

PR #39303 完整报告

sgl-project/sglang

[diffusion] Serve requests that turn CFG off on a CFG-parallel server

合并时间: 2026-09-14 11:26

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39303>

执行摘要

- 一句话: CFG 并行服务器放行关闭 CFG 的请求, 并修复 `warning_once` 格式参数崩溃
- 推荐动作: 值得精读。这个 PR 的亮点不在改动规模, 而在于作者对“可观察契约”的严谨处理: `stacklevel=2`、`logger.warning` 方法路由、`cache_clear` API 表面都是被测试字面断言的外部契约, 稍一“简化”就会让整套 GPU CI 快速失败。它对三类读者有参考价值: 维护 `diffusion serving` 的同学 (理解 CFG 并行的单分支语义)、想改公共日志工具的同学 (去重缓存与内存泄漏的权衡)、以及写分布式测试的同学 (`TestCfgParallelServesOneBranch` 用 `patch` 在 CPU 上模拟多 rank gather 的思路)。建议重点关注 `logging_utils.py` 的注释块与 5 个 `commit` 的演进顺序。

功能与动机

PR body 明确指出: `sglang serve --model-path Qwen/Qwen-Image-2512 --num-gpus 2 --model-type diffusion` 未传任何并行参数, 运行时从模型默认采样参数自动开启 CFG 并行 (`enable_cfg_parallel: true, cfg_parallel_degree: 2`), 随后每个 `guidance_scale=1.0` 请求都被拒绝, 报错 `Server was launched with --enable-cfg-parallel but this request does not use classifier-free guidance`——而用户从未传过该 flag。跨框架 benchmark 中 `Qwen-Image-2512`、`Qwen-Image-Edit-2511` 两个 cell 在默认命令下连一个请求都服务不了。作者还考证了守卫的历史来源: 该拒绝为 #23198 的 `warmup hang` (2026-04-23) 添加, 而 #23736 多分支重构 (2026-05-07) 已让 `dispatcher` 支持单分支 (`dispatch_branches(1, 2) == [[0], []]`, 无分支的 rank 也会运行 branch 0 保证 `all-gather` 有形状), `warmup` 路径本身也已有独立修复, 遗留的只是一个拒绝“自己能服务流量”的过期守卫。

实现拆解

实现按以下 5 步展开:

1. 移除 `input_validation.py` 中的过期守卫: `InputValidationStage.forward` (`python/sglang/multimodal_gen/runtime/pipelines_core/stages/input_validation.py`) 中原本对 `enable_cfg_parallel` and `not do_classifier_free_guidance` 抛 `ValueError` 的 `catch-all` 分支, 改为 `logger.warning_once(...)` 提示“单分支请求会被其余 CFG rank 冗余重算, 可用 `--cfg-parallel-size 1` 关闭”。这一改动的安全性依据是 `dispatcher` 的单分支语义: 无分支 rank 也运行 branch 0 使 `all-gather` 有形状, `reorder` 步骤把 `owner` 的预测分发给所有 rank。

2. 扩展日志 `once-helpers` 支持惰性格式参数: `logging_utils.py` 中 `info_once / warning_once` 从只接收 `msg` 改为接收 `msg, *args`, 并新增 `_log_once_guard` 统一完成 `msg % args` 格式化与去重判断。原因是 `cfg_parallel_utils` 中 `logger.warning_once("cfg_parallel_size=%d > n_branches=%d; ...", ws, n, ws - n)` 这类调用原本会抛 `TypeError`, 而该路径被旧守卫完全挡住从未执行过; 移除守卫后它变成首个单分支请求上的崩溃点。
3. 保留三个可观察契约: 其一, 去重改用 `_logged_once: set[tuple[str, int, str]]` (键为 `logger.name + level + 格式化文本`), 避免 `lru_cache` 按参数键控对张量持有强引用直至进程结束; 其二, `stacklevel=2` 是 `test_diffusion_bcg_padding` 字面断言的契约, 不能多包一层 helper 改变调用帧数; 其三, `_print_info_once.cache_clear = _logged_once.clear` 与 `_print_warning_once.cache_clear = _logged_once.clear` 保留旧 `lru_cache` 的 API 表面, 因为测试依赖它重置去重状态。另外回退了一次“改走 `logger.log` 路由”的尝试——测试会 patch `logger.warning`, 绕道会让 mock 计数为 0。
4. 同步 `server_args.py` 的提示文案: `ServerArgs._adjust_parallelism` 中自动开启 CFG 并行的日志补充说明“关闭 CFG 的请求仍会运行, 但其他 CFG rank 会冗余重算”, 引导用户用 `--cfg-parallel-size 1`、`--tp-size` 或 `--sp-degree / --ulysses-degree` 覆盖默认并行策略。
5. 测试配套: `test_cfg_parallel_warmup.py` 中将旧的 `TestInputValidationCfgParallelGuard` (断言拒绝) 改写成 `TestInputValidationCfgParallelSingleBranch` (断言放行), 并新增 `TestCfgParallelServesOneBranch`, 从“不拥有分支的 rank”视角 mock 掉 `rank/world_size/ 设备 /all-gather`, 钉住 `run_cfg_parallel` 在 `n_branches=1, world_size=2` 时仍返回 `owner` 预测这一安全性质; `test_logging_utils.py` 新增 `TestLogOnceTakesFormatArgs`, 覆盖惰性格式化、去重、弱引用防泄漏 (`weakref` 验证缓存不持有参数) 以及“记录文件名为 `logging_utils.py`”这一历史 wart。所有单测保持 CPU-only, 不加载模型、不做分布式初始化。

