PR #44549 完整报告

vllm-project/vllm

[Security] Replace diskcache to eliminate pickle deserialization

合并时间：2026-07-15 11:29

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44549

执行摘要

- 一句话：使用 SQLite 替换 diskcache，消除 pickle 反序列化风险
- 推荐动作：该 PR 值得精读，尤其是学习如何用标准库替换存在安全风险的第三方依赖并保持 API 兼容。OutlinesDiskCache 类实现简洁直观，展示了 sqlite3 + outlines_core 原生序列化的替代方案，对于类似 pickle 安全问题的修复有参考价值。建议关注其中 __reduce__ 的使用方式。

功能与动机

Pickle deserialization allows arbitrary code execution, posing a security risk. CVE-2025-69872 identifies diskcache's use of pickle as a potential arbitrary code execution vector (CVSS 9.8 Critical). While vLLM is not vulnerable in practice (the disk cache is disabled by default and gated behind VLLM_V1_USE_OUTLINES_CACHE, and we document that the cache directory must not be shared with any untrusted users or content), the presence of diskcache as an installed dependency causes it to appear on security scans of vLLM environments. Removing the dependency eliminates the finding.

实现拆解

1. 实现 OutlinesDiskCache 类（vllm/v1/structured_output/utils.py）：核心类使用 Python 标准库 sqlite3，在初始化时创建 outlines_cache.db 文件并启用 WAL 模式。类内部通过类型标签 _TYPE_INDEX / _TYPE_STRING 区分存储对象：字符串直接 utf-8 编解码，outlines_core.Index 对象则使用 __reduce__ 获取其 Rust serde 二进制数据（调用 Index.from_binary() 反序列化）。
2. 替换 get_outlines_cache 工厂函数（同一文件）：当环境变量 VLLM_V1_USE_OUTLINES_CACHE 启用时，将原来的 diskcache.Cache 替换为 OutlinesDiskCache(cache_dir)，保持相同的 get/set/contains 调用约定和版本失效逻辑。
3. 移除 diskcache 依赖（requirements/common.txt 及 requirements/test/*.txt）：从顶层 common.txt 和三个测试平台依赖文件（cuda/rocm/xpu/cpu）中删除 diskcache==5.6.3 条目，消除安全扫描中的告警。
4. 新增测试套件（tests/v1/structured_output/test_outlines_cache.py）：覆盖 OutlinesDiskCache 的存 / 取 Index 对象与字符串、__contains__、get 的默认值行为、KeyError、clear、覆盖写入、跨实例持久化、目录创建、版本失效流程等 9 个场景，确保缓存正确性与兼容性。

关键文件：

- vllm/v1/structured_output/utils.py（模块 缓存；类别 source；类型 dependency-wiring；符号 OutlinesDiskCache, init, contains, getitem）：核心变更文件：新增 OutlinesDiskCache 类替换 diskcache.Cache，修改 get_outlines_cache 工厂函数，消除 pickle 安全风险。
- tests/v1/structured_output/test_outlines_cache.py（模块 缓存测试；类别 test；类型 test-coverage；符号 vocab, index, cache, TestOutlinesDiskCache）：新增完整测试套件，覆盖 OutlinesDiskCache 所有公有方法及边界场景，保障替换正确性。
- requirements/common.txt（模块 依赖配置；类别 docs；类型 documentation）：从顶层依赖中移除 diskcache，切断安全扫描告警源头。
- requirements/test/cuda.txt（模块 依赖配置；类别 docs；类型 documentation）：同步移除 CUDA 测试环境中的 diskcache 依赖。
- requirements/test/rocm.txt（模块 依赖配置；类别 docs；类型 documentation）：同步移除 ROCm 测试环境中的 diskcache 依赖。
- requirements/test/xpu.txt（模块 依赖配置；类别 docs；类型 documentation）：同步移除 XPU 测试环境中的 diskcache 依赖。
- requirements/test/cpu.txt（模块 依赖配置；类别 docs；类型 documentation）：同步移除 CPU 测试环境中的 diskcache 依赖。

关键符号：OutlinesDiskCache, init, contains, getitem, setitem, get, set, clear, get_outlines_cache

关键源码片段

vllm/v1/structured_output/utils.py

核心变更文件：新增 OutlinesDiskCache 类替换 diskcache.Cache，修改 get_outlines_cache 工厂函数，消除 pickle 安全风险。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project
import os
import sqlite3

from outlines_core import Index as oc_Index


class OutlinesDiskCache:
    """SQLite-backed cache for outlines_core.Index objects.

