

PR #56941 完整报告

vllm-project/vllm

[CI] Drop root privileges for the OTel GPU sampler

合并时间: 2026-09-15 12:20

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56941>

执行摘要

- 一句话: OTel GPU 采样器降权运行, 失败则禁用采样
- 推荐动作: 值得精读, 但重点不在代码量, 而在“权限边界如何失败”的设计: 一是 fail-closed 而不是 fail-back-to-root 的选择, 二是通过 stat 推导 checkout 属主、再用 setpriv ... -- test -w 做真实写权限探测, 三是只给采样器而非整个命令链路降权。建议读者重点关注 `_ci_otel_prepare_gpu_user()` 的校验分支 (case "\$1:\$2" in) 与 `ci_otel_start()` 中三分支启动逻辑, 以及测试如何用 shell stub 精确断言 `--reuid=997 --regid=996 --clear-groups --no-new-privs` 与 `setpriv` 被调用 2 次 (一次探测 + 一次启动)。若要在自己仓库复用该模式, 需要额外确认 CI 镜像自带 `setpriv`, 并为“采样静默消失”补一条告警。

功能与动机

PR body 指出: Buildkite #88719 的 H200 CUDAGraph 作业 (01a09ded-...) 把后台 `ci_gpu.py` OTel 采样器当作 root 运行, 在该 revision (df31f348e4e0) 下其 CDI/MIG 路径经包导入 `vllm.third_party.pynvml`, 在挂载 checkout 里留下 34 个 root 属主的字节码 / 缓存条目, 导致 h200-ci-6-23 上的下一个作业无法清理该 checkout。此前 PR #56747 已移除整包导入、PR #56766 已禁用 helper 字节码写入, 但“GPU 采样器进程本身仍以 root 运行”这一权限边界没有封住, 因此本次要“closes this privilege boundary so a later helper regression cannot poison the checkout again”。

实现拆解

1. 新增降权身份解析函数: 在 `.buildkite/scripts/ci-otel/ci_otel.sh` 中新增 `_ci_otel_prepare_gpu_user()`。函数先清空 `_CI_INFRA_GPU_UID` / `_CI_INFRA_GPU_GID`, 若 `id -u` 不为 0 则直接返回 0 (非 root 无需处理, 沿用旧路径); 否则要求 `setpriv` 存在, 并把 workspace 目录解析为 `CI_INFRA_OTEL_WORKSPACE_DIR` (默认 `/workdir`)。
2. 从 checkout 属主推导 UID/GID 并做安全校验: 用 `stat -c '%u %g'` 取属主字段, 要求正好两列, 并用 `case "$1:$2" in *[!0-9:]* | 0: * | *: 0) return 1` 拒绝非数字、UID 为 0 或 GID 为 0 的情况——即“推导不出可信的非 root 身份就失败”, 而不是退回 root 运行。
 1. 按最小权限开放采样器所需的写路径: 把私有 `span pool` `CI_INFRA_OTEL_SPOOL_DIR` chown 给该身份并 `chmod 0700`; 若 runtime 目录由脚本自己创建 (`_CI_INFRA_OTEL_OWNS_RUNTIME=1`), 则 `chmod 0711` 允许穿越但

禁止列目录。最后用 `setpriv ... --test -w "${CI_INFRA_OTEL_SPOOL_DIR}"` 做一次真实写权限探测，探测失败同样让函数返回非零。

2. 改造 `ci_otel_start()` 的启动分支：原有 GPU 采样启动条件（`CI_INFRA_GPU_SAMPLING != 0`、`ci_gpu.py` 存在、`nvidia-smi` 可用）不变，但在其内部改为 `if _ci_otel_prepare_gpu_user; then`：解析到非 root 身份时用 `setpriv --reuid/--regid --clear-groups --no-new-privs` 启动采样器，保留 `PYTHONDONTWRITEBYTECODE=1`；身份为空（本来非 root）时走原来的直接启动；函数失败则向 `stderr` 打印 `non-root GPU sampler unavailable; sampling disabled` 并跳过采样。根身份仍然保留给命令记录器与最终 flush，因此降权只作用于采样器这一个子进程。
3. 测试与验证配套：`.buildkite/scripts/ci-otel/tests/test_ci_otel.py` 新增 `_write_executable` 辅助函数和参数化用例 `test_root_shell_runs_gpu_sampler_as_checkout_owner`，用 `PATH` 前置的 `id/stat/setpriv/chown/chmod/python3/nvidia-smi` stub 覆盖 `"997 996"`（期望降权并启动采样）与 `"0 0"`（期望失败关闭、不启动采样且 `stderr` 出现禁用提示）两种情况。PR 侧执行 `uv run --no-project --python 3.12 --with pytest pytest -q .buildkite/scripts/ci-otel/tests/test_ci_otel.py` 得到 62 passed，并通过 `shellcheck`、`ruff`、`mypy`、`SPDX` 等 pre-commit 检查。

