PR #34859 完整报告

sgl-project/sglang

Qwen3.8-27B Model Support

合并时间：2026-08-19 16:31

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34859

执行摘要

- 一句话：新增 Qwen3.8-27B 支持，配套 FP4/FP8 GEMV 与修复
- 推荐动作：值得精读。重点关注三点：JIT GEMV kernel 的 dispatch predicate 设计（如何把收益集中在 DRAM 带宽受限的形状并安全回退 cuBLAS）；FP4 融合中 scale 的 6-D swizzle 重排逻辑（极易出错但测试仅覆盖部分 batch）；以及通过布尔标志 + 外部注入方式把融合能力局部化到 MLP / down_proj 的改动风格。

功能与动机

PR body 仅一句话 Add Qwen3.8-27B model support。squash 提交信息展开的动机包括：启动 Qwen3.8-27B 模型支持；将 GDN prefill 路径的 cu_seqlens 转 int64；扩大 SM120 FP8 blockwise GEMM 的 swapAB 分派范围；修复 DSpark 在目标 lm_head 量化时的 draft logits；以及融合 SiLU 与 FP4 量化 kernel，减少 dense MLP 在 decode 时的 kernel 启动与访存开销。

实现拆解

1. 模型接入与注意力布局扩展：python/sglang/srt/models/qwen3_5.py 中模块级常量 _GDN_FUSED_QKVZBA_RATIOS 从 (1, 2, 4) 扩展为 CUDA 下 (1, 2, 3, 4)，CPU 保持 (1, 2, 4)，aiter 保持 (1, 2, 4, 8)，因为 Qwen3.8-27B 这类 dense hybrid 模型的 v/k head 比例为 3（非 2 的幂），需要 Triton kernel 的 per-head walk 支持；python/sglang/srt/layers/attention/linear/gdn_backend.py 与 python/sglang/srt/layers/attention/linear/kernels/gdn_flashinfer.py 相应调整控制流以适配 ratio 3。
2. 量化融合：qwen3_5.py 新增 _maybe_enable_silu_fp4_quant_fusion()，在 gate_up_proj 与 down_proj 均为 ModelOptFp4LinearMethod 且 flashinfer 可导入时启用融合，并提供 SGLANG_DISABLE_SILU_FP4_QUANT_FUSION kill switch；python/sglang/srt/models/qwen2_moe.py 新增 _silu_fp4_quant_fused() 完成 FlashInfer 融合 kernel 的调用与 scale 6-D swizzle 重排，同时为 down_proj 打上 _accepts_prequantized_fp4 标记，forward 中根据标志选择融合路径。
3. JIT GEMV kernel：新增 python/sglang/kernels/ops/gemm/hopper_bf16_gemv.py（SM90、bf16、fp32 累加）与 python/sglang/kernels/ops/gemm/sm120_fp8_gemv.py（SM120、FP8），均通过 cache_once + load_jit 按形状编译；use_hopper_bf16_gemv / use_sm120_fp8_gemv 的 predicate 严格限定 m==1 与 shape 范围，其余回退 cuBLAS。

python/sglang/srt/layers/quantization/modelopt_quant.py 的 FP8 apply 路径接入 SM120 快速路径并在权重加载后派生 sm120_gemv_alpha 组合 scale；python/sglang/srt/layers/quantization/unquant.py 增加 is_gemv 兼容。

4. 修复与配套：python/sglang/srt/models/dspark.py 与 python/sglang/srt/speculative/dspark_components/dspark_draft_sampler.py 修复量化 lm_head 下的 logits dtype 传递；GDN prefill 路径将 cu_seqlens 转 int64；SM120 fp8 blockwise GEMM 扩大 swapAB 分派范围。测试新增 test/registered/attention/test_gdn_fused_split_head_ratios.py 与 test/registered/gemm/test_hopper_bf16_gemv.py，注册到 base-b CI（1-gpu-large）。