关键文件:

- `python/sglang/multimodal_gen/runtime/utils/logging_utils.py` (模块 日志工具; 类别 source; 类型 core-logic; 符号 `_log_once_guard, _print_info_once, _print_warning_once, info_once`): `once-helpers` 从只收消息改为支持 `%-style` 惰性格式参数, 并新增按格式化文本去重的 set 缓存, 修复守卫移除后必然触发的 `TypeError`; 同时保留 `stacklevel=2`、`cache_clear` 等被测试字面断言的可观察契约, 是本 PR 风险最集中、设计最精巧的文件。
- `python/sglang/multimodal_gen/runtime/pipelines_core/stages/input_validation.py` (模块 校验管线; 类别 source; 类型 core-logic; 符号 `InputValidationStage.forward`): 请求路径的核心变更: 把 CFG 并行模式下对非 CFG 请求的 `ValueError` 拒绝改为 `warning_once` 放行。这是本 PR 的行为开关, 改动依据是 dispatcher 的单分支安全语义 (#23736 多分支重构)。
- `python/sglang/multimodal_gen/test/unit/test_cfg_parallel_warmup.py` (模块 单元测试; 类别 test; 类型 test-coverage; 符号 `TestInputValidationCfgParallelSingleBranch, test_input_validation_accepts_cfg_parallel_without_cfg, TestCfgParallelServesOneBranch, _run_on_rank`): 测试主战场: 旧守卫测试被改写为接

受单分支请求，并新增 TestCfgParallelServesOneBranch 从无分支 rank 视角钉住 run_cfg_parallel 在 n_branches=1 / world_size=2 下仍返回 owner 预测的性质——这是移除守卫安全性的直接证据。

- python/sglang/multimodal_gen/test/unit/test_logging_utils.py (模块 单元测试; 类别 test; 类型 test-coverage; 符号 TestLogOnceTakesFormatArgs, test_warning_once_formats_lazy_args, test_info_once_formats_lazy_args, test_arguments_are_not_retained) : 新增 TestLogOnceTakesFormatArgs 用 assertLogs 与 weakref 分别锁定惰性格式化、按文本去重、不持有参数强引用、记录归属 logging_utils.py 等契约, 防止后续“修复”破坏 test_diffusion_bcg_padding。
- python/sglang/multimodal_gen/runtime/server_args/server_args.py (模块 服务参数; 类别 source; 类型 core-logic; 符号 ServerArgs._adjust_parallelism) : 自动开启 CFG 并行的启动日志补充了“关闭 CFG 的请求仍会运行但有冗余重算”的说明, 并给出 --cfg-parallel-size 1 等覆盖手段, 是用户体验层面的配套改动。

关键符号: _log_once_guard, _print_info_once, _print_warning_once, info_once, warning_once, InputValidationStage.forward, ServerArgs._adjust_parallelism

