Движок LMDB
- особенный чемпион

Леонид Юрьев,
Петер-Сервис РнД

PETER-SERVICE

http://www.billing.ru

Для крупных операторов связи:
BSS, Telco, BigData, HA & HL

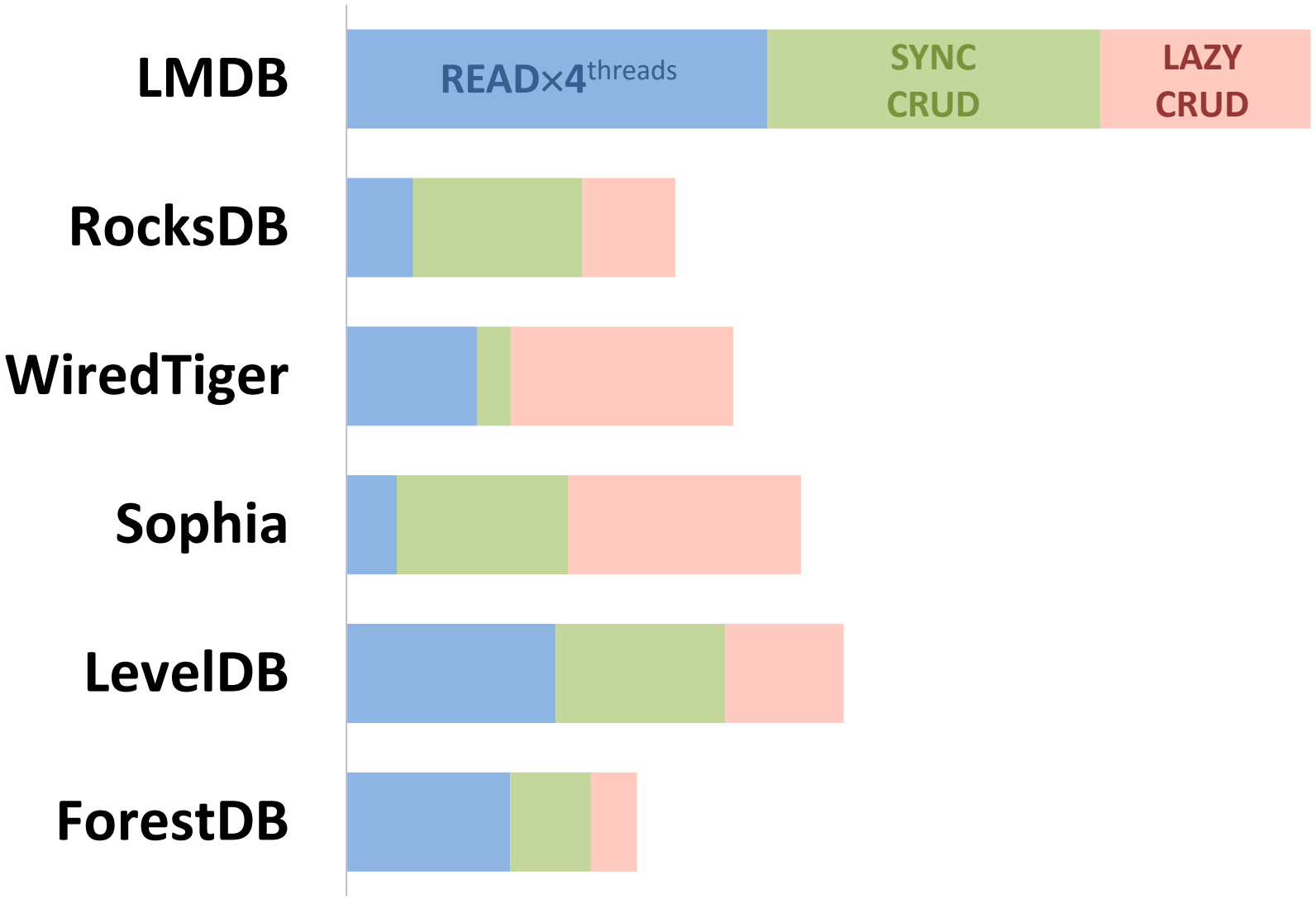
Более 20 лет разработки, внедрения и сопровождения

>125 миллионов абонентов обслуживается с применением нашего софта

Изменения после HL++

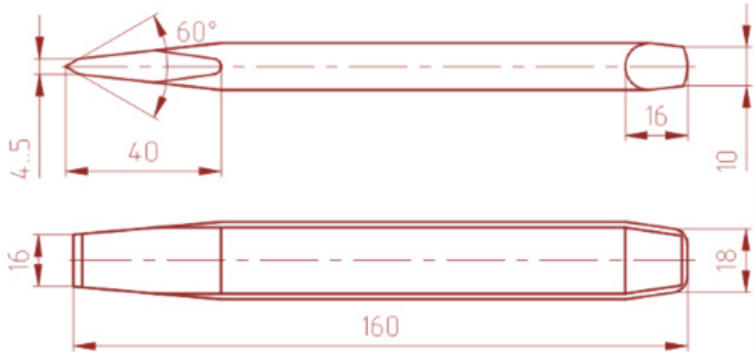
1. **sync** перед стартом бенчмарков
2. N для LAZY в **5 раз** больше
3. CRUD **без** фоновых GET и ITERATE
4. Чуть больше подписей на слайдах
5. Слайды «Стабильность» и этот