关键文件：

- `.buildkite/scripts/ci-otel/ci_otel.sh`（模块 采样器权限；类别 `infra`；类型 `core-logic`；符号 `_ci_otel_prepare_gpu_user`, `ci_otel_start`, `_CI_INFRA_GPU_UID`, `_CI_INFRA_GPU_GID`）：核心改动：新增 `_ci_otel_prepare_gpu_user()` 从 `/workdir` 属主推导非 root 身份，并改造 `ci_otel_start()` 用 `setpriv` 启动 `ci_gpu.py`，推导失败时禁用 GPU 采样。
- `.buildkite/scripts/ci-otel/tests/test_ci_otel.py`（模块 采样器测试；类别 `test`；类型 `test-coverage`；符号 `_write_executable`, `test_root_shell_runs_gpu_sampler_as_checkout_owner`）：新增参数化单测 `test_root_shell_runs_gpu_sampler_as_checkout_owner`，用 `shell stub` 同时覆盖降权成功与 root 属主失败关闭两条路径，是本 PR 唯一的行为护栏。

关键符号：`_ci_otel_prepare_gpu_user`, `ci_otel_start`, `_write_executable`, `test_root_shell_runs_gpu_sampler_as_checkout_owner`

关键源码片段

`.buildkite/scripts/ci-otel/ci_otel.sh`

核心改动：新增 `_ci_otel_prepare_gpu_user()` 从 `/workdir` 属主推导非 root 身份，并改造 `ci_otel_start()` 用 `setpriv` 启动 `ci_gpu.py`，推导失败时禁用 GPU 采样。

```
# Docker 里的测试命令以 root 运行，而 /workdir 属于 Buildkite checkout 用户。
# 让后台 GPU 采样器以那个非特权身份运行，这样将来任何项目代码 import
# 都无法再往 checkout 里写 root 属主的文件。
_ci_otel_prepare_gpu_user() {
    _CI_INFRA_GPU_UID=""
    _CI_INFRA_GPU_GID=""
    # 非 root：无需降权，返回 0 让调用方沿用原来的直接启动路径
```

```

[ "$(id -u 2>/dev/null || :)" = "0" ] || return 0
# 缺 setpriv 就失败关闭, 宁可停采样也不以 root 采样
command -v setpriv >/dev/null 2>&1 || return 1

_CI_INFRA_OTEL_WORKSPACE_DIR="${CI_INFRA_OTEL_WORKSPACE_DIR:-/workdir}"
[ -d "${CI_INFRA_OTEL_WORKSPACE_DIR}" ] || return 1
# GNU stat 的属主字段是纯数字、无空白, 这里靠 word splitting 取两列
# shellcheck disable=SC2046
set -- $(stat -c '%u %g' -- "${CI_INFRA_OTEL_WORKSPACE_DIR}" 2>/dev/null || :)
[ "$#" -eq 2 ] || return 1
# 只接受可信的非 root 身份: 字段非数字、UID=0 或 GID=0 一律拒绝
case "$1:$2" in
    *[!0-9:]* | 0:* | *:0) return 1 ;;
esac

_CI_INFRA_GPU_UID="$1"
_CI_INFRA_GPU_GID="$2"
# 私有 span spool 交给该身份; root 的命令记录器与收尾 flush 仍可访问
chown "${CI_INFRA_GPU_UID}:${CI_INFRA_GPU_GID}" \
    "${CI_INFRA_OTEL_SPOOL_DIR}" || return 1
chmod 0700 "${CI_INFRA_OTEL_SPOOL_DIR}" || return 1
if [ "${CI_INFRA_OTEL_OWNS_RUNTIME}" = "1" ]; then
    # 允许穿越到子目录 spool, 但不允许列 runtime 目录内容
    chmod 0711 "${CI_INFRA_OTEL_RUNTIME_DIR}" || return 1
fi
# 探测: 确认新身份确实能写 spool, 探测不过同样失败关闭
setpriv --reuid="${CI_INFRA_GPU_UID}" \
    --regid="${CI_INFRA_GPU_GID}" --clear-groups --no-new-privs \
    -- test -w "${CI_INFRA_OTEL_SPOOL_DIR}"
}

# ci_otel_start() 中真正启动采样器的分支 (条件与旧版一致, 只是包了一层降权)
if [ "${CI_INFRA_GPU_SAMPLING:-1}" != "0" ] &&