    Uses outlines_core's native binary serialization (via Rust serde)
    instead of pickle, eliminating arbitrary code execution risk on
    deserialization.
    """

    # 类型标签：区分字符串值和 Index 对象
```

```
_TYPE_INDEX = "I"
_TYPE_STRING = "S"

def __init__(self, path: str):
    os.makedirs(path, exist_ok=True)
    db_path = os.path.join(path, "outlines_cache.db")
    self._db = sqlite3.connect(db_path, check_same_thread=False)
    self._db.execute("PRAGMA journal_mode=WAL")
    self._db.execute(
        "CREATE TABLE IF NOT EXISTS cache "
        "(key TEXT PRIMARY KEY, type_tag TEXT NOT NULL, value BLOB NOT NULL)"
    )
    self._db.commit()

def __contains__(self, key: str) -> bool:
    row = self._db.execute("SELECT 1 FROM cache WHERE key=?", (key,)).fetchone()
    return row is not None

def __getitem__(self, key: str):
    row = self._db.execute(
        "SELECT type_tag, value FROM cache WHERE key=?", (key,)
    ).fetchone()
    if row is None:
        raise KeyError(key)
    type_tag, data = row
    if type_tag == self._TYPE_STRING:
        return data.decode("utf-8")
    # Index 对象使用 outlines_core 原生序列化恢复
    return oc_Index.from_binary(data)

def __setitem__(self, key: str, value):
    if isinstance(value, str):
        type_tag = self._TYPE_STRING
        data = value.encode("utf-8")
    else:
        type_tag = self._TYPE_INDEX
        # 通过 __reduce__ 获取二进制数据（Rust serde 序列化结果）
        data = value.__reduce__()[1][0]
    self._db.execute(
        "INSERT OR REPLACE INTO cache (key, type_tag, value) VALUES (?, ?, ?)",
        (key, type_tag, data),
    )
    self._db.commit()

def get(self, key: str, default=None):
    try:
        return self[key]
    except KeyError:
        return default
```

```
    def set(self, key: str, value):
        self[key] = value

    def clear(self):
        self._db.execute("DELETE FROM cache")
        self._db.commit()
```

tests/v1/structured_output/test_outlines_cache.py

新增完整测试套件，覆盖 OutlinesDiskCache 所有公有方法及边界场景，保障替换正确性。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project
import tempfile
import outlines_core as oc
import pytest
from vllm.v1.structured_output.utils import OutlinesDiskCache

pytestmark = pytest.mark.cpu_test


@pytest.fixture
def vocab():
    # 构造一个简单的词汇表，用于生成 Index
    return oc.Vocabulary(3, {b"a": [0], b"b": [1], b"ab": [2]})


@pytest.fixture
def index(vocab):
    # 构造一个 Index 对象（LLM 语法解析器）
    return oc.Index("ab", vocab)


@pytest.fixture
def cache(tmp_path):
    return OutlinesDiskCache(str(tmp_path))


class TestOutlinesDiskCache:
    def test_store_and_retrieve_index(self, cache, index):
        # 写入并读取 Index 对象，验证序列化完整性
        cache["test_key"] = index
        restored = cache["test_key"]
        assert restored.get_initial_state() == index.get_initial_state()
        assert restored.get_transitions() == index.get_transitions()
        assert restored.get_final_states() == index.get_final_states()

    def test_store_and_retrieve_string(self, cache):
        # 字符串存取（用于存储版本号等元数据）
        cache.set("__version__", "0.2.14")
```

```
        assert cache.get("__version__") == "0.2.14"

    def test_clear(self, cache, index):
        # 清空缓存
        cache["key"] = index
        cache.clear()
        assert "key" not in cache

    def test_persistence_across_instances(self, tmp_path, index):
        # 不同实例间共享同一个数据库文件，验证持久化
        cache1 = OutlinesDiskCache(str(tmp_path))
        cache1["key"] = index
        cache2 = OutlinesDiskCache(str(tmp_path))
        restored = cache2["key"]
        assert restored.get_transitions() == index.get_transitions()

    def test_version_invalidation_flow(self, cache, index):
        # 模拟版本失效流程：版本不一致则清空缓存
        cache.set("__version__", "0.2.13")
        cache["key"] = index
        cached_version = cache.get("__version__")
        new_version = "0.2.14"
        if cached_version != new_version:
            cache.clear()
        cache.set("__version__", new_version)
        assert cache.get("__version__") == "0.2.14"
        assert "key" not in cache
```

