PR #35339 完整报告

sgl-project/sglang

[diffusion] Per-request lossy accelerations: Cache-DiT, CFG gating, attention backend override

合并时间：2026-08-19 08:24

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35339

执行摘要

- 一句话：三套有损加速改为按请求开关，同部署混合精确与加速
- 推荐动作：值得精读。本 PR 的价值不只是加参数，而是确立了一个可复用的设计模式：request-scoped 有损加速的统一治理。重点关注三处——(1) DenoisingStage._maybe_override_attention_backend 的两阶段原子切换与 _validate_attention_backend_override 的集中式原因收集；(2) layer.py 中 prepare 与 apply 分离的可失败 / 不可失败接口划分；(3) cache_dit_overrides_key 用冻结元组做挂载变更检测的思路。这些模式可以直接借鉴到其他按请求切换执行路径的子系统。

功能与动机

PR body 明确提出设计原则：Lossy accelerations should be per-request switches, not process-wide env vars / server args: one deployment should be able to mix accelerated and lossless requests. 改动前 Cache-DiT 依赖约 20 个进程级 SGLANG_CACHE_DIT_* 环境变量，CFG gating 是进程级 SGLANG_DIFFUSION_CFG_GATE_STEP 浮点数，approximate attention（SageAttention）只能服务器级选择；而 TeaCache、Spectrum、progressive resolution、quality="high" 已有按请求控制的先例。PR 确立的规则是：任何 request-scoped 状态、或进程级但可在批次边界安全切换的状态，都应归入 SamplingParams 并参与动态批次签名，环境变量仅作为未显式设置时的服务器默认值。

实现拆解

1. 数据契约层：python/sglang/multimodal_gen/configs/sample/sampling_params.py 新增 enable_cache_dit: bool | None（None 跟随 SGLANG_CACHE_DIT_ENABLED 默认值）、cache_dit_params: dict | None（DBCache/SCM 旋钮覆盖，嵌套 secondary 承载双 DiT 第二组旋钮）、cfg_gate_step: float | None（1.0 表示本请求关闭 CFG 门控）、attention_backend_override: str | None（fa / torch_sdpa / sage_attn / sage_attn_3），并配套 CLI 参数（--enable-cache-dit、--cache-dit-params、--cfg-gate-step、--attention-backend-override）以及 OpenAI image/video API 的 extra_body 透传（image_api.py、video_api.py）。
2. Cache-DiT per-request 化：cache_dit_integration.py 新增 resolve_cache_dit_request_overrides() 做白名单校验（未知键直接让请求失败）与 cache_dit_overrides_key() 把覆盖字典冻结为可哈希元组用于变更检测；DenoisingStage 增加 _cache_dit_request_overrides 与 _cache_dit_active_key 两个状态字段，在

_maybe_enable_cache_dit() 内实现四分支转换（设置不变走 refresh_context 快速路径；参数变化先 _unmount_cache_dit() 再重挂；未请求则卸载）。MiniMax-H3 的 quality="high" 专用逻辑被泛化，且显式 enable_cache_dit=False 作为 kill switch 优先于 quality；LTX-2 的 TI2V 抑制从直接短路改为 _cache_dit_requested_for_batch() 返回 False，让基础阶段真正卸载先前 text-only 请求留下的 hooks；渐进分辨率路径用 _effective_scm_preset() 读取请求覆盖或 env 默认。

