PR #33655 完整报告

sgl-project/sglang

[diffusion] Prefer cuDNN SDPA over FA4 for dense attention on sm_100 (B200)

合并时间：2026-08-06 08:49

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33655

执行摘要

- 一句话：B200 扩散模型默认切 cuDNN SDPA，比 FA4 快 1.3-1.5 倍
- 推荐动作：值得精读。该 PR 是一个「以数据驱动改变默认值」的优秀范例：先修正外部结论（FA2 baseline → FA4 baseline），再用 11 个真实 shape 的 microbenchmark 和 9 轮端到端 A/B 支撑决策；同时通过 fail-safe 链、NVFP4 数值保护和单测把变更风险压到最小。值得关注的设计决策包括：sm_100 专用 gate 与 Hopper 启发式并存、_cudnn_failed 按层锁存避免重复探测、return_softmax_lse 强制走 FA 以兼容 ring attention、以及加载期 context manager 实现量化默认后端的局部覆盖。

功能与动机

PR body 指出 NVlabs Sana sol-engine 分支在 sm_100 上用 cuDNN SDPA 替换 flash-attn 2.8.3 varlen 拿到 1.81x 端到端收益，但该结论基于 FA2 baseline，不直接适用于 sglang：sglang 在 sm_100 上默认分发的是 vendored FA4 CuTe DSL 内核。作者在 B200（torch 2.11.0+cu130、cuDNN 9.19）上重新实测，发现 FA4 虽已快于 FA2 体系，cuDNN 9.19 SDPA 仍比 FA4 快 1.24-1.98x（跨 Wan2.2、LingBot-World、MiniMax-H3、Qwen-Image、FLUX 全部真实形状），因此现有 sm_100 默认分发是次优的，需要修正默认值。

实现拆解

该 PR 的核心是一次「仅针对 sm_100 默认值」的分发调整，不触碰任何 kernel 代码，配套了量化数值保护和测试。实现按以下步骤推进：

1. 默认后端解析入口改造（python/sglang/multimodal_gen/runtime/platforms/cuda.py）：在 CudaPlatform.get_attn_backend_cls_str 的 selected_backend is None 分支里，当自动解析结果为 AttentionBackendEnum.FA 且 cls.is_blackwell()（compute capability 10.x）时，先调用 _resolve_flash_attention_backend_cls_str 确认 FA 实际可用（head size、dtype 等 guard 不变）；若 FA 可用则返回 _DYNAMIC_CUDNN_SDPA_BACKEND_CLS_STR，若 FA 不可用（会落到 Torch SDPA）则保持原 Torch SDPA 路径。显式 --attention-backend 选择不受影响。
2. 运行时双实现调度（python/sglang/multimodal_gen/runtime/layers/attention/backends/sdpa.py）：DynamicCudnnSDPAImpl 新增 _is_sm100（get_device_capability()[0] == 10）和 _cudnn_failed 状态锁存；_use_cudnn_sdpa 在 sm_100 上对非 causal、CUDA、fp16/bf16、Sq == Skv 的 dense 形状直接返回 True（同时支持 cross-attn，Skv 为文本长度），非 sm_100 保留原有 Hopper 启发式（D=64、S=1024、B>=4）；forward 增加

**kwargs，当调用方要求 return_softmax_lse（如 ring attention）时强制走 FA，并对 cuDNN 抛出的 RuntimeError 做一次 warning 后永久钉回 FA，避免每个 step 重复探测失败 kernel。