Наш герой



ГОСТ 7211-86

ЗУБИЛА СЛЕСАРНЫЕ



Как мы *докатились* до LMDB

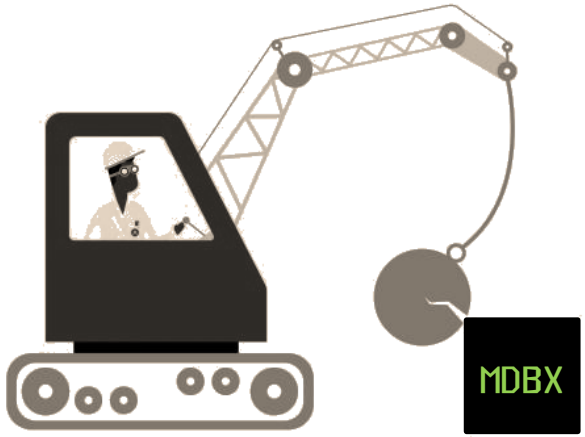
Немного о внутренностях

Плюсы и минусы

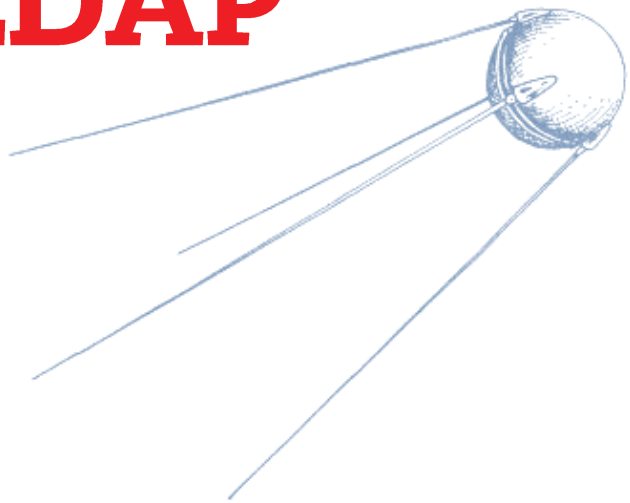
Наша MDBX

Telco → LMDB

1. LDAP *используется*, нужен сервер
2. Нечаянно выбрали OpenLDAP
3. **LMDB** внутри
4. Год сурка
5. Вот



Космический LDAP



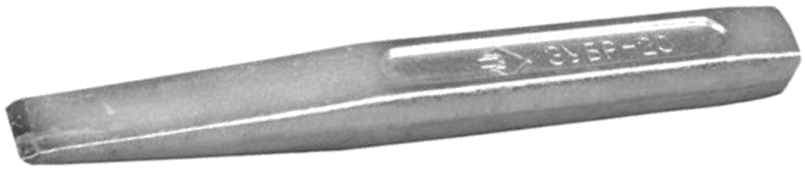
1. LDAP-кластер от **2+2**
2. До **100 миллионов** «*записей*»
3. Чтение 10K – 100K, **Запись 5K – 50K**

→ принципиально отличается от «обычных» сценариев использования...

Что такое LMDB ?

HighLoad++

LMDB (1)



Встраиваемый движок key-value (**!**)

Memory-mapped, просто файл и mmap()

B+tree, **без WAL**, COW
- ∅ восстановление
- ∅ компактификация

ACID поверх **MVCC**

10K SLOC, **64K** кода x86_64

LMDB (2)

N неблокируемых читателей
– чтение и поиск без ожидания, *всегда*

1 писатель
– изменения строго последовательны
– никаких deadlock и rollback error

Амортизационно $O(\log_2 N)$
– включая RAF и WAF
– *примерно* всегда

LMDB (3)

Приятные неожиданности:

– вложенные транзакции
– именованные таблицы
– сортировка для дубликатов
– *size_t* ключи

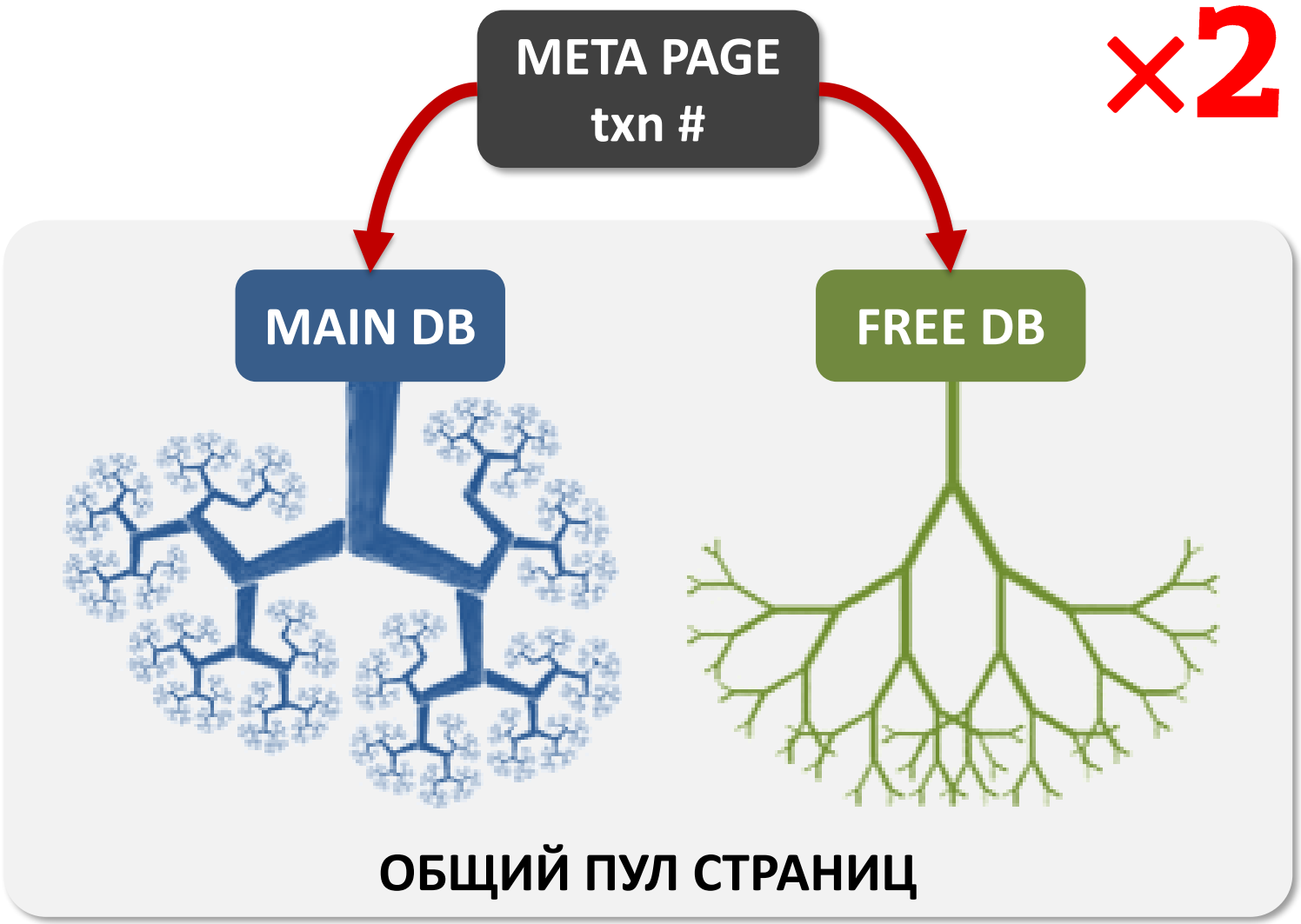
И ещё:

– горячее резервное копирование
– ∅ копирования и накладных расходов
– read-write mapping, *backed by kernel* **(!)**

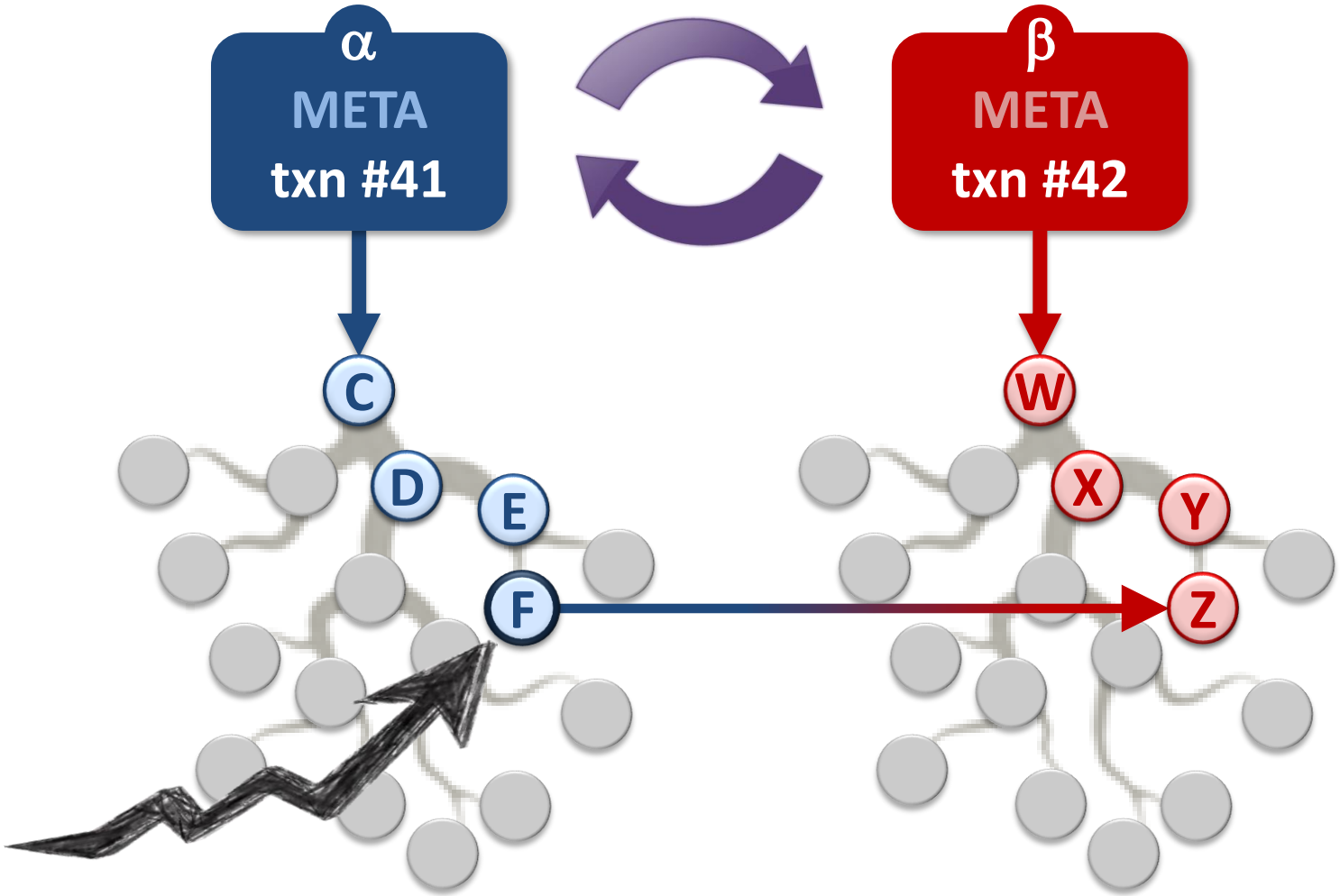
Препарируем...