3. CFG gating per-request 化：cfg_gate_step 的状态原本就按 denoising loop 重建，因此只需把它接入 _init_cfg_gate_state，请求值覆盖 env 默认、1.0 关闭、非法请求值拒绝，无需额外转换逻辑。
4. 注意力后端覆盖：layer.py 中 UlyssesAttention / LocalAttention / USPAttention 构造时保留 _attn_impl_ctor_kwargs 并建立 _attn_impl_by_backend 缓存，非默认 impl 懒构建、双驻留仅占少量 Python 对象无 GPU 内存；新增 prepare_attention_backend_override()（构建并缓存 impl，selector 静默解析到非目标后端时直接拒绝）与 apply_attention_backend_override()（不可失败的 flip）。DenoisingStage._maybe_override_attention_backend() 编排两阶段切换，_validate_attention_backend_override() 集中收集 breakable CUDA graph、torch.compile、sparse 服务器后端、ring 并行、无切换层、未知目标等全部不兼容原因并一次性拒绝。
5. 测试与文档：新增test_cache_dit_per_request.py（236行）覆盖参数校验、键变更检测、mount→refresh→unmount→remount 转换、kill switch、secondary 继承、warmup 不挂载；新增 test_attention_backend_override.py（228 行）覆盖默认 no-op、切换/恢复、全部 fail-fast 拒绝路径、两阶段原子性、prepare 不污染活动 impl；扩展 test_cfg_gating.py 的请求覆盖与拒绝用例，更新 test_diffusion_bcg_padding.py 与 test_minimax_h3_admission.py 的 fakes；docs/docs/sglang-diffusion/cache_dit.mdx 重写为 per-request 用法，attention_backends.mdx 增加 override 章节。

关键文件：

- python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising.py（模块 去噪编排；类别 source；类型 core-logic；符号 _maybe_override_attention_backend, _parse_attention_backend_override, _request_switchable_attention_layers, _validate_attention_backend_override）：所有 per-request 开关的编排中枢：新增 _maybe_override_attention_backend 两阶段切换、Cache-DiT 四分支状态转换、_cache_dit_requested_for_batch 请求级决策，是整个 PR 的核心入口。
- python/sglang/multimodal_gen/runtime/cache/cache_dit_integration.py（模块 缓存集成；类别 source；类型 core-logic；符号 resolve_cache_dit_request_overrides, cache_dit_overrides_key, _freeze）：新增请求覆盖参数的白名单校验与可哈希变更键，是 Cache-DiT per-request 化的契约基础。
- python/sglang/multimodal_gen/runtime/layers/attention/layer.py（模块 注意力层；类别 source；类型 core-logic；符号 prepare_attention_backend_override, apply_attention_backend_override）：注意力层的两阶段切换原语与 per-backend impl 缓存都落在这里，是 attention_backend_override 能原子生效的机制基础。

- python/sglang/multimodal_gen/configs/sample/sampling_params.py（模块 采样参数；类别 source；类型 data-contract；符号 cfg_gate_step, enable_cache_dit, cache_dit_params, attention_backend_override）：SamplingParams 新增四个 per-request 字段及对应 CLI 参数，是这次 API 契约变更的入口。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/ltx_2/denoising.py（模块 LTX-2 去噪；类别 source；类型 bugfix；符号 _cache_dit_requested_for_batch）：修复 TI2V 请求 stale-activation 隐患：从直接短路改为向基础阶段报告 not requested，让 hooks 真正卸载。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/minimax_h3/stages/denoising.py（模块 H3 去噪；类别 source；类型 data-contract；符号 _maybe_enable_cache_dit, _disarm_after_failed_mount）：处理请求级 kill switch 与 quality="high" 的优先级交互，并复用 _unmount_cache_dit 统一卸载逻辑。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/progressive_resolution/denoising.py（模块 渐进分辨率；类别 source；类型 core-logic；符号 _effective_scm_preset）：渐进分辨率路径的 SCM preset 改为读取请求覆盖再回落 env 默认，保证 per-request 语义贯通。
- python/sglang/multimodal_gen/test/unit/test_cache_dit_per_request.py（模块 缓存测试；类别 test；类型 test-coverage；符号 TestResolveCacheDitRequestOverrides, TestPerRequestCacheDitTransitions, test_overrides_key_detects_changes, test_changed_overrides_unmount_and_remount）：新增的 Cache-DiT per-request 专项测试，覆盖参数校验、kill switch、secondary 继承与挂载状态转换。
- python/sglang/multimodal_gen/test/unit/test_attention_backend_override.py（模块 注意力测试；类别 test；类型 test-coverage；符号 TestMaybeOverrideAttentionBackend, TestLayerPrepareApply, test_prepare_failure_leaves_layers_unswitched）：新增的注意力后端覆盖专项测试，覆盖全部 fail-fast 拒绝路径与两阶段原子性。
- python/sglang/multimodal_gen/test/unit/test_cfg_gating.py（模块 CFG 测试；类别 test；类型 test-coverage；符号 test_request_fraction_overrides_env_default, test_request_fraction_disables_env_default, test_rejects_invalid_request_fraction）：扩展 CFG gating 测试：请求分数覆盖 env 默认、请求 1.0 关闭 env 门控、非法分数拒绝。
- python/sglang/multimodal_gen/runtime/entrypoints/openai/image_api.py（模块 API 入口；类别 source；类型 entrypoint）：OpenAI image API 增加 per-request 字段透传，是外部用户触达新开关的入口之一。
- docs/docs/sglang-diffusion/cache_dit.mdx（模块 文档；类别 docs；类型 documentation）：文档围绕 per-request 用法重写，是用户理解新开关语义与 env 默认关系的重要材料。