3. ModelOpt NVFP4 数值保护（transformer_load_utils.py + transformer_loader.py）：TransformerQuantLoadSpec 新增 is_modelopt_fp4 property；transformer_loader 新增 _default_quantized_attention_backend，在 Blackwell + ModelOpt FP4 且没有任何显式 / 全局 / 组件级 attention 后端时返回 AttentionBackendEnum.FA，并在 load_customized 中通过 component_attn_backend_context_manager 把该默认包在 maybe_load_fsdp_model 构造期周围（无默认时用 nullcontext）。这是为了防止新的 cuDNN 默认路径改变 NVFP4 量化模型的数值行为。
4. 测试配套：test_cuda_attention_backend.py 新增 test_default_backend_prefers_dynamic_cudnn_sdpa_on_blackwell，断言 Blackwell 默认解析出 DynamicCudnnSDPABackend 类名；test_transformer_quant.py 新增两个用例分别验证 FP4 默认走 FA、显式后端不被覆盖；test_glm_image_ar.py 引入 _FakeBatchResponse 以匹配 GLM-Image AR 批式返回新契约，test_glm_image_multi_output.py 补充 extra={} 字段并适配 generate_prior_tokens 三元组返回。另在 PR body 中提供了 B200 上的 microbenchmark、9 轮端到端 A/B 和 PSNR/SSIM 质量对比数据作为验证。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/attention/backends/sdpa.py（模块 注意力后端；类别 source；类型 core-logic；符号 DynamicCudnnSDPAImpl.init, DynamicCudnnSDPAImpl._use_cudnn_sdpa, DynamicCudnnSDPAImpl.forward）：sm_100 默认路径的实际运行时实现：新增 _is_sm100 判断、_cudnn_failed 锁存、sm_100 dense 形状直通 cuDNN SDPA，并处理 return_softmax_lse 与 cuDNN 异常兜底。
- python/sglang/multimodal_gen/runtime/platforms/cuda.py（模块 平台选择器；类别 source；类型 core-logic；符号 CudaPlatform.get_attn_backend_cls_str）：默认 attention 后端的解析入口：auto-selection 命中 FA 且平台为 Blackwell 时改选 DYNAMIC_CUDNN_SDPA，显式选择不受影响。
- python/sglang/multimodal_gen/runtime/loader/component_loaders/transformer_loader.py（模块 模型加载；类别 source；类型 core-logic；符号 _default_quantized_attention_backend, load_customized）：新增 _default_quantized_attention_backend 并用 attention backend context manager 包裹 FSDP 模型构造，为 ModelOpt NVFP4 保留 FA4 数值路径。
- python/sglang/multimodal_gen/runtime/loader/transformer_load_utils.py（模块 量化加载；类别 source；类型 core-logic；符号 TransformerQuantLoadSpec.is_modelopt_fp4）：TransformerQuantLoadSpec 新增 is_modelopt_fp4 property，为量化后端默认决策提供统一判定入口。
- python/sglang/multimodal_gen/test/unit/test_transformer_quant.py（模块 量化测试；类别 test；类型 test-coverage；符号 test_modelopt_fp4_uses_fa_by_default_on_blackwell, test_modelopt_fp4_preserves_explicit_attention_backend）：新增两个单元测试覆盖 Blackwell FP4 默认走 FA 与显式 attention_backend 不被覆盖两条关键分支。

- python/sglang/multimodal_gen/test/unit/test_cuda_attention_backend.py（模块 后端测试；类别 test；类型 test-coverage；符号 test_default_backend_prefers_dynamic_cudnn_sdpa_on_blackwell）：验证 Blackwell 上默认后端解析为 DynamicCudnnSDPABackend，防止未来分发逻辑回归。
- python/sglang/multimodal_gen/test/unit/test_glm_image_ar.py（模块 AR 测试；类别 test；类型 test-coverage；符号 _FakeBatchResponse, test_srt_ar_forward_aggregates_usage）：随 GLM-Image AR usage 契约更新：新增 _FakeBatchResponse 并调整聚合测试，属于本 PR 的配套测试修复。
- python/sglang/multimodal_gen/test/unit/test_glm_image_multi_output.py（模块 多输出测试；类别 test；类型 test-coverage）：适配 generate_prior_tokens 三元组返回与 batch.extra 字段的契约变化。

