PR #34391 完整报告

sgl-project/sglang

[Diffusion] Support dynamic CPU offload components

合并时间：2026-08-12 11:38

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34391

执行摘要

- 一句话：--cpu-offload-components 支持 model_index 动态键与 all/none
- 推荐动作：值得精读。核心设计是三层：normalize_cpu_offload_components 做输入规范化、ServerArgs.should_cpu_offload_component 做统一决策、component_manager 的 _should_keep_single_dit 让性能快路径服从内存策略。建议重点对照 server_args.py 的 _adjust_cpu_offload_components 与 component_manager.py 的 finish_request 逻辑；如果团队在持续扩展新 diffusion 模型，这套“动态组件键 + 统一装载判定”模式可直接复用；若只维护现有模型，需关注 MPS 与 legacy flag 混用边界。

功能与动机

PR body 明确说明 --cpu-offload-components 应当支持 Diffusers model_index.json 中的组件键，包括“model-specific components that are not known to SGLang in advance”。以往只有 dit/text_encoder/image_encoder/vae 四个预定义分组，新增 checkpoint 若带 transformer_2、audio_vae、connectors 等动态键则无法细粒度控制卸载，甚至被忽略导致显存占用超出预期。本 PR 引入统一选择器并保持 legacy 别名兼容，同时解决 single-DiT 常驻性能优化与显式 offload 策略冲突的历史问题。

实现拆解

1. 新增统一参数与规范化：server_args.py 增加 cpu_offload_components 字段及 --cpu-offload-components CLI 参数；layerwise_offload_components.py 新增 normalize_cpu_offload_components（支持逗号分隔、连字符转下划线、大小写归一、none 单独出现）与 cpu_offload_component_matches（支持 all 通配、精确键、legacy 分组别名的匹配）。
2. 统一决策入口：ServerArgs.should_cpu_offload_component 成为唯一判定入口，cpu_offload_components 优先，未设置时回退到 legacy flag 分组判断，并把 connectors、unconditional_transformer、vision_language_encoder 归入 DiT 组、sound_tokenizer 归入 VAE 组。_adjust_cpu_offload_components 在 _adjust_parameters 早期执行，将统一选择同步到 dit/text_encoder/image_encoder/vae 四个 legacy flag，且与显式 legacy flag 冲突时直接报 ValueError。
3. 加载链路迁移：component_loader.py 基类 should_offload 增加 component_name 参数并默认走统一判断；text_encoder_loader.py、image_encoder_loader.py、sound_tokenizer_loader.py、vae_loader.py、vocoder_loader.py、upsampler_loader.py、

vl_encoder_loader.py、adapter_loader.py、transformer_loader.py 等逐一改为调用 server_args.should_cpu_offload_component(component_name)，component_loader.load() 中还增加对非 FSDP 模块的 CPU 兜底移动，覆盖通用组件加载路径。

4. 驻留策略调整：component_manager.py 删除模块级 should_cpu_offload_component，把 MPS/FSDP/ 统一选择判断内联进 build_component_residency_strategy；新增 _is_single_dit_component，_should_keep_single_dit 改为依赖实际策略是否为 ResidentStrategy，并移除 lru_cache；finish_request 中 preferred 预取不再覆盖 single-DiT 的 CPU/layerwise offload。
5. 内存分析与自动策略联动：gpu_worker.py 的 get_can_stay_resident_components 改为遍历 pipeline.memory_usages 全部键（含动态组件），用 should_cpu_offload_component 与 should_configure_layerwise_offload_for_lazy_component 判定，删除硬编码 offload flag 映射与 _format_offload_disable_suggestions；auto_tune.py 调整 FSDP 自动选择与组件驻留调整的顺序；wan.py、lingbot_world.py、zimage 等模型部署配置补充 keep_resident_min_available_gb 与 keep_resident_components 声明。
6. 测试配套：test_server_args.py 新增 CLI 解析、model_index 键保留、all/none 通配、legacy flag 冲突、5090/H100 自动驻留策略等大量单元测试；test_causal_denoising.py 新增 DiT 前向调用前 prepare 顺序测试；最后提交刷新 B200 Ideogram4 基准 ground truth（+631/-173，覆盖 31 个文件）。