关键符号：DenoisingStage._maybe_override_attention_backend, DenoisingStage._parse_attention_backend_override, DenoisingStage._validate_attention_backend_override, DenoisingStage._request_switchable_attention_layers, DenoisingStage._cache_dit_requested_for_batch, DenoisingStage._unmount_cache_dit, DenoisingStage._parse_cache_dit_scm_bins, resolve_cache_dit_request_overrides,

cache_dit_overrides_key, prepare_attention_backend_override,apply_attention_backend_override, LTX2DenoisingStage._cache_dit_requested_for_batch, ProgressiveDenoisingStage._effective_scm_preset

关键源码片段

python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising.py

所有 per-request 开关的编排中枢：新增 _maybe_override_attention_backend 两阶段切换、Cache-DiT 四分支状态转换、_cache_dit_requested_for_batch 请求级决策，是整个 PR 的核心入口。

```
# python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising.py
# 只有精确 / 可无损替换的稠密 kernel 才允许按请求切换；
# 稀疏家族需要 per-model mask 配置与 per-step 元数据，仍保持服务器级选择。
REQUEST_SWITCHABLE_ATTENTION_BACKENDS = frozenset(
    {
        AttentionBackendEnum.FA,
        AttentionBackendEnum.TORCH_SDPA,
        AttentionBackendEnum.SAGE_ATTN,
        AttentionBackendEnum.SAGE_ATTN_3,
    }
)

def _maybe_override_attention_backend(self, batch: Req) -> None:
    """两阶段 per-request 后端切换：先 prepare 所有层（可能抛异常，
    但完全不改动状态），再统一 flip——请求被拒时 transformers 保持原样。
    该字段参与动态批次签名，因此切换总发生在批次边界，安全。"""
    target = self._parse_attention_backend_override(
        batch.sampling_params.attention_backend_override
    )
    if target == self._attention_backend_active_override:
        return # 快速路径：目标未变化，跳过所有层操作
    layers = self._request_switchable_attention_layers()
    stage_backend = self._attn_backend_default
    if target is not None:
        # 校验服务器级不兼容配置；失败抛 ValueError，不会执行到 flip
        stage_backend = self._validate_attention_backend_override(target, layers)
        for layer in layers:
            prepare_attention_backend_override(layer, target)
    for layer in layers:
        apply_attention_backend_override(layer, target)
    self.attn_backend = stage_backend
    self._attention_backend_active_override = target
    logger.info(
        "Attention backend for this batch: %s (%d layers switched)",
        target.name.lower() if target else "server default",
        len(layers),
    )
```

```
def _cache_dit_requested_for_batch(self, batch: Req) -> bool:
    """per-request Cache-DiT 开关；未设置时回落到服务器默认。"""
    enable = batch.sampling_params.enable_cache_dit
    if enable is None:
        return self._cache_dit_requested()
    return enable

def _unmount_cache_dit(self) -> None:
    """移除 Cache-DiT hooks，让后续批次运行原生 forward。"""
    for transformer in filter(None, [self.transformer, self.transformer_2]):
        disable_cache_on_transformer(transformer)
    self._cache_dit_enabled = False
    self._cached_num_steps = None
    self._cache_dit_active_key = None
```