关键源码片段

python/sglang/multimodal_gen/runtime/utils/logging_utils.py

once-helpers 从只收消息改为支持 %-style 惰性格式参数, 并新增按格式化文本去重的 set 缓存, 修复守卫移除后必然触发的 TypeError; 同时保留 stacklevel=2、cache_clear 等被测试字面断言的可观察契约, 是本 PR 风险最集中、设计最精巧的文件。

```
# `logger.warning_once(msg, *args)` 通过 MethodType(_print_warning_once, logger)
# 绑定到 logger 实例, 因此调用者与 logger.warning 之间恰好只有一帧,
# 而 stacklevel=2 是 test_diffusion_bcg_padding 字面断言的可观察契约。
# 若在中间再插入一层 helper, 日志记录的文件名会变成 logging_utils.py,
# 破坏该断言 —— 所以去重不能放在第二个 lru_cache 函数上。
#
# 也不能在本函数上用 lru_cache: 以参数为 key 会对每个参数持有强引用
# 直到进程结束, 而这里的调用方会传入张量。因此改用“格式化后文本”的
# set 去重, 只保存字符串, 避免张量泄漏。
#
# 参数本身是新增能力: 旧实现只接收消息, 调用方若像标准 logger.warning
# 那样惰性格式化, 就会抛 TypeError 而不是打印日志 —— 而且总落在
# 极少执行的路径上, 从未被发现; 移除 CFG 守卫后该路径变成首请求必达。
_logged_once: set[tuple[str, int, str]] = set()
```

```
def _log_once_guard(logger: Logger, level: int, msg: str, *args: Any) -> str | None:
    """返回要打印的文本; 若该消息已打印过则返回 None。"""
    text = msg % args if args else msg
    key = (logger.name, level, text)
    if key in _logged_once:
        return None
    _logged_once.add(key)
```

```
return text
```

```
def _print_info_once(logger: Logger, msg: str, *args: Any) -> None:
    text = _log_once_guard(logger, logging.INFO, msg, *args)
    # stacklevel=2 是契约而非调参项; init_logger 还会把 warning 包成
    # 转发到 logger.log 的函数、多加一帧, 所以记录实际上归属本模块。
    if text is not None:
        logger.info(text, stacklevel=2)
```

```
def _print_warning_once(logger: Logger, msg: str, *args: Any) -> None:
    text = _log_once_guard(logger, logging.WARNING, msg, *args)
    if text is not None:
        logger.warning(text, stacklevel=2)
```

```
# 旧实现是 lru_cache 对象, `.cache_clear()` 属于其公开表面,
# test_diffusion_bcg_padding 依赖它重置去重状态, 故保留同名兼容入口。
_print_info_once.cache_clear = _logged_once.clear
_print_warning_once.cache_clear = _logged_once.clear
```

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

请求路径的核心变更: 把 CFG 并行模式下对非 CFG 请求的 ValueError 拒绝改为 warning_once 放行。这是本 PR 的行为开关, 改动依据是 dispatcher 的单分支安全语义 (#23736 多分支重构) 。

```
# 关闭 CFG 的请求在 CFG 并行下仍然可服务: dispatcher 把 branch 0 分给
# rank 0, 其它 rank 也会运行 branch 0, 保证 all-gather 有形状可用;
# reorder 步骤再把 owner 的预测分发给所有 rank, 结果就是单分支答案,
# 多余的 rank 只是冗余计算。
#
```

```
# 这里以前会抛异常。该守卫是为 warmup hang (#23198) 加的, 比多分支
# 重构 (#23736) 早两周; warmup 路径后来也长出了自己的修复 ——
# warmup builder 在启用 CFG 并行时强制打开 CFG。剩下的就是一个拒绝
# “自己能服务流量”的服务器: 运行时从模型默认采样参数自动开启
# CFG 并行, 于是 `sglang serve --num-gpus 2` 在一个默认启用 CFG 的
# 模型上, 会拒绝所有 guidance_scale=1.0 请求, 还把责任推给用户
# 从未传过的 --enable-cfg-parallel 参数。
```

```
if server_args.enable_cfg_parallel and not batch.do_classifier_free_guidance:
    logger.warning_once(
        "CFG parallelism is enabled but this request does not use "
        "classifier-free guidance (guidance_scale=%s, true_cfg_scale=%s), "
        "so it has one branch and the other CFG rank(s) recompute it "
        "redundantly. Pass --cfg-parallel-size 1 to spend those GPUs on "
        "another parallelism instead.",
        batch.guidance_scale,
        batch.true_cfg_scale,
```

)