关键文件：

- python/sglang/multimodal_gen/runtime/managers/memory_managers/component_manager.py（模块 驻留管理；类别 source；类型 core-logic；符号 should_cpu_offload_component, _should_keep_single_dit, _is_single_dit_component）：驻留策略主路径：删除模块级 should_cpu_offload_component，内联 MPS/FSDP/ 统一选择判断；新增 _is_single_dit_component，_should_keep_single_dit 改为依据实际 ResidentStrategy，并调整 finish_request 中 preferred 预取不覆盖 offload 策略，是本 PR 优先级修正的核心。
- python/sglang/multimodal_gen/runtime/server_args/server_args.py（模块 参数解析；类别 source；类型 core-logic；符号 _adjust_cpu_offload_components, should_cpu_offload_component）：新增 --cpu-offload-components 参数定义、_adjust_cpu_offload_components 与 should_cpu_offload_component 统一决策入口，并处理与 legacy flag 的显式冲突，是整个变更的配置契约层。
- python/sglang/multimodal_gen/runtime/managers/memory_managers/layerwise_offload_components.py（模块 卸载选择；类别 source；类型 core-logic；符号 normalize_cpu_offload_components, cpu_offload_component_matches）：新增 normalize_cpu_offload_components 与 cpu_offload_component_matches 两个核心辅助函数，负责参数规范化与动态组件匹配，是统一选择器的底层契约。
- python/sglang/multimodal_gen/runtime/managers/gpu_worker.py（模块 工作进程；类别 source；类型 core-logic；符号 _format_offload_disable_suggestions）：内存分析 get_can_stay_resident_components 改为遍历 pipeline.memory_usages 全部动态键并用

统一判定，删除硬编码 flag map 与 _format_offload_disable_suggestions，诊断信息同步更新。

- python/sglang/multimodal_gen/runtime/loader/component_loaders/component_loader.py（模块 加载器；类别 source；类型 dependency-wiring）：基类 should_offload 从默认 False 改为接收 component_name 并走统一判断，load() 增加对非 FSDP 模块的 CPU 兜底移动，是通用组件加载链路的出口。
- python/sglang/multimodal_gen/runtime/loader/component_loaders/image_encoder_loader.py（模块 加载器；类别 source；类型 core-logic；符号 should_offload）：代表 encoder 加载链路的迁移：should_offload 增加 component_name 参数，并透传给 FSDP 相关加载路径，确保 image_encoder 的动态判断一致。
- python/sglang/multimodal_gen/test/unit/test_server_args.py（模块 参数测试；类别 test；类型 test-coverage；符号 test_cpu_offload_components_cli_args, test_cpu_offload_components_preserves_model_index_names, test_cpu_offload_components_all_matches_dynamic_components, test_cpu_offload_components_none_disables_all_legacy_flags）：新增约 250 行测试，覆盖 CLI 解析、model_index 动态键保留、all/none 通配、legacy flag 冲突，以及 5090/H100 上 zimage/lingbot/wan 等自动驻留策略，是变更契约的主要回归保障。

关键符号：should_cpu_offload_component, _adjust_cpu_offload_components, normalize_cpu_offload_components, cpu_offload_component_matches, build_component_residency_strategy, _should_keep_single_dit, _is_single_dit_component, get_can_stay_resident_components, should_offload, load_customized

关键源码片段

python/sglang/multimodal_gen/runtime/managers/memory_managers/component_manager.py

