

Винил снова в моде, или История дискового движка в Tarantool

Сергей Бронников, VK Tech



HighLoad++
2025

Давайте знакомиться



Сергей Бронников

Ведущий разработчик Backend

Разработчик в Tarantool



Нашёл последние 2 бага
в дисковом движке
Tarantool



Большой опыт участия
в крупных программных
проектах: Parallels Desktop
for Mac, Virtuozzo,
KasperskyOS



О чем доклад?

- ➔ О разработке Tarantool без прикрас.
- ➔ Об инженерных и продуктовых ошибках.
- ➔ Об исправлении этих ошибок.
- ➔ Об использовании фаззинга при разработке ПО.

План

- ➔ Появление дискового движка в Tarantool
- ➔ Первый релиз, первые пользователи и проблемы
- ➔ Боль пользователей
- ➔ Чемодан с оторванной ручкой
- ➔ Работа над ошибками
- ➔ Vinyl снова в моде
- ➔ Выводы

Назад в будущее



2014 год

Появление дискового движка в Tarantool

Tarantool в 2014-м

- ➔ In-memory база данных.
- ➔ База данных. Очередь.
- ➔ Кеш со вторичными индексами.

Один движок – memtx

100% данных в памяти



Транзакционность



Персистентное
хранение на диске



Высокая
производительность:
1 MOps



Прост в использовании:
не надо настраивать



Надёжный
в эксплуатации



Зачем ещё дисковый движок?

- ➔ Данные могут не поместиться в оперативную память
- ➔ Долгий старт Tarantool с memtx
- ➔ Снижение расходов: RAM всё ещё дороже SSD/HDD
- ➔ Новый рынок
- ➔ Технологический вызов

Мотивация

«Мы это делаем, потому что мы можем».

Костя Осипов

Мотивация

«Мы это делаем, потому что мы можем. Когда мы начинали, мы не понимали, во что мы ввязываемся».

Костя Осипов

Библиотека Sophia

 [pmwkaa](#) / [sophia](#) Public

Modern transactional key-value/row storage library.

 [View license](#)

 **1.9k** stars  **150** forks  Branches  Tags  Activity

Интеграция Sophia в Tarantool

commit f47b3bdeccaa19983457164573cad63220dba95b

author: Dmitry Simonenko <pmwkaa@gmail.com>

date: Tue Mar 18 21:57:32 2014 +0400

sophia-index: add sophia space engine.

Sophia в Tarantool 1.6.3

Subject: Tarantool 1.6.3 released

Date: 18 Jul 2014

The major new features of 1.6 are:

- Sophia engine (disk-based)

...

Sophia is RAM-Disk hybrid storage. It is designed to provide best possible on-disk performance without degradation in time. It has guaranteed $O(1)$ worst case complexity for read, write and range scan operations.

...

<https://groups.google.com/g/tarantool/c/6t280KVTJT8/m/x0-FrSMHuH0J>

2016 год

Пути Tarantool
и Sophia расходятся

Sophia в репозитории Tarantool

commit 2da93daa891e59d2b9f1a332b1b6c21ed31fad67

author: Dmitry Simonenko <pmwkaa@gmail.com>

date: Tue Apr 26 17:47:27 2016 +0300

sophia: convert to a built-in source

Phia в Tarantool

```
commit 1da93e203e3bf5a183b1ef64c5195decd5b23e35  
author: Konstantin Osipov <kostja@tarantool.org>  
date: Tue Apr 26 21:43:00 2016 +0300
```

```
sophia: rename sophia -> phia
```

Vinyl в Tarantool

commit b82af8749709d74e8d7e676722d1799682c92b9e

author: Konstantin Osipov <kostja@tarantool.org>

date: Fri Jun 24 17:00:04 2016 +0300

phia -> vinyl

.._two-storage-engines:

- The two storage engines: memtx and phia
- + The two storage engines: memtx and vinyl

Что не так с Sophia?