关键文件：

- python/sglang/srt/models/qwen3_5.py（模块 模型接入；类别 source；类型 data-contract；符号 _maybe_enable_silu_fp4_quant_fusion, _GDN_FUSED_QKVZBA_RATIOS）：模型接入入口：扩展 _GDN_FUSED_QKVZBA_RATIOS 支持 CUDA ratio 3，新增 _maybe_enable_silu_fp4_quant_fusion 融合开关，并在模型层构建时启用。
- python/sglang/srt/models/qwen2_moe.py（模块 模型层；类别 source；类型 data-contract；符号 _silu_fp4_quant_fused, forward）：FP4 融合的实际执行端：新增 _silu_fp4_quant_fused 方法，调用 FlashInfer 融合 kernel 并完成 scale swizzle 重排，forward 增加融合分支。
- python/sglang/kernels/ops/gemm/hopper_bf16_gemv.py（模块 JIT 内核；类别 infra；类型 infrastructure；符号 _config, _jit_hopper_bf16_gemv_module, use_hopper_bf16_gemv, hopper_bf16_gemv）：新增的 SM90 bf16 GEMV JIT kernel，含 dispatch predicate（use_hopper_bf16_gemv）与 H200 调参配置，是 decode 性能优化的核心。
- python/sglang/kernels/ops/gemm/sm120_fp8_gemv.py（模块 JIT 内核；类别 infra；类型 infrastructure；符号 _config, _jit_sm120_fp8_gemv_module, use_sm120_fp8_gemv, sm120_fp8_gemv）：与 Hopper GEMV 配套的 SM120 FP8 GEMV JIT kernel，服务于 Blackwell 平台 FP8 权重的 decode 快速路径。
- python/sglang/srt/layers/quantization/modelopt_quant.py（模块 量化层；类别 source；类型 data-contract；符号 ModelOptFp8LinearMethod, apply, process_weights_after_loading）：FP8 量化线性层接入 SM120 GEMV 快速路径，并在权重加载后派生 sm120_gemv_alpha 组合 scale。
- test/registered/attention/test_gdn_fused_split_head_ratios.py（模块 测试；类别 test；类型 test-coverage；符号 _reference_split, TestGdnFusedSplitHeadRatios, _run_ratio, test_ratio_1）：新增测试覆盖 GDN 融合 split kernel 在 ratio 1/2/3/4 下的 bitwise 正确性，其中 ratio 3 是本 PR 新支持的布局。
- test/registered/gemm/test_hopper_bf16_gemv.py（模块 测试；类别 test；类型 test-coverage；符号 _is_sm90, TestHopperBf16Gemv, _run_case, test_dispatch_domain_shapes）：新增测试覆盖 Hopper bf16 GEMV 在 dispatch 域形状、tail rows 与 predicate 回退条件上的正确性。

关键符号：_maybe_enable_silu_fp4_quant_fusion, _silu_fp4_quant_fused, hopper_bf16_gemv, use_hopper_bf16_gemv, sm120_fp8_gemv, use_sm120_fp8_gemv

关键源码片段

python/sglang/srt/models/qwen3_5.py

模型接入入口：扩展 _GDN_FUSED_QKVZBA_RATIOS 支持 CUDA ratio 3，新增 _maybe_enable_silu_fp4_quant_fusion 融合开关，并在模型层构建时启用。

```
# 文件：python/sglang/srt/models/qwen3_5.py
def _maybe_enable_silu_fp4_quant_fusion(mlp: nn.Module) -> None:
    """把 SiLU + mul 与 down_proj 的 NVFP4 输入量化合成一个 FlashInfer kernel，
    并让 down_proj 直接接收预量化好的 (fp4, scale) 元组，减少 decode 路径 kernel 启动。"""
    # 提供环境变量 kill switch，便于线上快速回退到原两条 kernel 路径
    if os.environ.get("SGLANG_DISABLE_SILU_FP4_QUANT_FUSION", "0") == "1":
        return
    from sglang.srt.layers.quantization.modelopt_quant import ModelOptFp4LinearMethod

    # 只有 gate_up_proj 与 down_proj 均使用 NVFP4 (W4A4) 线性方法时才融合
    if not (
        isinstance(mlp.gate_up_proj.quant_method, ModelOptFp4LinearMethod)
        and isinstance(mlp.down_proj.quant_method, ModelOptFp4LinearMethod)
    ):
        return
    try:
        from flashinfer import silu_and_mul_scaled_nvfp4_experts_quantize # noqa: F401
    except ImportError:
        # flashinfer 版本较旧时保持原行为，不阻断模型加载
        return
    # 通过两个标志位把融合能力告知 MLP 与 down_proj，改动局部化：
    # forward 根据 _enable_silu_fp4_quant_fusion 选择融合路径；
    # down_proj 根据 _accepts_prequantized_fp4 接收元组输入。
    mlp._enable_silu_fp4_quant_fusion = True
    mlp.down_proj._accepts_prequantized_fp4 = True
    logger.info("Enabled fused SiLU+mul+FP4-quant for dense MLP down_proj input.")
```