驻留策略主路径：删除模块级 should_cpu_offload_component，内联 MPS/FSDP/ 统一选择判断；新增 _is_single_dit_component，_should_keep_single_dit 改为依据实际 ResidentStrategy，并调整 finish_request 中 preferred 预取不覆盖 offload 策略，是本 PR 优先级修正的核心。

```
# 源码：python/sglang/multimodal_gen/runtime/managers/memory_managers/component_manager.py
# 判断某个流水线组件是否应该做整模块 CPU offload（D2H）。
# 注意：MPS 与 FSDP 模块一律留在原设备，避免破坏已验证的 CUDA 行为。
def build_component_residency_strategy(
    component_name: str,
    module: nn.Module,
    server_args: ServerArgs,
) -> ComponentResidencyStrategy:
    if is_layerwise_offloaded_module(module):
        return LayerwiseOffloadStrategy()
    if (
```

```
            not current_platform.is_mps()
            and not server_args.use_fsdp_inference
            and not is_fsdp_managed_module(module)
            and server_args.should_cpu_offload_component(component_name)
        ):
            return VanillaD2HStrategy()
        return ResidentStrategy()


    class ComponentResidencyManager:
        # 单一 DiT 的"留在显存"只是性能快路径，不是内存策略，
        # 因此只有真正拿到 ResidentStrategy 时才保留 DiT。
        # finish_request 里也禁止 preferred 预取覆盖 CPU/layerwise offload。
        def _should_keep_single_dit(self, component_name: str, module: nn.Module) -> bool:
            """Keep a single DiT resident only when its effective strategy is resident."""
            if not self._is_single_dit_component(component_name):
                return False
            return isinstance(self.strategy_for(component_name, module), ResidentStrategy)

        def _is_single_dit_component(self, component_name: str) -> bool:
            modules = self.pipeline.modules
            return (component_name == "transformer" and "transformer_2" not in modules) or (
                component_name == "video_dit" and "video_dit_2" not in modules
            )
```

python/sglang/multimodal_gen/runtime/server_args/server_args.py

新增 --cpu-offload-components 参数定义、_adjust_cpu_offload_components 与 should_cpu_offload_component 统一决策入口，并处理与 legacy flag 的显式冲突，是整个变更的配置契约层。

```
    # 统一入口：--cpu-offload-components 会把 model_index.json 的任意组件键映射到
    # 各 legacy 布尔 flag，保证下游 loader 不再需要感知动态组件名。
    def _adjust_cpu_offload_components(self) -> None:
        if self.cpu_offload_components is None:
            return

        legacy_flags = (
            "dit_cpu_offload",
            "text_encoder_cpu_offload",
            "image_encoder_cpu_offload",
            "vae_cpu_offload",
        )
        conflicting_flags = [
            flag_name
            for flag_name in legacy_flags
            if self.is_arg_explicitly_set(flag_name)
        ]
        if conflicting_flags:
            # 显式互斥，避免两个 selector 同时生效导致行为不可预测
```

```
            formatted_flags = ", ".join(
                "--" + flag_name.replace("_", "-") for flag_name in conflicting_flags
            )
            raise ValueError(
                "--cpu-offload-components cannot be combined with the legacy "
                f"CPU offload flags: {formatted_flags}"
            )

        selected_components = (
            normalize_cpu_offload_components(self.cpu_offload_components) or []
        )
        self.cpu_offload_components = selected_components
        # 把统一选择同步到 legacy flag，保持旧 loader 与旧诊断逻辑可读
        self.dit_cpu_offload = self.should_cpu_offload_component("transformer")
        self.text_encoder_cpu_offload = self.should_cpu_offload_component("text_encoder")
        self.image_encoder_cpu_offload = self.should_cpu_offload_component("image_encoder")
        self.vae_cpu_offload = self.should_cpu_offload_component("vae")


    def should_cpu_offload_component(self, component_name: str) -> bool:
        # 统一选择器优先；未设置时回退到 legacy 分组判断，
        # 并把 connectors / sound_tokenizer 等归入既有组，保证动态组件可用。
        if self.cpu_offload_components is not None:
            return cpu_offload_component_matches(
                component_name, self.cpu_offload_components
            )
        if is_dit_component_name(component_name) or component_name in (
            "connectors",
            "unconditional_transformer",
            "vision_language_encoder",
        ):
            return bool(self.dit_cpu_offload)
        if is_text_encoder_component_name(component_name):
            return bool(self.text_encoder_cpu_offload)
        if is_image_encoder_component_name(component_name):
            return bool(self.image_encoder_cpu_offload)
        if is_vae_component_name(component_name) or component_name == "sound_tokenizer":
            return bool(self.vae_cpu_offload)
        return False
```