- ➔ «Сырой» код
- ➔ Дублирование кода с Tarantool (структуры данных)
- ➔ Дублирование функциональности (WAL, Tx)

Превращение Sophia в Vinyl

- ➔ Рефакторинг кода Sophia
- ➔ Полный редизайн

Vinyl в Tarantool 1.7.1

Subject: Tarantool 1.7.1 (alpha)

Date: 11 Jul 2016

The main feature of this release is a new storage engine, called "vinyl". Vinyl is a write optimized storage engine, allowing the amount of data stored exceed the amount of available RAM 10-100x times.

Vinyl is a continuation of Sophia engine from 1.6, and effectively a fork and a distant relative of Dmitry Simonenko's Sophia. Sophia is superseded and replaced by Vinyl.

<https://groups.google.com/g/tarantool/c/KGYj3VKJKb8>

Характеристики Vinyl

В основе — LSM (Log-Structured Merge) Tree



Фоновые операции сборки «мусора»



Многопоточная работа с диском



Оптимизация под запись и редкое чтение

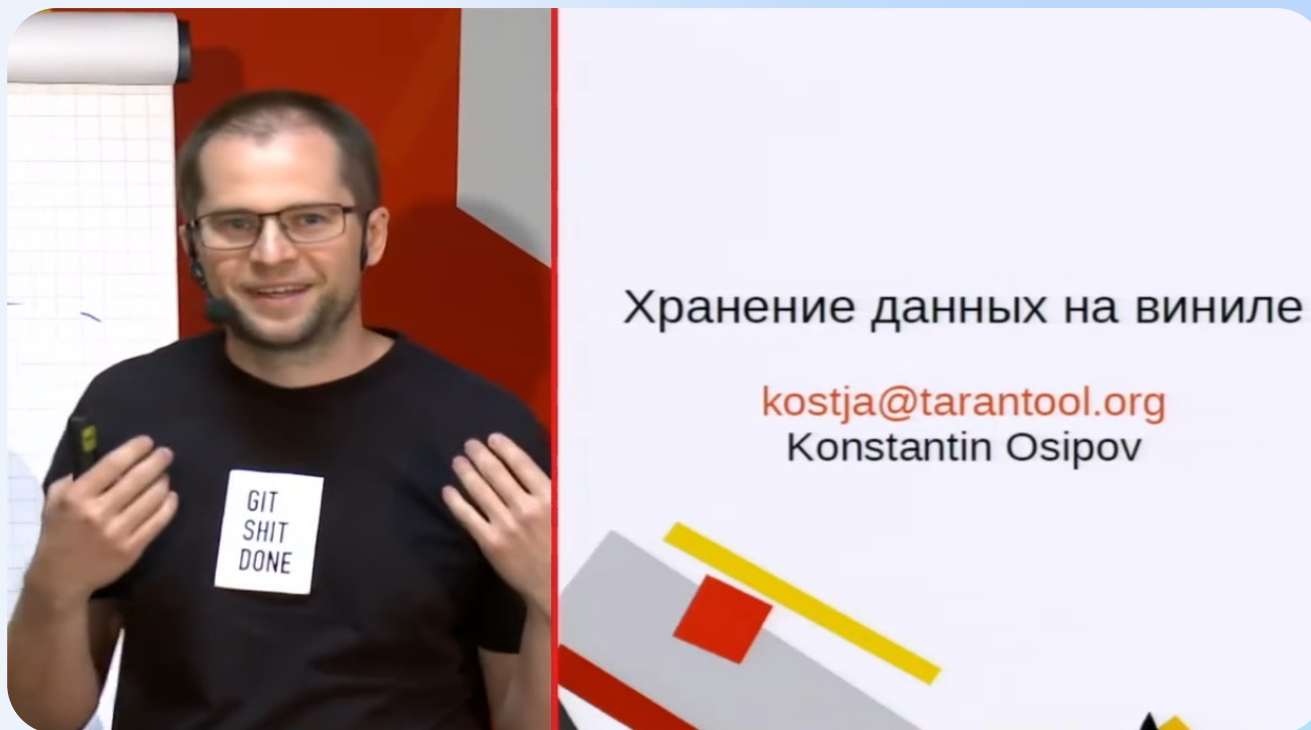


Tarantool один, а движка два.
Нужен паритет по фичам!