关键符号：_default_quantized_attention_backend, TransformerQuantLoadSpec.is_modelopt_fp4, CudaPlatform.get_attn_backend_cls_str, DynamicCudnnSDPAImpl.init, DynamicCudnnSDPAImpl._use_cudnn_sdpa, DynamicCudnnSDPAImpl.forward

关键源码片段

python/sglang/multimodal_gen/runtime/layers/attention/backends/sdpa.py

sm_100 默认路径的实际运行时实现：新增 _is_sm100 判断、_cudnn_failed 锁存、sm_100 dense 形状直通 cuDNN SDPA，并处理 return_softmax_lse 与 cuDNN 异常兜底。

```
# DynamicCudnnSDPAImpl 是 sm_100 上默认 attention 路径的运行时实现：
# cuDNN SDPA 为主、FA4 为兜底，并按 layer 级永久锁存失败状态。
class DynamicCudnnSDPAImpl(SDPAImpl):
    def __init__(self, num_heads, head_size, causal, softmax_scale, num_kv_heads=None, prefix=
    "", **extra_impl_args):
        from sglang.multimodal_gen.runtime.layers.attention.backends.flash_attn import (
            FlashAttentionImpl, set_fa_ver,
        )

        self.causal = causal
        self.head_size = head_size
        # sm_100 特判：compute capability 主版本等于 10，Hopper 与 sm_120 不受影响
        self._is_sm100 = (
            torch.cuda.is_available() and torch.cuda.get_device_capability()[0] == 10
        )
        # cuDNN SDPA 一旦在某层抛错，永久钉死 FA 路径，避免每个 step 重复探测失败 kernel
        self._cudnn_failed = False
        if torch.cuda.is_available() and torch.cuda.get_device_capability()[0] >= 10:
            set_fa_ver(4)
        self.cudnn_impl = CudnnSDPAImpl(
            num_heads=num_heads, head_size=head_size, causal=causal,
            softmax_scale=softmax_scale, num_kv_heads=num_kv_heads,
            prefix=f"{prefix}.cudnn", **extra_impl_args,
        )
```

```
        self.fa_impl = FlashAttentionImpl(
            num_heads=num_heads, head_size=head_size, causal=causal,
            softmax_scale=softmax_scale, num_kv_heads=num_kv_heads,
            prefix=f"{prefix}.fa", **extra_impl_args,
        )

    def _use_cudnn_sdpa(self, query, key, value):
        # causal 或已锁存失败时直接走 FA；其余 guard 保持原语义
        if self.causal or self._cudnn_failed:
            return False
        if query.device.type != "cuda":
            return False
        if query.dtype not in (torch.float16, torch.bfloat16):
            return False
        if query.shape[2] != key.shape[2]:
            return False
        if self._is_sm100:
            # B200/sm_100 实测 cuDNN SDPA 比 FA4 CuTe 快 1.25-1.5x；
            # 覆盖 dense self-attn（Sq == Skv，最长 506K）与 cross-attn（Skv = 文本长度）
            return True
        # 非 sm_100 保留原有 Hopper 启发式：仅 D=64、S=1024、B>=4 走 cuDNN
        if query.shape[1] != key.shape[1]:
            return False
        return query.shape[-1] == 64 and query.shape[1] == 1024 and query.shape[0] >= 4

    def forward(self, query, key, value, attn_metadata, **kwargs):
        # return_softmax_lse 只有 FA 实现支持（如 ring attention），必须强制走 FA
        if not kwargs.get("return_softmax_lse") and self._use_cudnn_sdpa(query, key, value):
            try:
                return self.cudnn_impl.forward(query, key, value, attn_metadata)
            except RuntimeError as e:
                # cuDNN 可能对个别 shape 报 "No available kernel"；记录一次后本层永久回退 FA
                logger.warning(
                    "cuDNN SDPA failed (%s); falling back to FlashAttention for %s.",
                    e, type(self).__name__,
                )
                self._cudnn_failed = True
        return self.fa_impl.forward(query, key, value, attn_metadata, **kwargs)
```

python/sglang/multimodal_gen/runtime/loader/component_loaders/transformer_loader.py