python/sglang/multimodal_gen/runtime/cache/cache_dit_integration.py

新增请求覆盖参数的白名单校验与可哈希变更键，是 Cache-DiT per-request 化的契约基础。

```
# python/sglang/multimodal_gen/runtime/cache/cache_dit_integration.py
# SamplingParams.cache_dit_params 中接受的键；"secondary" 嵌套承载
# 双 DiT 模型中第二个 transformer 的 DBCache 旋钮。
CACHE_DIT_REQUEST_KNOB_KEYS = frozenset(
    {
        "Fn_compute_blocks",
        "Bn_compute_blocks",
        "max_warmup_steps",
        "residual_diff_threshold",
        "max_continuous_cached_steps",
        "enable_taylorseer",
        "taylorseer_order",
    }
)
CACHE_DIT_REQUEST_SCM_KEYS = frozenset(
    {"scm_preset", "scm_compute_bins", "scm_cache_bins", "scm_policy"}
)
CACHE_DIT_REQUEST_PARAM_KEYS = (
    CACHE_DIT_REQUEST_KNOB_KEYS | CACHE_DIT_REQUEST_SCM_KEYS | {"secondary"}
)

def resolve_cache_dit_request_overrides(raw: dict | None) -> dict:
    """校验 cache_dit_params 并返回副本；未知键直接让请求失败。"""
    if raw is None:
        return {}
    if not isinstance(raw, dict):
        raise ValueError(f"cache_dit_params must be a dict, got {type(raw).__name__}.")
    unknown = set(raw) - CACHE_DIT_REQUEST_PARAM_KEYS
    if unknown:
        raise ValueError(
            f"Unknown cache_dit_params keys: {sorted(unknown)}. "
```

```
            f"Valid keys: {sorted(CACHE_DIT_REQUEST_PARAM_KEYS)}."
        )
    overrides = dict(raw)
    secondary = overrides.get("secondary")
    if secondary is not None:
        if not isinstance(secondary, dict):
            raise ValueError(
                "cache_dit_params['secondary'] must be a dict, got "
                f"{type(secondary).__name__}."
            )
        unknown = set(secondary) - CACHE_DIT_REQUEST_KNOB_KEYS
        if unknown:
            raise ValueError(
                f"Unknown cache_dit_params['secondary'] keys: {sorted(unknown)}. "
                f"Valid keys: {sorted(CACHE_DIT_REQUEST_KNOB_KEYS)}."
            )
        overrides["secondary"] = dict(secondary)
    return overrides

def cache_dit_overrides_key(overrides: dict) -> tuple:
    """请求覆盖的哈希快照，用于 mount 变更检测（忽略 dict 键序）。"""

    def _freeze(value):
        if isinstance(value, dict):
            return tuple(sorted((k, _freeze(v)) for k, v in value.items()))
        if isinstance(value, (list, tuple)):
            return tuple(_freeze(v) for v in value)
        return value

    return _freeze(overrides)
```

python/sglang/multimodal_gen/runtime/layers/attention/layer.py

注意力层的两阶段切换原语与 per-backend impl 缓存都落在这里，是 attention_backend_override 能原子生效的机制基础。

```
# python/sglang/multimodal_gen/runtime/layers/attention/layer.py
def prepare_attention_backend_override(
    layer: nn.Module, target: AttentionBackendEnum
) -> None:
    """构建并缓存 target 对应的 impl；可能抛异常，但不改动任何状态。"""
    if target in layer._attn_impl_by_backend:
        return # 已构建过的 impl 直接复用
    backend_cls = get_attn_backend(
        layer.head_size,
        layer.dtype,
        supported_attention_backends=layer._supported_attention_backends,
        selected_attention_backend=target,
    )
    resolved = backend_cls.get_enum()
```

```
    if resolved is not target:
        # selector 发生了静默 fallback：直接拒绝请求，而不是降低精度继续跑
        raise ValueError(
            f"Attention backend override '{target}' resolved to '{resolved}' on "
            f"{type(layer).__name__}; refusing the request instead of silently "
            "falling back."
        )
    impl = backend_cls.get_impl_cls()(**layer._attn_impl_ctor_kwargs)
    wrap_attention_impl_forward(impl)
    layer._attn_impl_by_backend[target] = impl

def apply_attention_backend_override(
    layer: nn.Module, target: AttentionBackendEnum | None
) -> None:
    """切换到已准备好的 impl（None 表示回到构造时的默认后端）；不会失败。"""
    target = target or layer._default_attn_backend
    if target is layer.backend:
        return
    layer.attn_impl = layer._attn_impl_by_backend[target]
    layer.backend = target
```

