config）：注册 nvfp4_online 为合法量化方法，并添加硬件和后端验证。
- python/sglang/srt/configs/model_config.py（模块 模型配置；类别 source；类型 configuration；符号 _verify_quantization）：添加 nvfp4_online 到量化验证映射，允许从 FP8 检查点加载。
- python/sglang/srt/layers/moe/moe_runner/flashinfer_trtllm.py（模块 MoE 驱动；类别 source；类型 core-logic）：调整 FlashInfer TRTLLM MoE runner 以兼容 nvfp4_online。
- python/sglang/srt/layers/quantization/__init__.py（模块 量化层；类别 source；类型 dependency-wiring）：注册 NvFp4OnlineConfig 到量化配置注册表。
- python/sglang/srt/environ.py（模块 环境管理；类别 source；类型 configuration）：添加 SGLANG_FP4_IGNORED_LAYERS 环境变量定义。
- docs_new/docs/advanced_features/quantization.mdx（模块 文档；类别 docs；类型 documentation）：添加 nvfp4_online 量化方法的文档。
- docs_new/docs/references/environment_variables.mdx（模块 文档；类别 docs；类型 documentation）：添加 SGLANG_FP4_IGNORED_LAYERS 环境变量参考。
- docs_new/docs/advanced_features/server_arguments.mdx（模块 文档；类别 docs；类型 documentation）：更新 --quantization 参数的可用选项。

关键符号：NvFp4OnlineConfig.init, NvFp4OnlineConfig._normalize_ignored_layers, NvFp4OnlineConfig.from_config, FlashinferTrtllmGenMoeBackendNvFp4OnlineBase.setUpClass, FlashinferTrtllmGenMoeBackendNvFp4OnlineBase.test_gsm8k, DefaultModelLoader.load_weights_and_postprocess, ServerArgs._handle_moe_kernel_config, ModelConfig._verify_quantization, ModelOptFp4Config.create_weights

关键源码片段

python/sglang/srt/layers/quantization/nvfp4_online.py

核心新增文件，实现 NVFP4 在线量化配置和 MoE 量化方法。

```
class NvFp4OnlineConfig(ModelOptQuantConfig):
    """Config for `--quantization nvfp4_online`.

    This mode is a load-time conversion path, not a serialized NVFP4 checkpoint
    format. It reuses the ModelOpt NVFP4 MoE parameter layout and fills those
    parameters by converting BF16/FP16/FP8 expert tensors as they are loaded.
    Dense layers stay in the source checkpoint precision or quantization path.
    """

    is_nvfp4_online = True # 标记为在线模式，被模型加载器识别
    is_checkpoint_nvfp4_serialized = False
    group_size = 16

    @staticmethod
    def _normalize_ignored_layers(ignored_layers: Optional[List[str]]) -> List[str]:
        # 规范化排除层名称，同时处理带 / 不带 "model." 前缀
        if not ignored_layers:
            return []
        normalized = []
        for layer in ignored_layers:
            base = layer.removeprefix("model.")
            normalized.append(base)
            normalized.append(f"model.{base}")
        return list(dict.fromkeys(normalized))

    def __init__(
        self,
        exclude_modules: Optional[List[str]] = None,
        packed_modules_mapping: Optional[Dict[str, List[str]]] = None,
        is_checkpoint_fp8_serialized: bool = False,
        activation_scheme: str = "dynamic",
        weight_block_size: Optional[List[int]] = None,
        use_mxfp8: bool = False,
    ) -> None:
        source_ignored_layers = self._normalize_ignored_layers(exclude_modules)
        fp4_ignored_layers = list(source_ignored_layers)
        # 从环境变量 SGLANG_FP4_IGNORED_LAYERS 读取额外的排除层
        if ignored_layers_str := envs.SGLANG_FP4_IGNORED_LAYERS.get():
            fp4_ignored_layers.extend(
                layer.strip() for layer in ignored_layers_str.split(",") if layer.strip()
            )
        fp4_ignored_layers = self._normalize_ignored_layers(fp4_ignored_layers)
        super().__init__(
            kv_cache_quant_algo=None,
            exclude_modules=source_ignored_layers,
            packed_modules_mapping=packed_modules_mapping or {},
        )
        self.fp4_ignored_layers = fp4_ignored_layers
        # 权重使用静态 NVFP4 缩放，激活使用运行时每 token FP32 缩放
```

```
        self.use_per_token_activation = True
        self.is_checkpoint_fp8_serialized = is_checkpoint_fp8_serialized
        self.is_fp4_experts = False
        self.activation_scheme = activation_scheme
        self.weight_block_size = weight_block_size
        self.use_mxfp8 = use_mxfp8
```