Надо догнать memtx!

- ➔ Многопоточность чтения и записи на диск
- ➔ Кеширование в памяти
- ➔ Отложенная операция DELETE
- ➔ Поддержка операции UPSERT
- ➔ Вторичные индексы
- ➔ «Выравниваем» Lua API для memtx и vinyl

Внутреннее устройство Vinyl



Хранение данных на Виниле, Константин Осипов, Highload 2016

Особенности разработки Vinyl

- ➔ Ограниченность ресурсов в команде
- ➔ Высокая скорость разработки
- ➔ Недостаточное нагрузочное и тестирование производительности

Особенности разработки Vinyl

- ➔ Ограниченность ресурсов в команде
- ➔ Высокая скорость разработки
- ➔ Недостаточное нагрузочное и тестирование производительности
- ➔ Недостаточное тестирование:
 - Нет CI
 - Нет санитайзеров
 - Нет фаззинга

2017 год

Первый релиз
и первые пользователи

Vinyl в Tarantool 1.7.4

Subject: Tarantool 1.7.4 (Release)

Date: 17 May 2017

Vinyl Engine, the flagship feature of 1.7.x, is now feature-complete.

All core features were finished and there are no known crashes, bad results or other showstopper bugs.

<https://groups.google.com/g/tarantool/c/3x88ATX9YbY>

Первые пользователи

- ➔ Пользователи из сообщества
- ➔ Mail.ru FRS:
 - отрепортили 10 крэшей
- ➔ Почта Mail.ru: отрицательный опыт использования

2019 год

Попытки стабилизации

Боль эксплуатации



Проблемы производительности

- Задержки из-за компрессии и фоновых процессов
- Проблема «залипания» диапазонов
- Производительность точечных запросов по вторичным ключам

Боль эксплуатации



Проблемы производительности



Проблемы настройки

- Сложность настройки и «чёрный ящик»
- Транзакционные ограничения

Боль эксплуатации

-  Проблемы производительности
-  Проблемы настройки
-  Низкая надёжность

Растеряли экспертизу по Vinyl

- ❗ Внешние крупные пользователи не появились
- ❗ Внутренних пользователей было мало
- ❗ Основные разработчики Vinyl постепенно покинули команду

Работаем над ошибками

- ➔ Багфиксинг силами остальных членов команды
- ➔ Разработка фокусных тестов для Vinyl
- ➔ По коркам тяжело понять причины, время инженеров тратится впустую

Работаем над ошибками

- ➔ Tarantool крошится, пользователи продолжают использовать и страдают
- ➔ Релиз Tarantool 2.4.2: исправлено 10 крэшей в Vinyl

Всё ли так плохо?

- ➔ Проект VK MyTracker
 - 2800 инстансов
 - 32 MOps
 - 106 ТБ данных в Vinyl



2023 год

Чемодан
с оторванной ручкой

Чемодан с оторванной ручкой

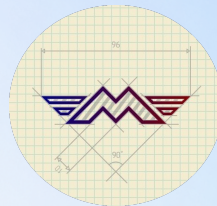
- ➔ Все известные проблемы исправлены
- ➔ Есть подозрение, что Vinyl всё ещё нестабильный
- ➔ Старые пользователи продолжают использовать
- ➔ Новым пользователям предлагать нельзя



*Две причины, по которым я перешёл на винил:
это дорого и неудобно.*

Статус поддержки Vinyl

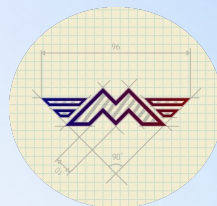
Не используйте
ВИНИЛ.