新增 _default_quantized_attention_backend 并用 attention backend context manager 包裹 FSDP 模型构造，为 ModelOpt NVFP4 保留 FA4 数值路径。

```
# 在 Blackwell 上，ModelOpt NVFP4 模型默认保持 FA4，避免 cuDNN SDPA
# 改变 bf16 数值路径、破坏量化的稳定输出；用户显式选择则优先。
def _default_quantized_attention_backend(
    quant_spec: TransformerQuantLoadSpec, server_args: ServerArgs
) -> AttentionBackendEnum | None:
```

```
    if not current_platform.is_blackwell() or not quant_spec.is_modelopt_fp4:
        return None
    if (
        get_global_forced_attn_backend() is not None
        or get_component_forced_attn_backend() is not None
        or server_args.attention_backend is not None
    ):
        return None
    return AttentionBackendEnum.FA

# 加载 transformer 组件时，把量化默认后端应用到“模型构造期”：
# attention 实现在构造时解析，因此用 context manager 包住 FSDP init + load。
quantized_attn_backend = _default_quantized_attention_backend(
    quant_spec, component_server_args
)
if quantized_attn_backend is not None:
    logger.info(
        "Using %s attention for ModelOpt NVFP4 to preserve output precision",
        quantized_attn_backend.name.lower(),
    )
attn_backend_context = (
    component_attn_backend_context_manager(
        quantized_attn_backend, component_name=component_name
    )
    if quantized_attn_backend is not None
    else nullcontext()
)
with attn_backend_context:
    model = maybe_load_fsdp_model(
        model_cls=model_cls,
        init_params=init_params,
        weight_dir_list=safetensors_list,
        device=local_torch_device,
        hsdp_replicate_dim=server_args.hsdp_replicate_dim,
        hsdp_shard_dim=server_args.hsdp_shard_dim,
        cpu_offload=component_server_args.dit_cpu_offload,
        pin_cpu_memory=component_server_args.pin_cpu_memory,
        fsdp_inference=component_server_args.use_fsdp_inference,
        param_dtype=quant_spec.param_dtype,
        reduce_dtype=torch.float32,
        output_dtype=None,
        strict=False,
        weight_load_plan=weight_load_plan,
    )
```

评论区精华

PR 本身没有 review 评论，但合并后 issue 评论区出现了一次完整的 H100 回归误报调查，过程很有价值：

- mickqian 首先报告 flux_image_t2i_2_gpus 在 H100 2-GPU 套件上 Denoise Step 从 73.6 ms 基线涨到 420-656 ms，怀疑「要么 sm_100 gate 泄漏到 H100，要么新选择路径每次 step 有 probe 开销」，并给出基于 #33725 CI 时间线（00:56 失败，本 PR 00:49 落地）的归因证据。
- 随后他阅读 diff 后做了机制分析：失败日志中 cuDNN SDPA failed ... falling back to FlashAttention 警告出现 0 次，_is_sm100 在 H100 上读取正确，说明运行时 fallback 路径不是原因；而回归是每个 step 恒定的 6-9x 而非 step 0 单次尖峰，指向 load-time 默认 resolution 的副作用。
- 关键转折：mickqian 更正称自己的归因是错的——失败 job 的 checkout 时间是 00:18 UTC，早于本 PR 落地 31 分钟，运行不可能包含本变更；离线 A/B（ba12a16^ vs ba12a16，H200 ABABx2）两臂一致健康（约 82 ms/step）。
- 最终结论：同一 merge commit 在 rerun（attempt 2）上全绿，失败原因是 runner 状态（当天该 runner 有 incomplete-weight-cache 和 OOM 事故），不存在代码回归。