python/sglang/srt/models/qwen2_moe.py

FP4 融合的实际执行端：新增 _silu_fp4_quant_fused 方法，调用 FlashInfer 融合 kernel 并完成 scale swizzle 重排，forward 增加融合分支。

```
# 文件：python/sglang/srt/models/qwen2_moe.py（Qwen2MoeMLP 节选）
def _silu_fp4_quant_fused(self, gate_up: torch.Tensor) -> tuple:
    # FlashInfer 融合 kernel：SiLU + mul + NVFP4 量化一步完成，
    # 替代原来的 act_fn 加 per-token FP4 量化两个 kernel。
    from flashinfer import silu_and_mul_scaled_nvfp4_experts_quantize

    # input_scale_inv 在权重加载后才存在，因此这里惰性缓存为 1-D 全局 scale
    if self._down_input_scale_inv_1d is None:
        self._down_input_scale_inv_1d = self.down_proj.input_scale_inv.reshape(1)
```

```
    num_tokens = gate_up.shape[0]
    masked_m = self._masked_m_cache.get(num_tokens)
    if masked_m is None:
        # masked_m 只与 token 数相关，按 batch 大小缓存避免重复建 tensor
        masked_m = torch.tensor(
            [num_tokens], dtype=torch.int32, device=gate_up.device
        )
        self._masked_m_cache[num_tokens] = masked_m

    y_fp4, y_sf = silu_and_mul_scaled_nvfp4_experts_quantize(
        gate_up.unsqueeze(0),
        masked_m,
        self._down_input_scale_inv_1d,
    )
    # [M, K/2, 1] -> [M, K/2]；scale 是 expert-grouped 的 6-D swizzle
    # 布局 (32, 4, m_blocks, 4, K/64, 1)，需重排成 fp4_gemm 期望的
    # dense swizzled 布局（已与 fp4_quantize 对比验证到 FP4 舍入 tie）。
    y_fp4 = y_fp4.squeeze(-1).view(torch.uint8)
    m_padded = y_sf.shape[2] * y_sf.shape[0] * y_sf.shape[3]
    y_sf = y_sf.view(torch.uint8).permute(2, 4, 0, 1, 3, 5).reshape(m_padded, -1)
    return y_fp4, y_sf


def forward(self, x):
    gate_up, _ = self.gate_up_proj(x)
    # 融合只对非元组输出（未走其他量化分支）生效
    if self._enable_silu_fp4_quant_fusion and not isinstance(gate_up, tuple):
        x, _ = self.down_proj(self._silu_fp4_quant_fused(gate_up))
        return x
    x = self.act_fn(gate_up)
    x, _ = self.down_proj(x)
    return x
```