Mons Anderson

Статус поддержки Vinyl

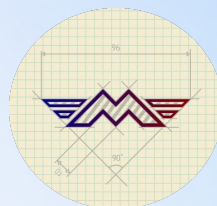
Оно не нелюбимое...
Оно недоношенное.



Mons Anderson

Статус поддержки Vinyl

Уходит в статус
beta/deprecated
и лишается саппорта.



Mons Anderson

2024 год

Работа над ошибками

Накопившиеся проблемы в Vinyl

- ❗ Сложности в настройке
- ❗ Отсутствие стабильности при эксплуатации
- ❗ Репутация перед пользователями оставляет желать лучшего

Что делать с нестабильностью?

- ➔ Переписывать дорого, нет таких планов
- ➔ Покрытие кода регрессионными тестами хорошее
- ➔ Написать тесты? Но какие?

Сделали статический анализ

- ✓ Не требует дополнительной работы
- ✓ Использовали Coverity, Svace, PVS Studio, Infer
- ✓ Были не критичные срабатывания

Устранили «тёмные» места в коде

- ✓ Запуск всех тестов под санитайзерами
- ✓ Исправляем все подавленные ранее проблемы, найденные санитайзерами
- ✓ Включение всех проверок в санитайзерах для всего кода
- ✓ Исправили «моргающие» тесты (и нашли баги!)
- ✓ Добавили поддержку ASAN в SMALL (и нашли баги!)

Типичные компоненты теста

- ➔ Tarantool Lua API
- ➔ Встроенный механизм сбоев (a-la BUGGIFY в FoundationDB)

Типичные компоненты теста

- ➔ Tarantool Lua API
- ➔ Встроенный механизм сбоев (a-la BUGGIFY в FoundationDB)
- ➔ Типичный тест:

```
s = box.schema.create_space(...)
```

```
s:create_index('pk')
```

```
s:insert({1})
```

```
box.error.injection.set('ERRINJ_WAL_DELAY', true)
```

```
...
```


Сделали фаззинг-тест

- ➔ Тест «черного ящика» для DML/DDDL-операций
- ➔ Рандомизация всего
 - параметров БД и таблиц
 - типов данных
 - последовательностей операций над данными
 - последовательностей сбоев
- ➔ Высокая конкурентность с помощью фиберов
- ➔ Санитайзеры, ассерты
- ➔ Всего 1.5 KLOC на Lua

Сделали фаззинг-тест (псевдокод)

```
local function main()  
  setup_box()  
  create_space()  
  
  for id = 1, num_workers do  
    fiber.new(worker_func, id, space, test_gen, deadline)  
  end  
  start_error_injections(space, deadline)  
end
```