HighLoad++

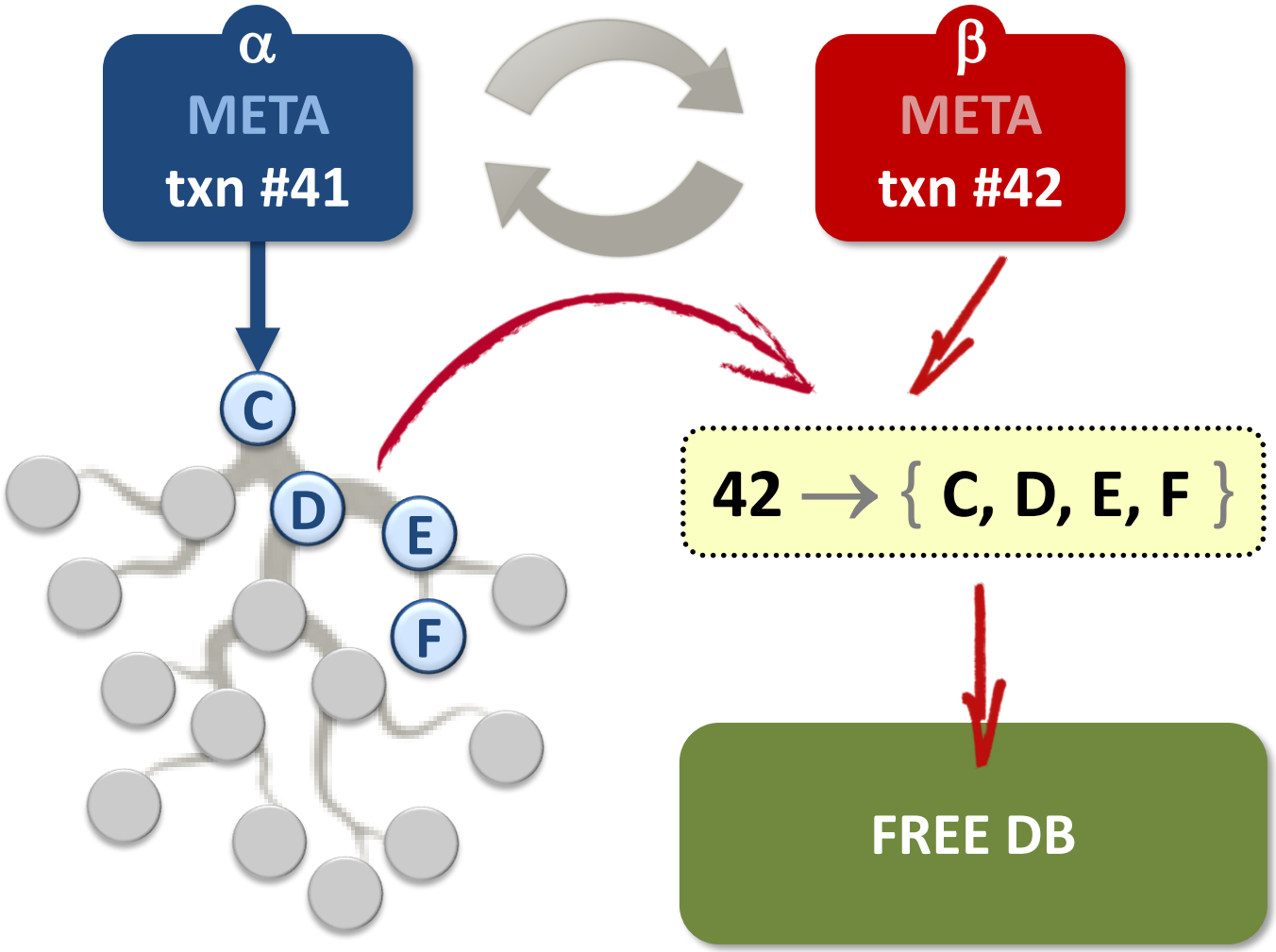
Три притопа, два прихлопа...



Copy-On-Write + FLIP-FLOP



Отслеживание мусора



Рабочий цикл

Проблемы и особенности

HighLoad++

Ошибки

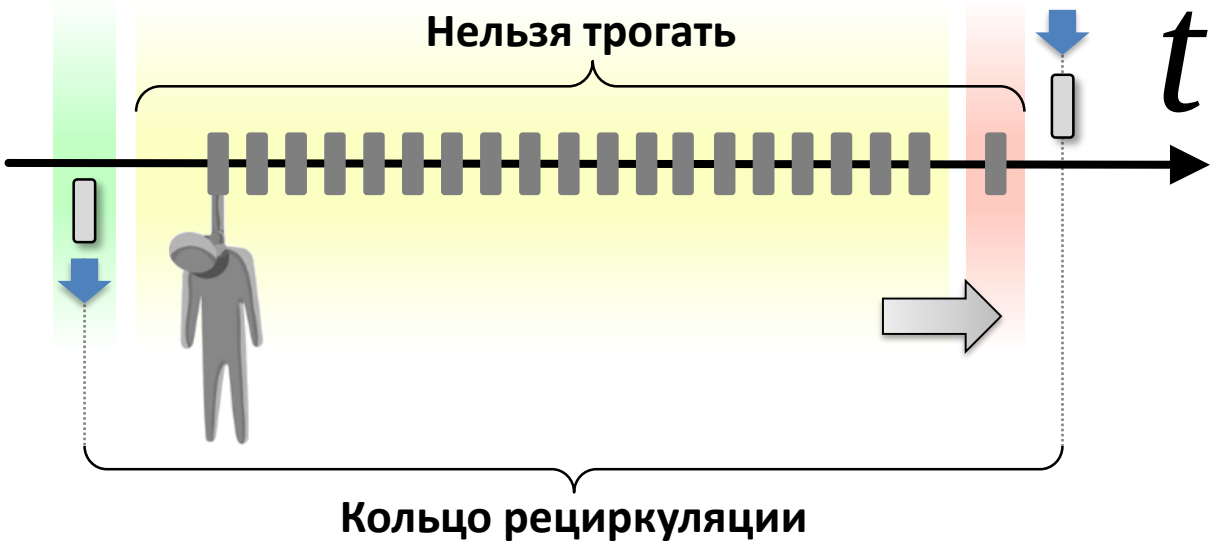
1. Курсоры, страницы, ...

2. **Volatile, барьеры памяти**
   ⇒ компилятор гуляет
   ⇒ **heisenbug**

3. **Read-Write Mapping**
   ⇒ порядок записи не гарантирован
   ⇒ **повреждение** базы

Проблемы

1. **Зависший читатель**
   ⇒ MDB_**MAP_FULL**
   ⇒ **вымывание I/O-кэша**



Особенности

1. **Нет WAL**
   ⇒ $\mathbf{WAF} = \boldsymbol{O}(\log_2 \boldsymbol{N})$ в страницах
   ⇒ восстановление не нужно

2. **Простой memory-mapping**
   ⇒ $\text{paging} = \boldsymbol{O}(\log_2 \boldsymbol{N})$
   ⇒ новое = старое
   ⇒ zero-copy

3. ⇒ **Прозрачность поведения***

Иногда падает,
надо бы поправить...