python/sglang/multimodal_gen/test/unit/test_cfg_parallel_warmup.py

测试主战场：旧守卫测试被改写为接受单分支请求，并新增 TestCfgParallelServesOneBranch 从无分支 rank 视角钉住 run_cfg_parallel 在 n_branches=1 / world_size=2 下仍返回 owner 预测的性质——这是移除守卫安全性的直接证据。

```
class TestCfgParallelServesOneBranch(unittest.TestCase):
```

```
    """“接受单分支请求是安全”这一性质由 dispatcher 保证：
```

```
    单分支落在 2-rank CFG 组时：rank 0 拥有该分支，其它 rank 也运行它，
    从而 all-gather 有形状；reorder 步骤把 owner 的预测交给两个 rank。
    这里专门从“不拥有任何分支”的 rank 视角钉住该性质 —— 正是旧注释
    说会返回 None、并在 gloo broadcast 上挂半小时的那个 rank。
    """
```

```
def _run_on_rank(self, cfg_rank: int, n_branches: int = 1, world_size: int = 2):
    from sglang.multimodal_gen.runtime.distributed.cfg_policy import (
        CFGBranch,
        CFGPolicy,
    )
```

```
    mod = "sglang.multimodal_gen.runtime.distributed.cfg_parallel_utils"
    branches = [CFGBranch(f"b{i}", i == 0, {"tag": i}) for i in range(n_branches)]
    policy = CFGPolicy(branches=branches)
    seen: list[int] = []
```

```
def predict_fn(branch):
    seen.append(branch.kwargs["tag"])
    return torch.full((1, 2), float(branch.kwargs["tag"]))
```

```
# 真实 2-rank gather 每个 rank 返回一个张量；这里两个 rank 都
# 运行了同一分支，所以两份贡献值相同。
```

```
def fake_all_gather(t, dim=0, separate_tensors=False):
    return [t.clone() for _ in range(world_size)]
```

```
with (
    patch(f"{mod}.get_classifier_free_guidance_rank", return_value=cfg_rank),
    patch(
        f"{mod}.get_classifier_free_guidance_world_size",
        return_value=world_size,
    ),
    patch(f"{mod}.get_local_torch_device", return_value=torch.device("cpu")),
    patch(f"{mod}.cfg_model_parallel_all_gather", side_effect=fake_all_gather),
):
    from sglang.multimodal_gen.runtime.distributed.cfg_parallel_utils import (
        run_cfg_parallel,
    )
```

```

return run_cfg_parallel(policy, predict_fn), seen

def test_rank_without_a_branch_still_returns_the_owners_prediction(self):
    preds, seen = self._run_on_rank(cfg_rank=1)
    self.assertEqual(
        seen,
        [0],
        "无分支的 rank 也必须运行一个分支，否则 all-gather 没有形状可用",
    )
    self.assertEqual(len(preds), 1)
    self.assertIsNotNone(preds[0])
    self.assertTrue(torch.equal(preds[0], torch.zeros(1, 2)))

```

评论区精华

该 PR 没有正式 review 评论 (`review_comments_count: 0`)，但 5 个 commit 构成了非常完整的“自审闭环”，讨论精华几乎都沉淀在 commit message 与最终 CI 说明中：

- 去重键的选择 (commit 0a1eb29)：作者明确否定了两种初版方案——`lru_cache` 键控参数会对张量持有强引用，二次函数嵌套会破坏 `stacklevel` 契约；最终以“格式化文本的 `set`”落地。
- `logger.warning` vs `logger.log` 路由 (commit dbdf2a5)：`logger.log(WARNING, ...)` 看起来是等价简化，但“callers and tests patch `logger.warning`”，`test_diffusion_bcg_padding` 断言 `warning.assert_called_once_with(msg, stacklevel=2)`，绕道后 `mock` 计数为 0，直接导致 GPU shard 与 `coverage` 检查快速失败。
- 记录归属与 `stacklevel` 的悖论 (commit 1c56d4b)：作者原本断言“记录会命名调用者文件”，实测发现 `init_logger` 把 `warning` 替换为转发到 `logger.log` 的 `forwarder`，多出一帧，记录始终归属 `logging_utils.py`——这是先于本 PR 就存在的 wart，文档化它以避免后人“修坏”断言。
- CI 失败排查 (issue 评论)：作者逐条核对了 9 个 AMD/NPU 失败，结论是 AMD 侧为 MiniMax-H3 的 ROCm 平台缺口 (`RuntimeError: requires CPU, CUDA, MPS, XPU, or Ascend NPU`) 与无关的 LLM weight-transfer 测试，NPU 侧为 self-hosted runner 容器错误与 498MiB OOM，均与本 diff（只触及一个请求时校验分支和一条日志字符串）无关。
- `warning_once` 是否支持 %-style 惰性格式参数 (correctness): `helpers` 改为接收 `msg, *args`，先 `msg % args` 格式化再去重，与标准 `logger.warning` 契约一致；测试断言结果文本而非异常。
- `once` 去重缓存应键控于参数还是格式化文本 (performance): 采用 `_logged_once: set[tuple[str, int, str]]`，键为 `logger.name + level + 格式化文本`；测试用 `weakref` 验证参数不被缓存持有。
- `stacklevel=2` 与调用帧数契约 (correctness): 保持 `warning_once` 与 `logger.warning` 之间恰好一帧，`stacklevel=2` 作为契约保留；记录归属 `logging_utils.py` 是先于本 PR 的 wart，用测试文档化而非修复。
- `warning_once` 应路由 `logger.warning` 还是 `logger.log` (testing): 回退到直接调用 `logger.warning / logger.info`，保持方法路由可被测试观测。