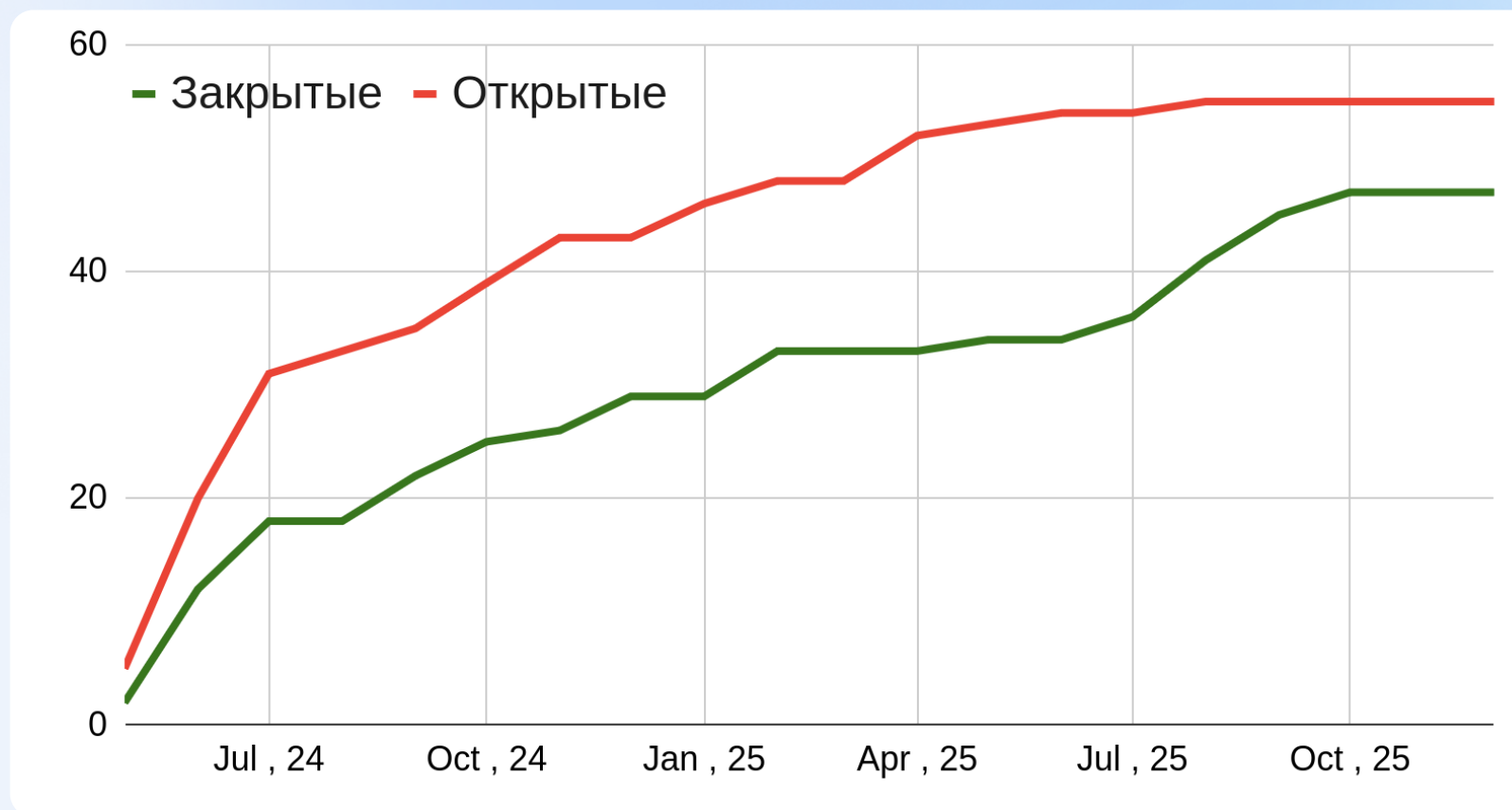

Тест нашёл 30 крэшей



Тест нашёл 30 крэшей



Количество открытых и закрытых проблем



Недостатки фаззинг-теста

- ❗ Вероятностный
- ❗ Непредсказуемость -> нельзя использовать в CI
- ❗ Нет «памяти»

Почему не использовать libFuzzer/AFL?

- ❗ Основной API реализован для Lua
- ❗ Публичный Tarantool C API слишком низкоуровневый

Приватные Си-функции

`datetime.parse (Lua) →`

`datetime_parse_from (Lua) →`

`builtin.tnt_datetime_strptime (LuaJIT FFI) →`



`tnt_datetime_strptime (C) →`

`datetime_strptime (C) →`

`tm_to_datetime (C)`



Много кода для реализации Lua API

`msgpack.decode (Lua) →`

`luamp_iterator_decode (Lua C API) →`

`luamp_iterator_decode (Lua C API) →`

`luamp_decode (Lua C API) →`

`msgpuck (C)`



Нужен фаззинг с поддержкой Lua

Фаззинг для Lua — luzer

Результаты: пример фаззинг-теста для Lua

```
local function TestOneInput(buf)
    local ok, res = pcall(msgpack.decode, buf)
    if ok == true then
        pcall(msgpack.encode, res)
    end
end
luzer.Fuzz(TestOneInput, nil, args)
```

45



Добавляем поддержку скриптового языка в AFL и LibFuzzer на примере Lua, Heisenbug 2022

Фаззинг-тест для vinyl: можно лучше!