PETER-SERVICE

Горизонт
событий

1

2

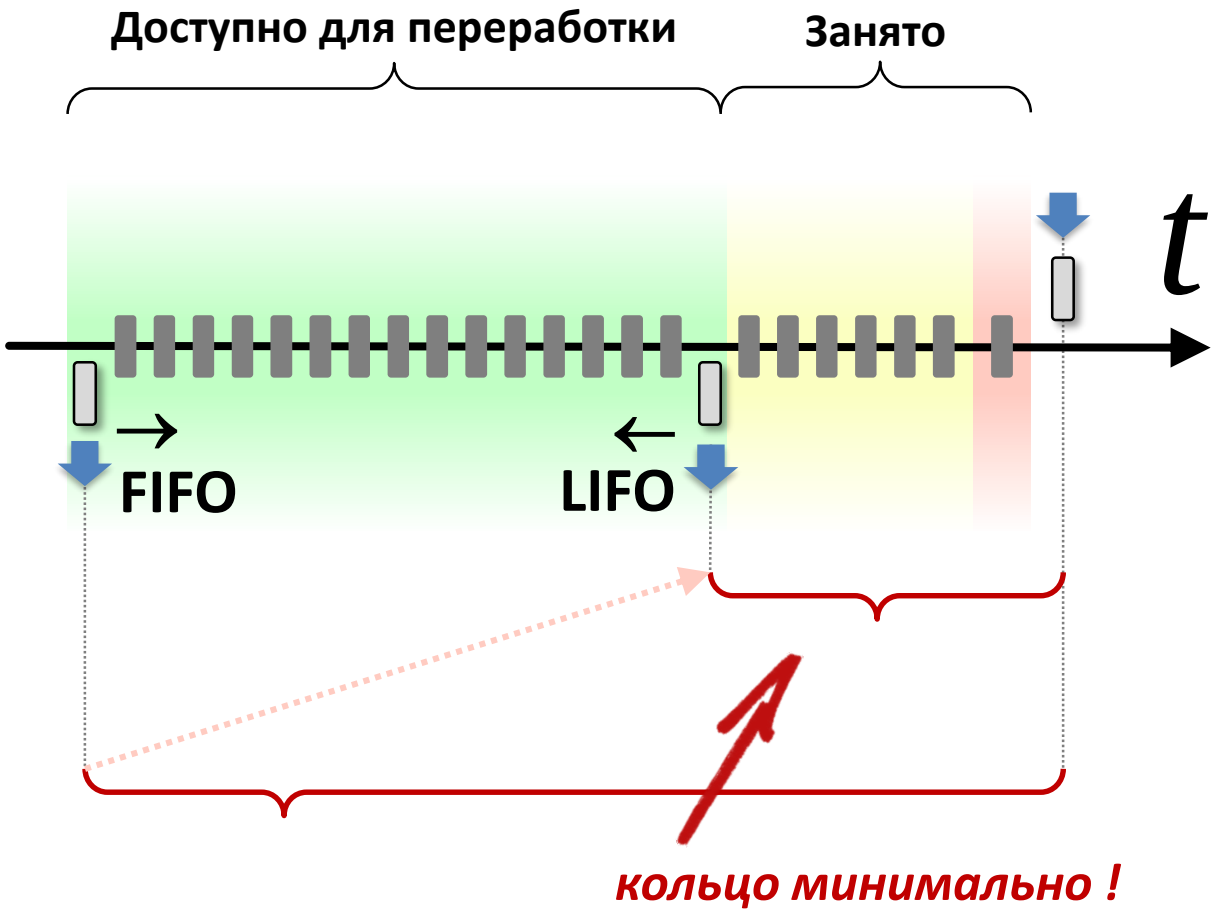
OpenLDAP

HighLoad++

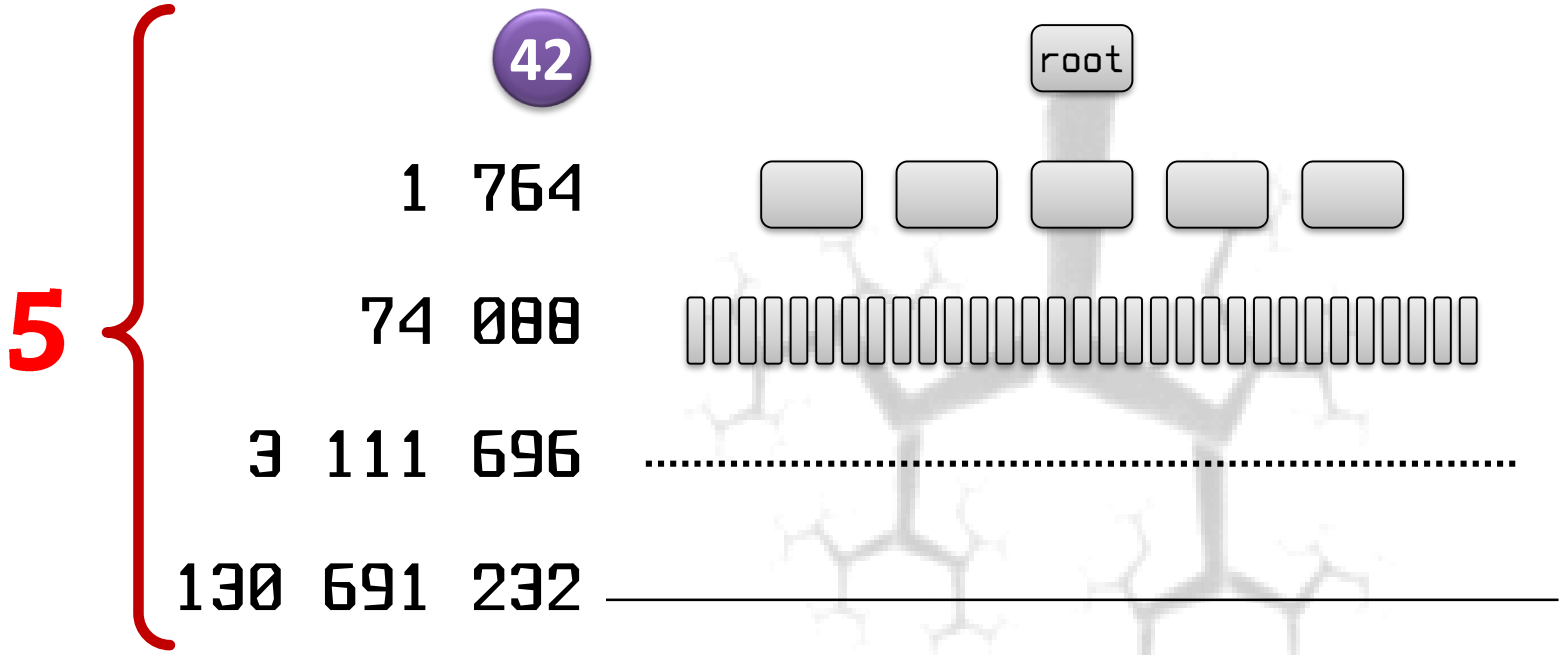
Доработки
→ MDBX

1. **АВТО-SYNC** и **LIFO** ⇒ лучше IOPS
2. **OOM-Handler** ⇒ бодрит читателей
3. **Путь записи**
   ⇒ надежность для Write-Mapping
   ⇒ steady/weak, есть TODO...
4. **MDBX_CHK**

LIFO в MDBX



Рост B+tree



LIFO и BBC battery-backed cache

