PR #27173 完整报告

sgl-project/sglang

Fix trace_modules gate disabling default trace contexts

合并时间：2026-06-04 02:03

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27173

执行摘要

- 一句话：重构 trace 模块过滤器，修复默认上下文被错误过滤
- 推荐动作：推荐阅读 trace.py 中 process_tracing_init 和 TraceReqContext.__init__的变更，以及 trace_wrapper.py 的简化。设计决策（模块过滤器不应在 context 初始化中依赖全局 server_args）值得借鉴。合并迅速，逻辑自洽，CV 风险低。

功能与动机

PR #23755 引入的 --trace-modules 过滤器导致两个 CI 失败：test_tracing.py 中默认 module_name 为空被过滤，未发出 spans；multimodal_gen/test/unit/test_disagg_trace.py 中 TraceReqContext.__init__调用 get_global_server_args() 在未设置全局 server_args 的进程中抛出 ValueError。

实现拆解

1. 在 trace.py 模块级别新增 global_trace_modules 变量，初始为 None。
2. 修改 process_tracing_init，接受可选 trace_modules 参数，解析为列表赋值给 global_trace_modules。
3. 修改 TraceReqContext.init，移除对 get_global_server_args() 的调用，直接检查 global_trace_modules：若 module_name 为空则始终不过滤；若在列表中则允许，否则禁用 tracing。
4. 更新 SRT 中四个调用 process_tracing_init 的地方（engine、scheduler、http_server、data_parallel_controller），传递 server_args.trace_modules。
5. 清理 diffusion 的 trace_wrapper.py：移除 SimpleNamespace hack 和 set_global_server_args_for_scheduler，直接调用 process_tracing_init 并传入 trace_modules='diffusion'。
6. 更新测试：移除 test_trace.py 中所有对 get_global_server_args 的 mock，新增 test_module_filtering 覆盖空 module_name 不过滤、已列入和未列入模块的过滤行为。

关键文件：

- python/sglang/srt/observability/trace.py（模块 追踪核心；类别 source；类型 core-logic；符号 process_tracing_init）：核心追踪模块；引入 global_trace_modules 变量，修改 process_tracing_init 和 TraceReqContext.__init__过滤逻辑

- python/sglang/multimodal_gen/runtime/utils/trace_wrapper.py（模块 扩散追踪；类别 source；类型 dependency-wiring）：扩散 trace 初始化简化，移除了 get_global_server_args hack
- test/registered/unit/observability/test_trace.py（模块 单元测试；类别 test；类型 test-coverage；符号 _mock_get_global_server_args, test_module_filtering）：新增 test_module_filtering 覆盖模块过滤逻辑；移除不再需要的 get_global_server_args mock
- python/sglang/srt/entrypoints/engine.py（模块 引擎入口；类别 source；类型 core-logic）：SRT 引擎入口的 trace 初始化调用点，添加传递 trace_modules
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic）：调度器进程的 trace 初始化调用点，添加传递 trace_modules
- python/sglang/srt/entrypoints/http_server.py（模块 HTTP 服务；类别 source；类型 core-logic）：HTTP 服务进程的 trace 初始化调用点，添加传递 trace_modules
- python/sglang/srt/managers/data_parallel_controller.py（模块 数据并行；类别 source；类型 entrypoint）：数据并行控制器进程的 trace 初始化调用点，添加传递 trace_modules

关键符号：process_tracing_init, init_diffusion_tracing, TraceReqContext.init, test_module_filtering