- ➔ Тест «серого ящика» для DML/DDDL-операций
- ➔ В основе — фаззинг-движок для Lua — luzer
- ➔ Параметры теста максимизируют покрытие кода
- ➔ Накопленный корпус позволяет быстро проверять изменения
- ➔ Минус: низкая производительность
- ➔ Тест можно запускать в CI

Исчерпывающее тестирование НЕВОЗМОЖНО



Program testing can be used to show the presence of bugs, but never to show their absence!

Edsger W. Dijkstra

Нужно убедительное доказательство!



The only effective way to raise the confidence level of a program significantly is to give a convincing proof of its correctness.

Edsger W. Dijkstra

Верификация с помощью моделей

- ➔ Строительные блоки Tarantool
 - SMALL — семейство аллокаторов памяти
 - SALAD (Specialized ALgorithms And Data structures) — эффективные реализации структур данных
 - MATRAS (Memory Address TRanslation Allocator)
- ➔ Специфицируемы
- ➔ Нет зависимостей
- ➔ Компактный код

Верификация моделей CBMC

- ➔ Ограничивает модель для решения проблемы взрыва числа состояний
- ➔ Модель — код на C
- ➔ Отсутствие ложноположительных срабатываний
- ➔ Генерирует контрпример при нарушении спецификации
- ➔ Спецификация:
 - отсутствие UB
 - отсутствие OOM
 - пользовательские утверждения

CBMC в Tarantool

- ✓ Использовали CBMC для SMALL и SALAD
- ✓ Нашли несколько UB
- ✓ Получили больше уверенности в корректности!
(Дейкстра был бы доволен)

2025 год

Винил снова в моде

Работаем над репутацией

- ✓ Оперативная работа с фидбэком от эксплуатации
- ✓ Осторожно предлагаем новым пользователям
- ✓ Пишем руководство по эксплуатации Vinyl
- ✓ Новые внедрения: VK WorkSpace, Почта Mail.Ru
- ✓ Почта Mail.Ru: хотят использовать Vinyl, потому что слышаны о его стабильности

Выводы

- ✓ Отсутствие грамотного проектирования ведёт к проблемам
- ✓ Некачественное тестирование дорого обходится
- ✓ Думайте о тестировании во время проектирования
- ✓ Инжиниринг и труд все перетрут
- ✓ Держите bus-фактор под контролем

Винил снова в моде, или История дискового движка в Tarantool

Сергей Бронников

VK, VK Tech, Tarantool

- Телеграм: @ligurio
- Слайды: brnkv.ru/hl2025



Оцените доклад



HighLoad++
2025

Генеральный
партнер



Механизм сбоев в Tarantool (Си)

```
#define ERROR_INJECT_SLEEP(ID) \  
    ERROR_INJECT_WHILE(ID, usleep(1000))  
  
static int  
vy_task_compaction_execute(struct vy_task *task)  
{  
    ERROR_INJECT_SLEEP(ERRINJ_VY_COMPACTION_DELAY);  
    return vy_task_write_run(task, false);  
}
```


Механизм сбоев в Tarantool (Lua)

— Включить задержку.

```
box.error.injection.set("ERRINJ_VY_COMPACTION_DELAY", true)
```

— Выключить задержку.

```
box.error.injection.set("ERRINJ_VY_COMPACTION_DELAY",  
false)
```

Материалы

- ➔ «Хранение данных на Виниле» — К. Осипов
- ➔ «Работа с памятью в Tarantool: Small — Specialized Memory ALLocators» — А. Ляпунов
- ➔ «The Humble Programmer» — Edsger W. Dijkstra
- ➔ «The Fuzzing Book» на русском