    [ -f "${CI_INFRA_OTEL_DIR}/ci_gpu.py" ] &&
    command -v nvidia-smi >/dev/null 2>&1; then
    if _ci_otel_prepare_gpu_user; then
        if [ -n "${CI_INFRA_GPU_UID}" ]; then
            # root 场景: 只降权后台采样器, 测试命令与记录器仍是 root
            PYTHONDONTWRITEBYTECODE=1 setpriv \
                --reuid="${CI_INFRA_GPU_UID}" --regid="${CI_INFRA_GPU_GID}" \
                --clear-groups --no-new-privs -- \
                "${CI_INFRA_OTEL_PYTHON}" "${CI_INFRA_OTEL_DIR}/ci_gpu.py" "$$" \
                </dev/null &
        else
            # 本来就不是 root: 保持原有启动方式, 不引入 setpriv 依赖
            PYTHONDONTWRITEBYTECODE=1 \
                "${CI_INFRA_OTEL_PYTHON}" "${CI_INFRA_OTEL_DIR}/ci_gpu.py" "$$" \
                </dev/null &
        fi
    fi

```

```

_CI_INFRA_GPU_PID=$!
else
# 失败关闭：禁用 GPU 采样，但测试命令照常执行、退出码不受影响
echo "vLLM CI OTel: non-root GPU sampler unavailable; sampling disabled" \
    >&2 || :
fi
fi

```

[.buildkite/scripts/ci-otel/tests/test_ci_otel.py](#)

新增参数化单测 `test_root_shell_runs_gpu_sampler_as_checkout_owner`，用 shell stub 同时覆盖降权成功与 root 属主失败关闭两条路径，是本 PR 唯一的行为护栏。

```

def _write_executable(path: Path, body: str) -> None:
    """写一个 0755 的 shell stub，配合 PATH 前置来劫持真实命令。"""
    path.write_text(f"#!/bin/sh\n{body}")
    path.chmod(0o755)

# "997 996" (checkout 属主非 root) → 期望降权并启动采样器
# "0 0" (checkout 属主是 root) → 期望失败关闭，完全不启动采样器
@pytest.mark.parametrize(
    "workspace_owner,expect_sampler",
    [("997 996", True), ("0 0", False)],
)
def test_root_shell_runs_gpu_sampler_as_checkout_owner(
    tmp_path, workspace_owner, expect_sampler
):
    bin_dir = tmp_path / "bin"
    workspace = tmp_path / "workdir"
    runtime = tmp_path / "runtime"
    log = tmp_path / "calls.log"
    bin_dir.mkdir()
    workspace.mkdir()

    # 伪造 id -u 返回 0，迫使脚本认为自己在 root 下运行、走降权分支
    _write_executable(
        bin_dir / "id",
        '[ "$1" = "-u" ] && { echo 0; exit 0; }; exec /usr/bin/id "$@"\n',
    )

    # 伪造 stat，用参数化值模拟 /workdir 的属主 UID/GID
    _write_executable(bin_dir / "stat", f"printf '%s\\n' '{workspace_owner}'\n")
    for command in ("chown", "chmod"):
        _write_executable(bin_dir / command, "exit 0\n")

    # 伪造 setpriv：先记录完整参数，再跳过 -- 之后原样 exec，模拟真实降权
    _write_executable(
        bin_dir / "setpriv",
        'printf "setpriv %s\\n" "$*" >> "$CI_OTEL_TEST_LOG"\n'
        'while [ "$#" -gt 0 ] && [ "$1" != "--" ]; do shift; done\n'
        '[ "$#" -gt 0 ] && shift\n'
    )