评论区精华

该 PR 没有收到实质 review 评论（review_comments_count: 0；仅有的 issue 评论是 Mintlify 预览部署通知和作者的 /tag-and-rerun-ci 命令），设计讨论都沉淀在 PR body 与 commit message 中，可视为作者自述的设计决策：

- 两阶段原子性：switching is two-phase (prepare every layer's impl first — may raise, mutates nothing — then flip all), so a rejected request leaves the transformers untouched。
- fail-fast 拒绝：incompatible server settings reject the request with a server log instead of silently falling back。
- 批次边界安全：All three fields participate in the dynamic-batch signature, so requests with different settings never share a batch and every transition happens at a batch boundary.
- 环境变量降级为默认值：Env vars remain as server-wide defaults for unset requests, and are planned to be deprecated/removed later.
- 两阶段原子切换 vs 静默回退 (design)：采用 fail-fast：不兼容的服务器设置以服务器日志拒绝请求，并集中收集全部原因一次性报错。
- 请求级开关与服务器默认值的优先级 (design)：请求优先、env 仅作默认；三个字段全部参与动态批次签名，不同设置的请求永不共享批次。

风险与影响

- 风险：

1. 状态机复杂度：DenoisingStage 的 Cache-DiT 现在有 mount/refresh/unmount/remount 四分支转换，_cache_dit_active_key 与 _cache_dit_request_overrides 两个共享状态字段在异常路径上容易不一致；MiniMax-H3 已需要专门的 _disarm_after_failed_mount 清理失败挂载现场，证明此处防御复杂度确实存在。
2. API 契约变更：SamplingParams 新增四个字段影响所有构造方；enable_cache_dit=False 覆盖 quality="high" 的优先级语义是敏感点，依赖方需要明确知晓。
3. fail-fast 行为变更：开启 breakable CUDA graph 或 torch.compile 的服务器收到 sage_attn 请求会被直接拒绝，旧版会静默继续跑——属于刻意行为变更，但需通过文档提示用户。
4. 兼容期双配置并存：env 与请求参数并存的过渡期，None 跟随 env、显式值覆盖的语义需要用户理解，可能出现"配了 env 但请求显式关闭"的困惑。
5. 性能面：每批次边界多一次字符串解析与 override 键冻结，均为廉价操作；非默认 attention impl 懒构建避免 GPU 内存双驻留，性能风险低。 - 影响：对用户：单一部署即可混合服务无损与有损（加速）请求，OpenAI image/video API 通过 extra_body 即可按请求选择 Cache-DiT、CFG 门控和近似注意力，无需重启服务器切换全局开关。对系统：DenoisingStage 每批次边界多做参数解析与状态比对，但仅影响 SGLang-Diffusion 子系统，不触及 SGLang SRT 核心推理路径；LTX-2 的 stale-activation 隐患被修复，属于正确性收益。对团队：确立了"request-scoped 状态进 SamplingParams、env 仅作默认并计划弃用"的架构原则，后续新的加速特性应沿用此模式；同时新增两个专项测试文件，为状态转换与 fail-fast 路径提供了回归保障。 - 风险标记：核心路径变更：DenoisingStage 去噪循环入口，API 契约变更：SamplingParams 新增字段，状态机复杂度：mount/refresh/unmount/remount 转换，fail-fast 拒绝新请求

关联脉络

- PR #34581 [Diffusion] Optimizing MiniMax-H3 for consumer-level GPUs: INT8 Linear + pluggable DiT attention backends: 同属注意力后端可插拔架构的演进线：本 PR 的 attention_backend_override 建立在 get_attn_backend selector 之上，把原来服务器级的后端选择下沉到请求级。
- PR #35114 [kernels] Reorganize ops/diffusion by operator domain behind a lazy facade: 同为 diffusion 子系统基础设施演进，前者重组内核组织方式，本 PR 在其上增加按请求切换的执行路径。