PR #33787 完整报告

sgl-project/sglang

[diffusion] server: gate /health and /health_generate on warmup completion (#33719)

合并时间：2026-08-07 17:59

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33787

执行摘要

- 一句话：/health 门控 warmup，新增 /liveness 区分存活与就绪
- 推荐动作：值得精读。该 PR 展示了一个经典的微服务健康检查语义拆解案例（liveness vs readiness），并解决了由端点语义混用导致的启动死锁问题。关注点：/health 门控的实现方式、内部探测为何必须切换 /liveness、以及文档中对 warmup-mode 行为差异的说明。

功能与动机

Issue #33719 指出 /health 在 warmup_done.wait() 完成前可访问，导致编排器看到绿色健康检查而提前路由流量，实际请求在中间件后阻塞 264s。PR 描述明确：'/health and /health_generate are exempt from the wait_for_server_warmup middleware (via SERVER_WARMUP_BYPASS_PATHS) so they respond immediately even during warmup — but the handlers themselves never checked app.state.server_warmup_done, so they always returned 200 regardless of actual warmup readiness.'

实现拆解

1. 改造 /health 与 /health_generate 为 readiness 端点：在 python/sglang/multimodal_gen/runtime/entrypoints/http_server.py 中，health 函数改为接收 Request 并检查 request.app.state.server_warmup_done.is_set()，未就绪时返回 Response(status_code=503)；health_generate 直接委托 health，保留兼容语义。
2. 新增 /liveness 端点：同一文件新增 liveness 函数，始终返回 200，表示 HTTP 进程存活，进入 SERVER_WARMUP_BYPASS_PATHS 以绕过中间件。
3. 调整内部 warmup 探测：将 _wait_until_http_ready 重命名为 _wait_until_http_live，轮询目标从 /health 改为 /liveness；_run_server_warmup_after_http_ready 同步改为 _run_server_warmup_after_http_live，并在 lifespan 中更新创建任务处的调用名。这避免“warmup 等待自身完成”的死锁——因为 /health 在 warmup 完成前返回 503，而 warmup 完成又依赖探测返回。
4. 新增单元测试：python/sglang/multimodal_gen/test/unit/test_health_warmup_gate.py，用 SimpleNamespace 伪造 request 验证三个端点 warmup 前后的状态码，并用 _FakeAsyncClient 验证 _wait_until_http_live 的轮询 URL 与重试行为。
5. 更新文档：docs/docs/sglang-diffusion/deployment_cookbook.mdx 增加 Kubernetes startup/readiness/liveness 探针配置示例；docs/docs/sglang-diffusion/api/cli.mdx 说明

端点契约、warmup-mode 行为及“不要用 /health 做 liveness 探针”的警告。

关键文件：

- python/sglang/multimodal_gen/runtime/entrypoints/http_server.py（模块 健康端点；类别 source；类型 core-logic；符号 _wait_until_http_ready, _wait_until_http_live, _run_server_warmup_after_http_ready, _run_server_warmup_after_http_live）：核心变更文件：实现 /health 与 /health_generate 的 warmup 门控，新增 /liveness 端点，并调整内部探测避免死锁。
- python/sglang/multimodal_gen/test/unit/test_health_warmup_gate.py（模块 单元测试；类别 test；类型 test-coverage；符号 _make_request, TestHealthWarmupGate, test_liveness_returns_200_before_warmup, test_health_returns_503_before_warmup）：新增单元测试，覆盖三个端点在 warmup 前后的状态码以及 /liveness 轮询逻辑。
- docs/docs/sglang-diffusion/deployment_cookbook.mdx（模块 部署文档；类别 other；类型 configuration）：增加 Kubernetes 探针配置示例，指导用户区分 liveness 与 readiness。
- docs/docs/sglang-diffusion/api/cli.mdx（模块 CLI 文档；类别 other；类型 documentation）：文档化端点契约与 warmup-mode 行为，明确 /health 不能当 liveness。

关键符号：health, liveness, health_generate, _wait_until_http_live, _run_server_warmup_after_http_live