N	$\Delta \times 2$	% 128M / 512M	IOPS FIFO / LIFO	результат 128M / 512M
42	0,34	—	2 / 0,00	—
1 764	14,12	—	3 / 0,00	—
74 088	592,93	21,59%	4 / 0,78	5,10 / 29,31
3 111 696	24,32	0,51%	5 / 1,78	2,81 / 4,48
130 691 232	1021,41	0,01%	6 / 2,78	**2,16 / 2,84**

Снижение нагрузки на диск в разы

LIFO и BBC battery-backed cache

Устраняет архитектурную проблему

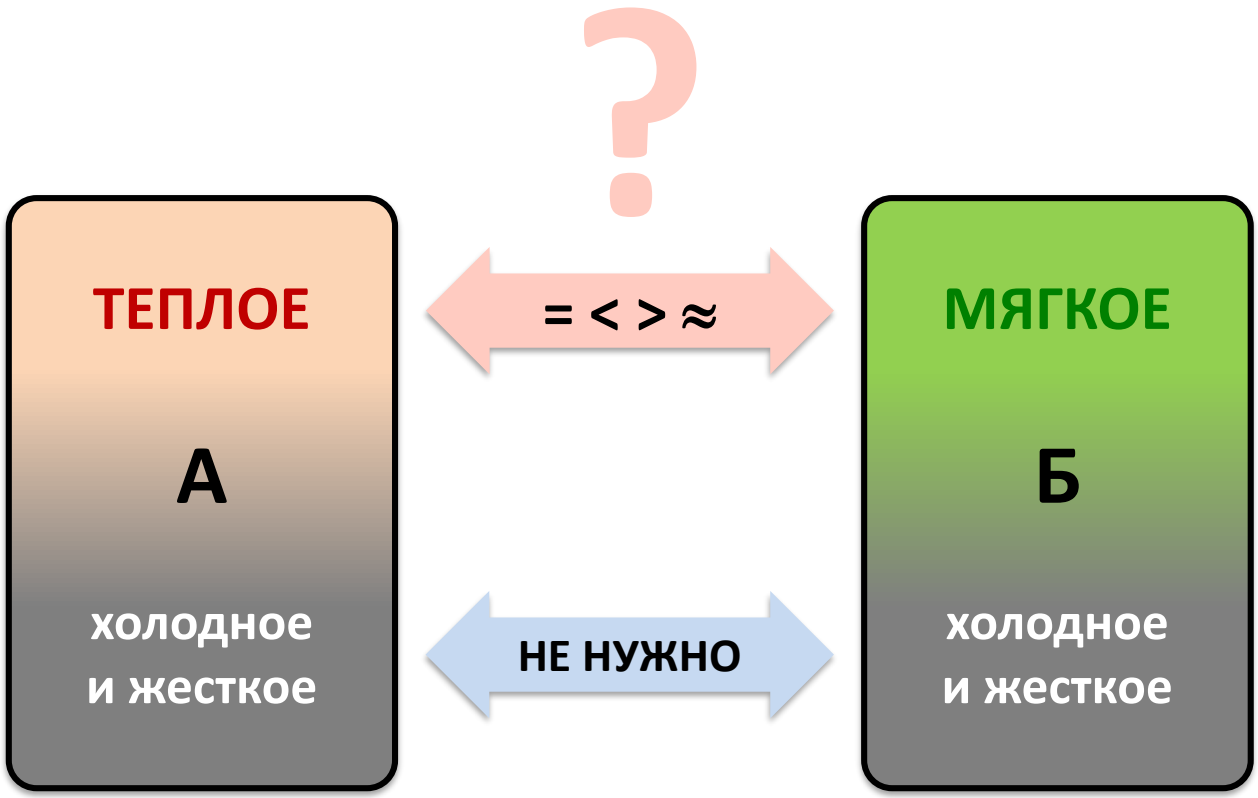
В разы:
– снижает нагрузку на диски
– повышает производительность