这次讨论展示了如何用 checkout 时间、日志出现频率、per-step 常量因子等证据链做回归归因，并最终以 rerun 证伪。

- H100 flux_image_t2i_2_gpus 回归误报 (performance): 回归报告被作者自己推翻：失败 job 的 checkout 时间早于本 PR 落地 31 分钟，不可能包含本变更；离线 A/B 两臂一致，最终 rerun 全绿，确认为 runner 状态问题。
- 回归机制分析：fallback 路径被排除 (correctness): 该机制分析本身有效，但因 checkout 时间判断错误而指向了错误嫌疑对象；最终证伪后，分析思路仍值得保留。
- 最终归因：runner 状态而非代码回归 (other): 本 PR 与嫌疑 commit 均无代码回归，case closed。

风险与影响

- 风险：
  1. 平台泄漏风险（已证伪但需留意）：默认切换由 is_blackwell()（cc 10.x）和 _is_sm100 双重守卫，sm_120、Hopper 及更早平台不受影响；H100 回归误报最终被证实为 runner 状态问题，而非代码路径泄漏。
  2. 运行时 fallback 依赖异常类型：DynamicCudnnSDPAImpl.forward 只捕获 RuntimeError 并永久钉回 FA。若 cuDNN 对某 shape 返回非 RuntimeError 的异常（如 torch.cuda.OutOfMemoryError 继承自 RuntimeError，会被错误地当作 kernel 不支持而静默降级），可能掩盖真实资源错误；当前 except RuntimeError 偏宽。
  3. 输出不再 bit-exact：PR body 说明后端切换在 bf16 下非 bit-exact，kernel 级 max-abs-diff 为 2.4e-4（self）/ 7.8e-3（cross），50-step 迭代去噪会放大差异；同 seed 全视频对比 PSNR 28.54 dB、SSIM 0.9468，属于用户使用 --attention-backend 已能获得的自由度，但依赖严格数值复现的流水线需要知晓。
  4. 量化模型数值保护依赖新逻辑：_default_quantized_attention_backend 仅在 is_modelopt_fp4 时生效，若未来新增其他量化格式（如 FP8）也需要类似保护，当前判断口径较窄，需要后续扩展时保持同步。

5. 回归面：transformer_loader.py 新增 context manager 包裹 FSDP 加载，若量化默认解析出错可能影响模型构造期；已有单测覆盖 Blackwell 与显式后端两条路径。 - 影响：影响范围集中在 sglang.multimodal_gen 的 sm_100 平台：B200 上运行 Wan2.2-TI2V/A14B、LingBot-World、MiniMax-H3、Qwen-Image、FLUX 等扩散模型的用户会自动获得约 1.13-1.5x 的端到端或 kernel 级收益，其中长序列（LingBot 506K）收益显著。系统层面，默认值变更不引入新配置项，用户可通过显式 --attention-backend fa 恢复旧行为；ModelOpt NVFP4 量化模型被单独排除在 cuDNN 默认之外，保证数值稳定。对 CI 而言，H100/Hopper 和 sm_120 的 golden 输出不受影响（construction 上仅 cc 10.x 生效），但团队需要关注后续测试矩阵中 B200 job 的 perf 阈值是否要被新的更快基线重新标定。 - 风险标记：sm_100 默认后端变更，NVFP4 数值保护，运行时 fallback 依赖 RuntimeError，非 bit-exact 输出，H100 回归误报已澄清

关联脉络

- PR #33731 [CI] Fix GLM-Image usage unit tests: 与本 PR 最后的提交「Fix GLM-Image usage unit tests」直接关联：同一批测试文件 test_glm_image_ar.py / test_glm_image_multi_output.py 都在适配 GLM-Image AR 批式返回与 usage 聚合的新契约。
- PR #33678 chore: bump sgl-kernel version to 0.4.6: 在 H100 回归误报调查中被列为头号嫌疑（FA kernels 从 sgl-kernel 分发），后经 rerun 证伪排除；与本次默认后端切换同属扩散密集 attention 性能敏感区。