- lru_cache 的 cache_clear API 表面是否保留 (design): 以 `_print_info_once.cache_clear = _logged_once.clear` 形式保留同名兼容入口。
- AMD/NPU CI 失败是否由本 diff 引入 (testing): 作者逐条核对后确认均与本 diff (一个请求时校验分支、一条日志字符串、日志 helper) 无关, NVIDIA 侧全绿; 但 ROCm/NPU 未被本 PR 实测仍是覆盖盲区。

风险与影响

- 风险: 具体风险如下:
- 冗余计算开销 (input_validation.py): 移除守卫后, CFG 并行服务器上的 `guidance_scale=1.0` 请求会触发所有 CFG rank 冗余执行同一分支。单分支请求越多, 浪费的 GPU 算力越大; 当前仅以 `warning_once` 与启动日志提示用户改用 `--cfg-parallel-size 1`, 没有自动降级或调度优化。
- 共享日志工具行为变更 (logging_utils.py): `info_once / warning_once` 是 `sglang/multimodal_gen` 包内共享的日志助手, 签名从 `(msg)` 扩为 `(msg, *args)`, 去重键也从 lru_cache 的 logger 对象身份改为 `(logger.name, level, text)`。两个同名不同实例的 logger 现在会互相去重; 同一文本但不同 % 参数会各打一次, 语义与旧版不完全等价。
- 惰性格式化新增异常面: `text = msg % args` 若调用方误传与占位符不匹配的参数会抛 `TypeError` (与标准 logging 行为一致), 此前该异常被 lru_cache 的“只传 msg”签名天然挡住。
- 历史遗留记录归属问题: `warning_once` 的日志记录文件名始终是 `logging_utils.py` 而非调用者, 诊断日志来源时可能误导, 本 PR 选择文档化而非修复 (修复会破坏 `test_diffusion_bcg_padding` 的字面断言)。
- 覆盖盲区: 作者明确说明在 4xB200 验证, 未在 ROCm 与 NPU 实测; CI 也确认 AMD/NPU 存在本 diff 之外的既有失败。
- 影响: 影响范围集中在 diffusion 多模态生成子系统的请求路径:
- 用户侧: 使用默认启用 CFG 的模型 (`Qwen-Image-2512`、`Qwen-Image-Edit-2511` 等) 以 `sglang serve --num-gpus 2` 默认命令启动时, `guidance_scale=1.0` 请求从 HTTP 500 变为 HTTP 200, 跨框架 benchmark 的 2 个 cell 恢复可用; CFG 请求路径不受影响 (实测输出字节数基本一致)。
- 系统侧: CFG 并行服务器将接受单分支请求, 代价是非 owner rank 的冗余 denoise 计算; 日志系统行为发生变化 (`once-helpers` 支持格式参数、去重键调整), 影响包内所有 `warning_once / info_once` 调用点。
- 团队侧: 新增约 80 个单元测试锁定 dispatcher 单分支性质与日志契约, 后续改动这些约束会被 CI 拦截; CI 矩阵中 AMD/NPU 的既有失败与本次 diff 无关, 但仍是需要单独跟进的基础设施问题。
- 风险标记: 过期守卫移除, 共享日志助手行为变更, 冗余计算开销, AMD/NPU 未实测

关联脉络

- PR #23198 (PR 正文引用, 标题未提供) warmup hang 修复: 本 PR 移除的守卫正是 2026-04-23 为 #23198 的 `--warmup-resolutions` hang 添加的; warmup 路径已有独立

修复（强制 CFG on），故守卫过期。

- PR #23736（PR 正文引用，标题未提供）多分支 dispatcher 重构：2026-05-07 的重构让 dispatcher 支持单分支（`dispatch_branches(1, 2) == [[0], []]`），是移除守卫安全性的前提；本 PR 的新测试专门钉住该性质。
- PR #39291 [Diffusion] Make the SP sequence gather pass contiguous shards: 同属 diffusion 模块的 bugfix；本 PR 最终 CI 评论将其合并时的 8 个 AMD 失败作为基线，证明本 PR 的 AMD 失败非新增。
- PR #39292 [Diffusion] Don't route an unreadable checkpoint into the native fallback: 同属 diffusion 模块的 bugfix 与测试配套；同样被 CI 评论引用为 AMD 失败基线，与本 PR 同批验证。