Паралимпиец → Чемпион:
– зависшие читатели не так страшны
– только в **MDBX**

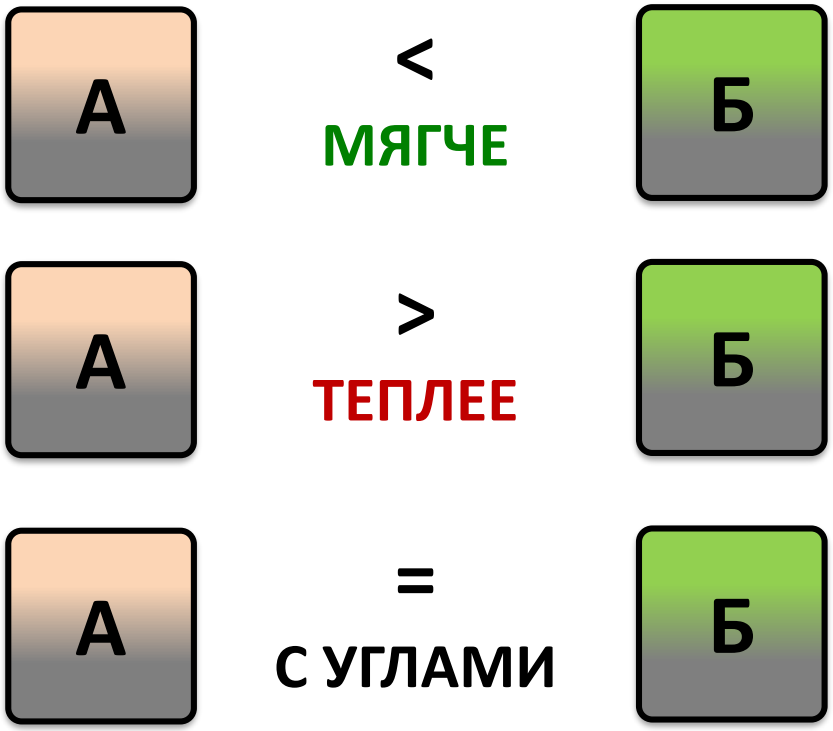
Попробуем сравнить...

HighLoad++

Теплое и Мягкое



Теплое и Мягкое



Метрики

1. **КАЧЕСТВО ОБСЛУЖИВАНИЯ**
   - задержка
   - вариация задержки
2. **СТОИМОСТЬ**
   - CPU
   - IOPS, WAF, RAF
   - место на диске

Режимы

SYNC ±WAL

– непосредственно на диск

– ничего не теряется

± redo

LAZY

– асинхронная фиксация

– хвост может отпасть

NOSYNC

– можно потерять все изменения

Бенчмарки

SET – добавляем/обновляем

GET – читаем

CRUD – классическая четверка

BATCH – много четверок пачкой

ITERATE – упорядоченно по ключам

DELETE – чистим

Многопоточность

	1	NR	NR+W	NR+NW	R+NW	NW
SET	●					
GET	●	●	●○	●○	●○	
CRUD	●		○●	○●	○●	●
BATCH	●		○●	○●	○●	●
ITERATE	●	●	●○	●○	●○	
DELETE	●					