python/sglang/multimodal_gen/runtime/managers/memory_managers/layerwise_offload_components.py

新增 normalize_cpu_offload_components 与 cpu_offload_component_matches 两个核心辅助函数，负责参数规范化与动态组件匹配，是统一选择器的底层契约。

```
# 规范化 --cpu-offload-components 的输入：
# 支持逗号分隔、连字符转下划线、大小写归一；'none' 必须单独出现。
def normalize_cpu_offload_components(
    component_names: str | Sequence[str] | None,
```

```
) -> list[str] | None:
    if component_names is None:
        return None
    raw_components = (
        [component_names] if isinstance(component_names, str) else component_names
    )
    normalized_components: list[str] = []
    for raw_component in raw_components:
        if not isinstance(raw_component, str):
            raise ValueError(f"Invalid CPU offload component name: {raw_component}.")
        normalized_components.extend(
            component_name
            for value in raw_component.split(",")
            if (component_name := value.strip().replace("-", "_").lower())
        )

    unique_components = list(dict.fromkeys(normalized_components))
    if "none" in unique_components:
        if len(unique_components) != 1:
            raise ValueError("'none' cannot be combined with other components.")
        return [] # none 表示显式关闭所有组件 offload
    return unique_components or None


# 匹配一个运行时组件名是否落在用户选择中：
# 'all' 通配所有模块；显式键优先；其次兼容 legacy 分组别名。
def cpu_offload_component_matches(
    component_name: str,
    selected_component_names: Collection[str] | None,
) -> bool:
    if selected_component_names is None:
        return False
    if CPU_OFFLOAD_ALL_COMPONENTS in selected_component_names:
        return True
    if component_name in selected_component_names:
        return True
    if LAYERWISE_OFFLOAD_DIT_GROUP in selected_component_names:
        return is_dit_component_name(component_name)
    if LAYERWISE_OFFLOAD_TEXT_ENCODER_GROUP in selected_component_names:
        return is_text_encoder_component_name(component_name)
    if LAYERWISE_OFFLOAD_IMAGE_ENCODER_GROUP in selected_component_names:
        return component_name in ("image_encoder", "condition_image_encoder")
    if LAYERWISE_OFFLOAD_VAE_GROUP in selected_component_names:
        return is_vae_component_name(component_name)
    return False
```

评论区精华

该 PR 没有外部 review 评论（review_comments_count=0），核心权衡体现在 10 次作者自审提交中，从提交信息可读出关键决策：

- a0b41c5 "honor offload policy in component residency"：开始让驻留策略服从 offload 配置，这是优先级修正的起点。
- 9ff7336 "restore causal DiT residency"、e2df3120 "keep Wan single DiTs resident on H100"、df96ed0 "keep distilled video DiTs resident on H100"：在 offload 与 keep-resident 性能之间反复调优，最终把 single-DiT 常驻从“名称匹配”改为“实际策略为 ResidentStrategy 才保留”。
- 94d5feb "preserve auto FSDP selection"：调整 auto_tune 执行顺序，避免统一选择器干扰 FSDP 自动判定。
- 结论：单一 DiT 常驻是性能快路径而非内存策略，必须服从显式 / 自动 CPU 与 layerwise offload；finish_request 的 preferred 预取也不得覆盖该策略。
- 单一 DiT 驻留快路径与 offload 策略的优先级 (design): _should_keep_single_dit 从名称匹配改为策略匹配，finish_request 中 preferred 预取不覆盖 CPU/layerwise offload。
- 统一选择器与 auto FSDP 自动调整的执行顺序 (design): 调整 auto_tune 顺序：先 FSDP 相关调整，再做组件驻留 after-offload 调整。
- MPS 平台下 --cpu-offload-components 与强制禁用 legacy flag 的交互 (question): 未解决；现有测试未覆盖 MPS 与统一选择器组合，建议补充短路或测试。