test/registered/backends/test_flashinfer_trtllm_gen_moe_backend.py

新增 NVFP4 在线量化集成测试，验证 GSM8K 准确率。

```
class FlashinferTrtllmGenMoeBackendNvFp4OnlineBase:
    backend = None
    extra_env = {}

    @classmethod
    def setUpClass(cls):
        cls.model = "Qwen/Qwen3-Next-80B-A3B-Instruct-FP8"
        cls.base_url = DEFAULT_URL_FOR_TEST
        # 启动服务器时指定 --quantization nvfp4_online 和必要的 env 变量
        cls.process = popen_launch_server(
            cls.model,
            cls.base_url,
            timeout=DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH,
            env={**os.environ, **cls.extra_env, "SGLANG_ENABLE_JIT_DEEPGEMM": "False"},
            other_args=[
                "--attention-backend", "triton",
                "--moe-runner-backend", cls.backend,
                "--cuda-graph-max-bs", "128",
                "--tp-size", "4",
                "--ep-size", "2",
                "--quantization", "nvfp4_online",
                "--mem-fraction-static", "0.7",
                "--mamba-ssm-dtype", "bfloat16",
            ],
        )

    @classmethod
    def tearDownClass(cls):
        kill_process_tree(cls.process.pid)

    def test_gsm8k(self):
        args = SimpleNamespace(
            base_url=self.base_url,
            model=self.model,
            eval_name="gsm8k",
            api="completion",
            max_tokens=512,
            num_examples=200,
            num_threads=128,
```

```
    )
    metrics = run_eval(args)
    # 设定准确率阈值 0.90，确保 NVFP4 量化质量
    self.assertGreater(metrics["score"], 0.90)
```

评论区精华

1. 命名争议：Edwardf0t1 指出 per_token_nvfp4 中对权重来说不是 per-token，建议更名。作者最终改为 nvfp4_online，反映了加载时转换而非运行时每 token 的本质。
2. 代码组织：b8zhong 建议将 online FP4 量化移动到单独文件，避免 modelopt_quant.py 过度膨胀。作者执行了拆分。
3. 数值正确性：gemini-review 指出 nvfp4_quantize 返回值解包和 nvfp4_max 计算可能错误。作者澄清了意图并增加注释加以说明。
4. 性能优化：gemini-review 建议避免不必要的 FP32 转换、使用 Python 标量计算 scale。作者进行了优化。
5. 线程安全：关于 temp_set_env 修改环境变量可能非线程安全。作者承认局限并限制作用域到模型加载阶段。

- 量化接口命名争议 (design): 作者采纳建议，最终改为 nvfp4_online。
- 代码组织建议 (design): 作者执行拆分，创建了独立的 nvfp4_online.py。
- 数值正确性 (correctness): 作者澄清了意图，nvfp4_quantize 在非 per-token 模式下只返回两个值；nvfp4_max 的 factor 是期望的，并增加了注释。
- 性能优化 (performance): 作者优化了 _weight_amax 直接在原始张量上计算，并在 _weight_scale_2_from_amax 中使用 Python 标量计算。
- 环境变量线程安全性 (correctness): 作者承认局限性，但指出 FlashInfer 目前只提供环境变量控制；该操作限制在模型加载阶段，使用类级别锁保护。
- FP8 检查点映射原因 (question): 作者解释：允许从 FP8 检查点加载，SGLang 会反量化再重新量化到 NVFP4。

风险与影响

- 风险：
  1. 核心路径变更：模型加载流程中增加了条件分支，可能影响已有量化路径的稳定性。
  2. 硬件限制：仅支持 Blackwell (SM100/SM103) GPU，其他 GPU 上会报错退出。
  3. 数值精度：NVFP4 量化可能对模型质量产生不可逆影响；测试显示 GSM8K 精度可接受，但其他任务未验证。
  4. 线程安全性：加载时设置环境变量 TRTLLM_DISABLE_FP4_QUANT_FAST_MATH 影响进程全局，在并发请求下可能被观测到。
  5. 兼容性：仅支持 FlashInfer TRTLLM MoE 后端，与 Triton MoE 后端不兼容。 - 影响：对用户：提供了一种简便的在线量化方式，无需预量化检查点即可在 Blackwell GPU 上减少 MoE 权重内存占用。对系统：加载时转换会增加初始启动延迟，但避免了保留未量化权重副本，降低峰值内存。对团队：维护新的量化代码路径，需要同步 FlashInfer 的相关演进。 - 风险标记：核心路径变更，硬件限制 (Blackwell)，数值精度风险，线程安全

性

关联脉络

- 暂无明显关联 PR