关键源码片段

python/sglang/multimodal_gen/runtime/entrypoints/http_server.py

核心变更文件：实现 /health 与 /health_generate 的 warmup 门控，新增 /liveness 端点，并调整内部探测避免死锁。

```
# python/sglang/multimodal_gen/runtime/entrypoints/http_server.py
# 健康检查端点核心实现：liveness 与 readiness 分离。

# /liveness 只负责报告“HTTP 进程活着”，不依赖模型状态，
# 因此 warmup 期间也稳定返回 200，适合作为编排器存活探针。
@health_router.get("/liveness")
async def liveness():
    """Report that the HTTP server is accepting requests."""
    return {"status": "ok"}


# /health 是 readiness：warmup 未完成时返回 503。
# 注意：不能把 /health 同时当 liveness 用，因为 warmup 任务本身会
# 等待 HTTP 就绪后再启动，若 /health 在 warmup 完成前保持 503，
# warmup 会等待一个只有自己才能置位的事件，形成启动死锁。
@health_router.get("/health")
async def health(request: Request):
    """Report readiness for normal inference traffic."""
    if not request.app.state.server_warmup_done.is_set():
        return Response(status_code=503)
```

```
    return {"status": "ok"}


# 兼容性别名：直接复用 /health 的读取逻辑，不真正发起生成。
@health_router.get("/health_generate")
async def health_generate(request: Request):
    """Compatibility readiness endpoint; no generation is issued."""
    return await health(request)
```

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

新增单元测试，覆盖三个端点在 warmup 前后的状态码以及 /liveness 轮询逻辑。

```
# python/sglang/multimodal_gen/test/unit/test_health_warmup_gate.py
# 验证端点状态与 warmup 事件的关系。
class TestHealthWarmupGate(unittest.IsolatedAsyncioTestCase):
    async def test_liveness_returns_200_before_warmup(self):
        # liveness 不依赖 warmup 状态，始终 200
        self.assertEqual(await liveness(), {"status": "ok"})

    async def test_health_returns_503_before_warmup(self):
        warmup_done = asyncio.Event() # 未 set，表示 warmup 未完成
        resp = await health(_make_request(warmup_done))
        self.assertEqual(resp.status_code, 503)

    async def test_health_returns_200_after_warmup(self):
        warmup_done = asyncio.Event()
        warmup_done.set() # 模拟 warmup 完成
        resp = await health(_make_request(warmup_done))
        self.assertEqual(resp, {"status": "ok"})

    async def test_health_generate_returns_503_before_warmup(self):
        # 兼容别名同样受 warmup 门控
        warmup_done = asyncio.Event()
        resp = await health_generate(_make_request(warmup_done))
        self.assertEqual(resp.status_code, 503)

    async def test_health_generate_returns_200_after_warmup(self):
        warmup_done = asyncio.Event()
        warmup_done.set()
        resp = await health_generate(_make_request(warmup_done))
        self.assertEqual(resp, {"status": "ok"})
```

评论区精华

核心讨论围绕 mickqian 对 /liveness 拆分的建议展开。作者在 issue 评论中说明：'/health couldn't serve both roles because startup itself consumed it. Warmup waits for a green health check before it runs, so once /health stayed red until warmup finished, warmup was waiting on a condition only it could satisfy — 120s of retries, then the error path SIGTERMs the process, and three jobs sat until the workflow timeout.' 最终结论是采用

/liveness 作为独立存活端点，/health 只负责就绪判定。

- 是否应引入 /liveness 端点 (design): 采用 /liveness 方案：/liveness 负责存活，/health 只负责就绪，内部 warmup 探测改轮询 /liveness。

风险与影响

- 风险：
  1. 端点语义变更：已有编排器若将 /health 当作存活探针，在 server warmup 期间会收到 503，可能被误判为崩溃；但这是预期行为，且文档已建议改用 /liveness 做存活探测。
  2. 启动死锁风险：已通过将内部探针改为 /liveness 规避，但需确保所有调用点（如 lifespan 中的任务创建）都已同步更新，否则仍可能死锁（当前代码已覆盖）。
  3. warmup_mode 非 server 时行为变化：文档明确在 --warmup-mode off/request 下 /health 始终 200，不承诺编译等首请求工作已完成，编排器需理解此契约。
  4. 兼容性：/health_generate 改为委托 health，行为与之前不同（原先总是 200），依赖此端点的旧集成需适配，风险较低。 - 影响：对用户：Kubernetes、Modal 等编排器部署的 Diffusion 服务在 warmup 期间会正确显示未就绪，避免流量提前进入导致的黑洞；新增 /liveness 可独立探活。对系统：启动阶段行为更准确，warmup 失败时 /health 持续 503，便于 orchestrator 感知。对团队：确立了 liveness/readiness 分离的端点契约，为后续服务治理提供基础。 - 风险标记：端点语义变更，启动死锁（已修复），编排器兼容性

关联脉络

- 暂无明显关联 PR