关键源码片段

python/sglang/srt/observability/trace.py

核心追踪模块；引入 global_trace_modules 变量，修改 process_tracing_init 和 TraceReqContext.__init__过滤逻辑

```
# 模块级全局变量，类似 global_trace_level
# None 表示不过滤
_global_trace_modules: Optional[List[str]] = None

def process_tracing_init(
    otlp_endpoint: str,
    server_name: str,
    trace_modules: Optional[str] = None,
) -> None:
    global _global_trace_modules
    if trace_modules is not None:
        # 将逗号分隔的模块名解析为数组
        _global_trace_modules = [
            m.strip() for m in trace_modules.split(",") if m.strip()
        ]
    # ... 原有 OTel 初始化逻辑保持不变 ...

class TraceReqContext:
    def __init__(self, rid: str, module_name: str = ""):
        self.tracing_enable = (
            opentelemetry_initialized and self.trace_level > 0
```

```
        )
        if self.tracing_enable:
            # 只有非空 module_name 才受 _global_trace_modules 过滤
            if (
                module_name
                and _global_trace_modules is not None
                and module_name not in _global_trace_modules
            ):
                self.tracing_enable = False
```

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

扩散 trace 初始化简化，移除了 get_global_server_args hack

```
def init_diffusion_tracing(server_args, thread_label: str):
    if not server_args.enable_trace:
        return

    from sglang.srt.observability.trace import (
        process_tracing_init,
        trace_set_thread_info,
    )

    # 直接传递 trace_modules 参数，不再依赖 SRT 全局 server_args
    process_tracing_init(
        server_args.otlp_traces_endpoint,
        "sglang-diffusion",
        trace_modules=DIFFUSION_TRACE_MODULE, # "diffusion"
    )
    trace_set_thread_info(thread_label)
```

test/registered/unit/observability/test_trace.py

新增 test_module_filtering 覆盖模块过滤逻辑；移除不再需要的 get_global_server_args mock

```
def test_module_filtering(self):
    """global_trace_modules 仅过滤显式指定的模块。"""
    orig_modules = mod._global_trace_modules
    mod._global_trace_modules = ["request"]
    try:
        # 默认空 module_name 不被过滤
        ctx = TraceReqContext(rid="req-1")
        self.assertTrue(ctx.tracing_enable)
        # 已列入的模块被追踪
        ctx = TraceReqContext(rid="req-1", module_name="request")
        self.assertTrue(ctx.tracing_enable)
        # 未列入的模块被过滤
        ctx = TraceReqContext(rid="req-1", module_name="mooncake")
        self.assertFalse(ctx.tracing_enable)
    finally:
```

```
        mod._global_trace_modules = orig_modules
```

评论区精华

无公开 review 讨论。PR body 描述了原设计缺陷：模块过滤器在 TraceReqContext.__init__ 中使用 get_global_server_args()，既耦合全局状态又导致默认 module_name 被错误过滤。作者选择将过滤器提升至模块级全局变量，与 trace_level 模式一致，并在所有调用点显式传递 trace_modules。

- 模块过滤器放置位置不当导致回归 (design): 作者将过滤提升至模块级全局变量，让 process_tracing_init 负责接收 trace_modules，消除了对 get_global_server_args 的依赖。

风险与影响

- 风险：重构改变了 module_name 为空时的过滤语义：之前空 module_name 会进入过滤判断（'' not in trace_modules 导致 tracing 关闭），现在空 module_name 直接不过滤，默认上下文 tracking 更合理。此改动影响所有使用默认 module_name 创建 TraceReqContext 的地方（主要是请求主上下文）。扩散模块的 trace 初始化路径简化，不再依赖 SRT 全局 server_args，降低初始化出错风险。
- 影响：对用户：默认启用 trace 但不指定 --trace-modules 时，所有上下文均产生 spans（与 #23755 之前行为一致）；指定 --trace-modules 时，仅明确列出的模块会出 spans，但默认空 module_name 的上下文不受限制。对系统：消除了扩散运行时对 SRT 全局 server_args 的侵入式依赖。对团队：tracing 过滤器设计更清晰，新模块接入只需在调用 process_tracing_init 时传入 trace_modules 参数。
- 风险标记：全局状态初始化时序，默认上下文过滤语义变更

关联脉络

- PR #23755 [SGLang Tracing] Add pd disaggregation mooncake backend tracing: 该 PR 引入的 --trace-modules 过滤器是本 PR 修复的根源