```

```

    'exec "$@"\n',
)
# 伪造 python3: 回答 span 上下文查询, 并在被要求跑 ci_gpu.py 时打点
_write_executable(
    bin_dir / "python3",
    '[ "$1" = "-c" ] && exit 0\n'
    '[ "$2" = "new-context" ] && {\n'
    '  echo "01010101010101010101010101010101010101010101010101010101 0202020202020202 - 1"\n'
    '  exit 0\n'
    "}\n"
    'case "$1" in *ci_gpu.py) echo gpu >> "$CI_OTEL_TEST_LOG";; esac\n'
    "exit 0\n",
)
_write_executable(bin_dir / "nvidia-smi", "exit 0\n")
shell = (
    f'. "{SCRIPTS_DIR} / "ci_otel.sh"'; '
    f"ci_otel_start 1 {quoted('true')}; ci_otel_finish 0"
)

result = subprocess.run(
    ["/bin/sh", "-c", shell],
    check=False,
    capture_output=True,
    text=True,
    env={
        **os.environ,
        "PATH": f"{bin_dir}:{os.environ['PATH']}",
        "CI_INFRA_OTEL_DIR": str(SCRIPTS_DIR),
        "CI_INFRA_OTEL_RUNTIME_DIR": str(runtime),
        "CI_INFRA_OTEL_WORKSPACE_DIR": str(workspace),
        "CI_OTEL_TEST_LOG": str(log),
    },
)

assert result.returncode == 0, result.stderr
calls = log.read_text() if log.exists() else ""
if expect_sampler:
    expected = "--reuid=997 --regid=996 --clear-groups --no-new-privs --"
    # 2 次 setpriv: 一次是 spool 写权限探测, 一次是采样器实际启动
    assert calls.count("setpriv") == 2
    assert calls.count(expected) == 2
    assert "gpu" in calls # 确认 ci_gpu.py 真的被拉起
else:
    # 失败关闭: 没有 setpriv 调用, 也没有任何采样器进程
    assert calls == ""
    assert "non-root GPU sampler unavailable" in result.stderr

```

评论区精华

Review 区没有实质人工逐行评审：唯一一条 review 来自 `claude[bot]`，内容是仓库的 Claude Code 评审配置说明（“Comment `@claude review` for a one-time review”），`review_comments_count` 为 0。真正的技术交锋集中在 PR body 与 issue 评论里，且基本由作者 khluu 自证：

“This change closes this privilege boundary so a later helper regression cannot poison the checkout again.” (PR body)

作者主动披露了验证边界，值得肯定：

“PR-sourced Buildkite commands intentionally omit the CI-OTel wrapper, so #89008 validates the exact tree and leaves the checkout clean but does not produce GPU samples. The full runtime sampler proof must come from the first natural main job after merge.”

合并后补充的运行时证据也较完整：合并提交 `a491b6039d8d` 对应的 #89013 (CPU-only) 与首个 GPU 构建 #89014 全部通过，“the OTel backend retained 1,033 / 1,415 / 1,158 / 1,220 GPU samples for those jobs, proving the unprivileged sampler remained operational end to end”，且四台 H200 的 checkout slot 均为 0 个 root 属主条目。另一方面，PR body 明确写有“Human line review is required before marking it ready”，但记录中未见人工 line review，PR 由作者本人 (`merged_by: khluu`) 合并。