评论区精华

1. 类常量未使用问题：审核人 hmellor 指出 _TYPE_INDEX 和 _TYPE_STRING 定义为 b"I" / b"S" 但未在方法中使用（Magic strings 被直接使用）。作者 russellb 立即确认并修复，最终代码将这些常量改为字符串形式 "I" / "S" 并在 __getitem__ 和 __setitem__ 中正确引用，已解决。
2. 缓存目录权限安全建议：depthfirst-app[bot] 建议在 os.makedirs 添加 mode=0o700 以限制目录权限，防止本地其他用户读取缓存。该建议未被采纳，可能由于缓存默认禁用且文档已明确要求目录不可共享，属于低优先级增强。状态未解决。

- 未使用的 _TYPE_INDEX / _TYPE_STRING 常量 (design): 作者 russellb 确认并修复，最终将常量改为字符串并在方法中正确引用。
- 缓存目录权限安全建议 (security): 该建议未被采纳（未收到批复），可能因为缓存默认禁用且文档已要求目录不共享。

风险与影响

- 风险：

1. 序列化兼容性：使用 Index.__reduce__ 获取二进制数据，该接口依赖 outlines_core 的 Rust serde 稳定格式，若升级 outlines_core 后二进制格式不兼容，可能导致缓存失效（通过版本键 __version__ 自动清空，风险可控）。
2. 并发安全：SQLite 连接使用 check_same_thread=False 及 WAL 模式允许跨线程读写，但同一进程内多个 OutlinesDiskCache 实例指向同一数据库时仍需注意；多进程同时写入同一数据库文件（如共享缓存目录）可能存在竞态，但该场景极少见且官方文档已警告不应共享。
3. 资源消耗：SQLite 缓存是无界缓存，与原来 diskcache 的 eviction_policy="none" 行为一致，可能占用大量磁盘空间，维持原有的日志警告。
4. 测试覆盖：新增测试较为全面，但缺少并发写入的回归测试。 - 影响：影响范围：仅限于开启 VLLM_V1_USE_OUTLINES_CACHE 环境变量的用户（默认禁闭）。生产环境通常不启用，因此绝大多数用户无感知。依赖变化：diskcache 不再作为 vLLM 的依赖，CI/CD 环境需要移除已安装的包。安全收益：消除 CVE-2025-69872 的检出项，安全扫描可顺利通过。向后兼容：新的 OutlinesDiskCache 类完全兼容原有 diskcache 的 API（get/set/__contains__/clear），升级过程对调用方透明。 - 风险标记：Pickle 反序列化消除，SQLite 并发处理，依赖移除影响 CI，outlines_core 序列化兼容

关联脉络

- PR #48379 [Bugfix] Set kv_quant_mode on the generic MLA KV-cache spec: 同一仓库内结构化输出方向的关联 PR，但无直接依赖，仅作为上下文参考。
- PR #48428 fix: size FlashInfer prefill workspace to batch head footprint: 同样涉及 v1 后端的 bugfix，与 outlines 缓存无关，但同属 v1 模块。