风险与影响

- 风险：
  1. MPS 平台组合未覆盖：_adjust_platform_specific 对 MPS 会强制把所有 legacy flag 置 False 并告警，但 should_cpu_offload_component 在 cpu_offload_components 非 None 时直接返回匹配结果，MPS 上可能仍把组件移到 CPU，与“仅 CUDA 验证”的注释相矛盾，存在设备行为不一致风险。
  2. 兼容性：--cpu-offload-components 与任一显式 legacy flag 同时设置会直接 ValueError 导致启动失败，虽然错误信息清晰，但既有启动脚本若混用会中断。
  3. 加载峰值内存：component_loader.load() 通用路径对非 FSDP 模块执行 to("cpu")，若后续又参与 layerwise offload 配置可能发生重复搬运，对低显存场景峰值内存影响需要 GPU/CI 实测（PR body 也注明请求 GPU/CI 验证）。
  4. lru_cache 生命周期：_should_keep_single_dit 移除缓存后每次请求多一次 strategy_for 查找与 isinstance 判断，开销可接受；但 strategy_for 本身仍以 module 为键做 lru_cache，若 pipeline 模块动态重建可能累积旧引用。
  5. auto_tune 顺序敏感：_adjust_cpu_offload_components 在 auto_tuner 之前执行，若用户在 performance_mode=auto 下同时传 --cpu-offload-components，部分 auto 默认逻辑仍可能改写 legacy flag，需关注测试未覆盖组合。 - 影响：影响范围集中在 sglang.multimodal_gen 模块：所有 Diffusion 组件加载路径（native、FSDP、transformer、VAE、encoder、bridge、adapter、vocoder、sound-tokenizer、upsampler、generic）与内存驻留管理、内存分析诊断均改为统一入口。用户侧获得可扩展的统一 offload 参数，对现有模型完全向后兼容 legacy flags；系统侧把“组件名 ->

是否卸载 -> 驻留策略”判定收敛到单点，后续新增 diffusion 模型无需改动四个旧 flag。团队侧需要理解 should_cpu_offload_component 单入口与 auto_tune 顺序，诊断日志不再给出具体 --xxx-cpu-offload 建议，改为提示调整 --cpu-offload-components 或 --layerwise-offload-components。DB/ 部署无外部依赖变更。 - 风险标记：核心路径变更，需要 GPU/CI 验证，MPS 平台组合未覆盖，与 legacy flag 混用会启动失败，移除 lru_cache 后 module 引用生命周期

关联脉络

- PR #34401 Fix model-driven DiT layerwise offload auto policy: 同一 offload/auto-tune 策略线，改动 auto_tune.py、model_deployment_config.py、wan.py、mova.py、lingbot_world.py 与平台文件，与本 PR 的 keep_resident 配置、auto FSDP 顺序调整直接相关。
- PR #34314 [diffusion] Ideogram-4: fuse Qwen3-style RoPE and SwiGLU silu-mul (denoise -5.1% H100 / -4.7% H200, bit-exact): 本 PR 最后一个提交刷新 B200 Ideogram4 基准 ground truth，与 Ideogram4 的性能基准线关联。
- PR #32921 [diffusion][model] Add native SANA-Video T2V support: 新增 dits/loader/registry 路径，与本 PR 动态组件加载链路互补，是统一组件选择器受益的模型新增场景。
- PR #31590 Add Cosmos3 Edge and Distilled checkpoints support: 引入新的 model_index 组件与 loader 扩展，是“SGLang 不预知组件键也可支持”这一需求的典型来源。