- 失败关闭策略：推导不出非 root 身份就禁用 GPU 采样 (design): 选择失败关闭而非回退 root，符合“避免再次污染 checkout”的目标；代价是采样可能静默消失，作者通过在 stderr 打印提示与单测断言来降低排查成本。
- PR 构建无法产出 GPU 采样，运行时证明只能来自合并后 (testing): 合并后补齐：合并构建 #89013 通过，首个 GPU 构建 #89014 四个 H200 作业全过，OTel 后端保留 1,033 / 1,415 / 1,158 / 1,220 条 GPU 样本，四台机器 checkout slot 均为 0 个 root 属主条目。
- 与 #56747 / #56766 的职责划分（去根 vs 去字节码）(other): 本次是三段加固的最后一环：前两个 PR 消除写入内容，本 PR 消除写入权限，形成互补而非重复。
- AI 参与与人工逐行评审的缺口 (other): 验证证据（62 项单测、pre-commit、H200 实测、合并后样本数）相对充分，但“人工 line review”这一自设门槛在流程记录里没有被满足，属于流程性遗留。

风险与影响

- 风险：
 1. 采样静默降级导致观测数据缺口：失败关闭路径只向 stderr 打印 non-root GPU sampler unavailable; sampling disabled，测试命令返回码不受影响。如果某个 CI 镜像缺少 setpriv (util-linux)，或 /workdir 属主恰为 root（例如未来挂载方式变化），GPU 采样会整体消失而不会让作业变红，指标断档需要靠人工发现。
 2. 依赖 GNU 工具与特定挂载布局：stat -c '%u %g' 是 GNU 语义，set -- \$(stat ...) 依赖 setpriv、stat 与 word splitting；脚本已用 # shellcheck disable=SC2046 显式豁免。若 workspace 路径解析异常或 CI_INFRA_OTEL_WORKSPACE_DIR 指向非 checkout 目录，会走到失败关闭分支而非报错。

3. spool 属主与权限语义变化: `chown + chmod 0700` 把 `CI_INFRA_OTEL_SPOOL_DIR` 交给非 root 身份, `chmod 0711` 让 runtime 目录可被穿越。虽然 root 仍可写入, 但 spool 目录的属主不再一致, 后续如有其它以 root、依赖 0600/0700 语义读写 spool 的逻辑 (例如命令记录器与 flush 路径), 需要确认不会因为属主变化而出现行为差异——PR 未展示这部分回归验证。
4. 降权范围有限: 真正执行测试命令 (可能 import 项目代码) 的进程仍是 root, 本 PR 只把后台采样器拉出 root 边界; PR body 也承认这是针对“sampler 自身”的加固, root 污染 checkout 的根本面仍需后续持续收敛。
5. 缺少人工逐行评审: 改动虽小 (+129/-3), 但涉及 CI 权限边界与 fail-closed 语义, 仅凭自动化测试与作者自证合并, 长期看建议补一位 CI/infra owner 的明确 review。
 - 影响: 影响面严格限定在 Buildkite CI: 所有启用 OTel 包装、且以 root 运行 Docker 测试命令的 GPU 作业都会改变后台采样进程的身份, H200 系列作业是首要受益者。对用户与生产推理路径零影响——该脚本不参与 vLLM 运行时。对 CI 可靠性是正向收益: 同一台机器上前后作业之间不再互相污染 checkout, 减少了此前那种“上一个作业留下 root 文件、下一个作业清理失败”的连锁失败。对团队而言, 代价是观测数据可用性变成条件性的 (依赖 `setpriv` 与非 root checkout 属主), 需要在 CI 镜像变更时把 `setpriv` 当作隐含依赖维护。
 - 风险标记: 采样可能静默降级, 缺少人工逐行评审, 依赖 `setpriv` 与 `GNU stat`, 目录权限由 0700 放宽为 0711, 降权范围仅限采样器

关联脉络

- PR #56747 移除 OTel 采样器中的整包 `vllm.third_party.pynvml` 导入 (标题依据 PR body 描述推断): PR body 明确引用: 该 PR 移除了 CDI/MIG 路径的整包导入, 是 root 属主缓存条目问题的第一段修复, 本 PR 在其基础上继续封堵权限边界。
- PR #56766 禁用 helper 的字节码写入 (标题依据 PR body 描述推断): PR body 明确引用: 该 PR 通过抑制字节码写入减少 checkout 污染, 与本 PR 的降权方案互补; 作者专门说明两个改动“distinct”, 本 PR 去掉的是采样器进程的 root 权限本身。