python/sglang/kernels/ops/gemm/hopper_bf16_gemv.py

新增的 SM90 bf16 GEMV JIT kernel，含 dispatch predicate（use_hopper_bf16_gemv）与 H200 调参配置，是 decode 性能优化的核心。

```
# 文件：python/sglang/kernels/ops/gemm/hopper_bf16_gemv.py
# bs=1 的 decode 是纯权重流式（weight-streaming）负载，cuBLAS 在中等 N 上
# 只能达到 50-70% 的 DRAM 带宽（H200 实测 3.2-3.9 TB/s vs 4.3 TB/s 拷贝上限）。
# 设计：一个 warp 计算连续几行，激活向量放共享内存，权重用 evict-first 加载。

_MAX_K = 17408 # 静态 smem 上限（K * 2 字节 <= 48KB）并留余量
_MAX_N = 65536 # 超大 N（如 lm_head）本身已达带宽上限，无需 GEMV


def _config(n: int) -> tuple[int, int, int]:
    """(rows_per_warp, k_unroll, num_warps)，H200 上实测调参。"""
    if n >= 8192:
```

```
        return (2, 2, 8)
    return (1, 2, 8)


@cache_once
def _jit_hopper_bf16_gemv_module(n: int, k: int) -> Module:
    # 按 n/k 编译专用 kernel，参数编译进模板，避免运行时分支
    rows, unroll, warps = _config(n)
    args = make_cpp_args(n, k, rows, unroll, warps)
    return load_jit(
        "hopper_bf16_gemv",
        *args,
        cuda_files=["gemm/hopper_bf16_gemv.cuh"],
        cuda_wrappers=[("run", f"sglang::HopperBf16GemvKernel<{args}>::run")],
        extra_cuda_cflags=["-O3"],
    )


def use_hopper_bf16_gemv(m: int, n: int, k: int) -> bool:
    # 只接管 m==1 且形状对齐的 decode GEMM，其余一律回退 cuBLAS
    if not (
        m == 1
        and k % 512 == 0
        and 512 <= k <= _MAX_K
        and n % 8 == 0
        and 64 <= n <= _MAX_N
    ):
        return False
    # cuBLAS 在 N≈16K 时已接近 3.9 TB/s，收益集中在中小 N 与超宽 N
    return n < 12288 or n >= 32768


def hopper_bf16_gemv(x: torch.Tensor, w: torch.Tensor) -> torch.Tensor:
    """y[1, N] = x[1, K] @ w[N, K]^T，全 bf16、fp32 累加。"""
    out = torch.empty((1, w.shape[0]), dtype=x.dtype, device=x.device)
    module = _jit_hopper_bf16_gemv_module(w.shape[0], w.shape[1])
    module.run(x, w, out)
    return out
```

评论区精华

该 PR 的 review 评论为空，BBuf 直接 APPROVED 后合并。Issue 区仅保留两条自动化 / 确认类评论：作者 yhyang201 触发 /tag-and-rerun-ci 重跑 CI；合并者 BBuf 确认 Merged with all ci green 并附 CI 运行链接。因此没有留下关于设计权衡的实质讨论。

- CI 重跑与合并确认 (other): PR 在 base 与 extra 两套 CI 均通过后由 BBuf 合并，无遗留技术讨论。

风险与影响

- 风险：GEMV 新 kernel 只在限定 shape（m==1、特定 N/K 范围）生效，predicate 之外全部回退 cuBLAS，回归面可控；但 JIT 编译首次调用有额外延迟，且 hopper_bf16_gemv 依赖 SM90、sm120_fp8_gemv 依赖 SM120，跨硬件行为依赖 predicate 正确性。FP4 融合依赖 flashinfer 的 silu_and_mul_scaled_nvfp4_experts_quantize，虽有 ImportError guard 与 SGLANG_DISABLE_SILU_FP4_QUANT_FUSION kill switch，但 qwen2_moe.py 中手工 permute 的 scale swizzle 逻辑较脆弱，测试仅覆盖部分 M 范围。DSpark dtype 修复影响投机解码一致性，若 _base_logits_dtype 处理不当可能改变采样结果。GDN ratio 3 仅 CUDA 启用，CPU/AMD 路径不变，但新增的 CUDA 专属分支需要持续维护。
- 影响：对用户：可直接以 SGLang 服务 Qwen3.8-27B，decode 在 Hopper 与 Blackwell SM120 上获得 GEMV 加速，NVFP4 权重场景下 dense MLP 少一次激活量化 kernel。对系统：新增两个 JIT kernel 编译入口，首次调用有编译延迟；FP4 融合路径依赖 flashinfer 版本。对团队：为后续 dense hybrid 模型（非 2 的幂 head ratio）提供了可复用的接入模式与测试范式，也验证了按 SM 分派的 GEMV 优化思路。
- 风险标记：新增 JIT kernel 依赖硬件条件与形状匹配，FP4 融合依赖 flashinfer 且含手工 swizzle，DSpark dtype 修复影响投机解码，CUDA 专属 ratio 3 路径

关联脉络

- PR #34953 [Perf] Restore the 16-token router GEMM threshold on SM10X: 同属 GEMM/GEMV dispatch 阈值调优线，均针对特定 SM 的 kernel 选择做微调。
- PR #34680 [diffusion][Minimax H3]support subblock sparse attention on SM90: 同为在 SM90 上扩展 JIT kernel 能力的变更，体现仓库自定义 kernel 的演进趋势。