SYNC

LAZY

NOSYNC

НЕ ИССЛЕДОВАНИЕ, А СРАВНЕНИЕ

30 мин

МИНИМУМ СЦЕНАРИЕВ
– иначе не рассказать

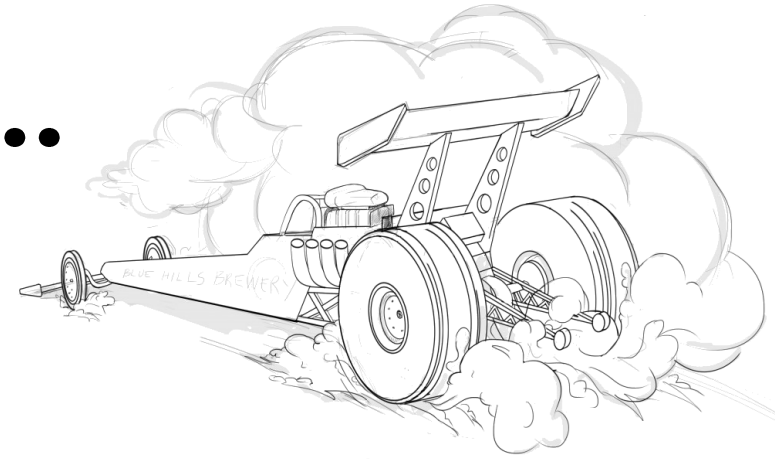
24 ЧАСА
– иначе никогда не успеть

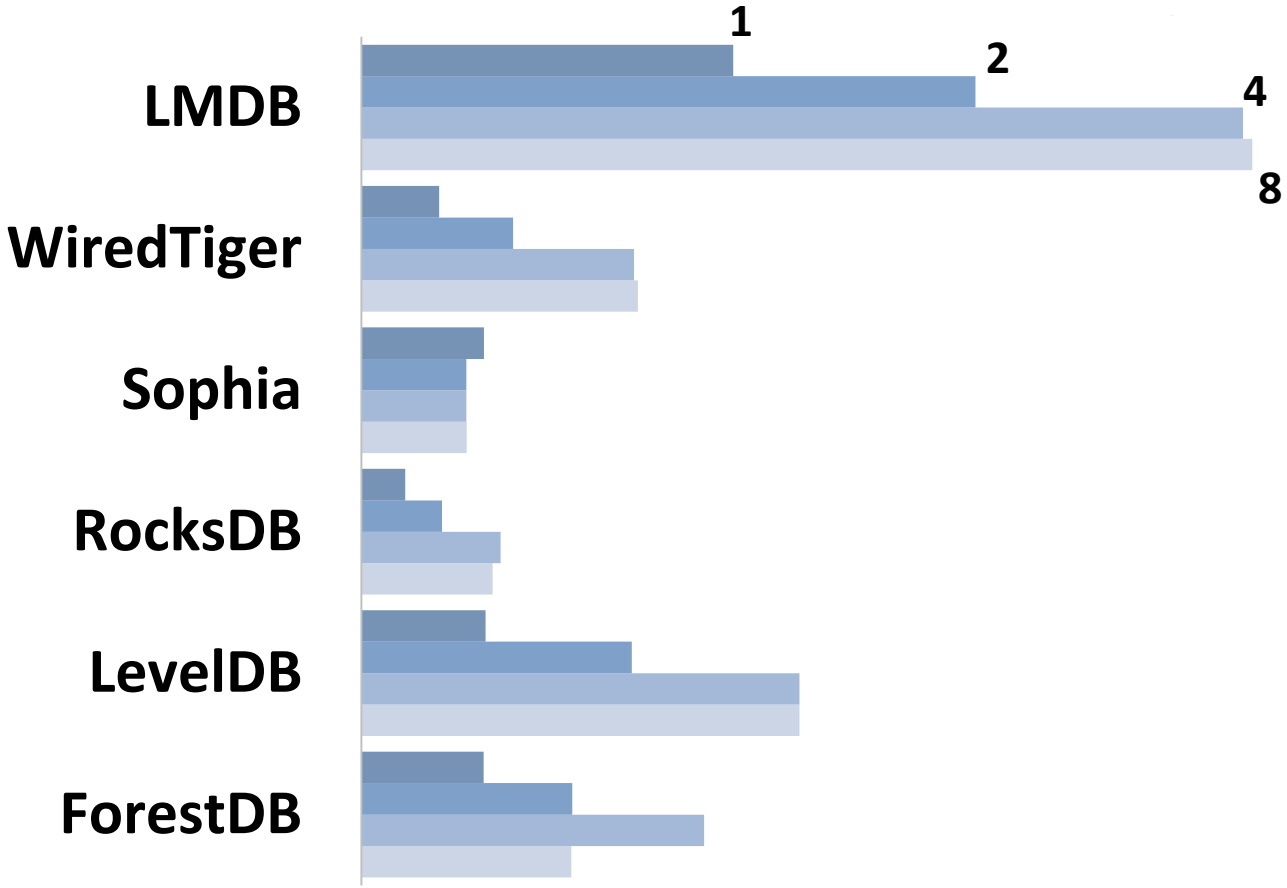
DATA < RAM
– иначе долго, либо не репрезентативно

Результаты!

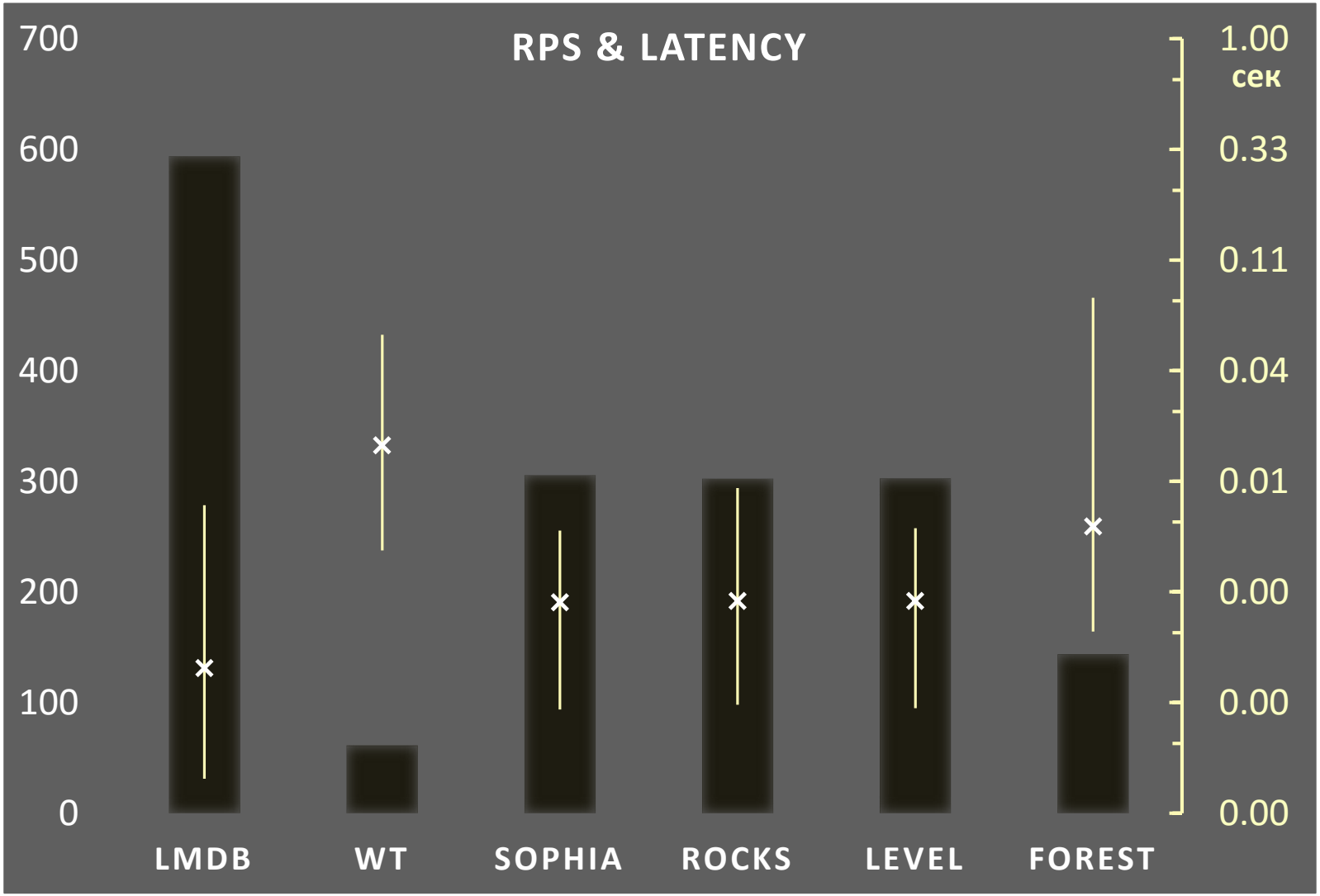
HighLoad++

Разгон по CPU...

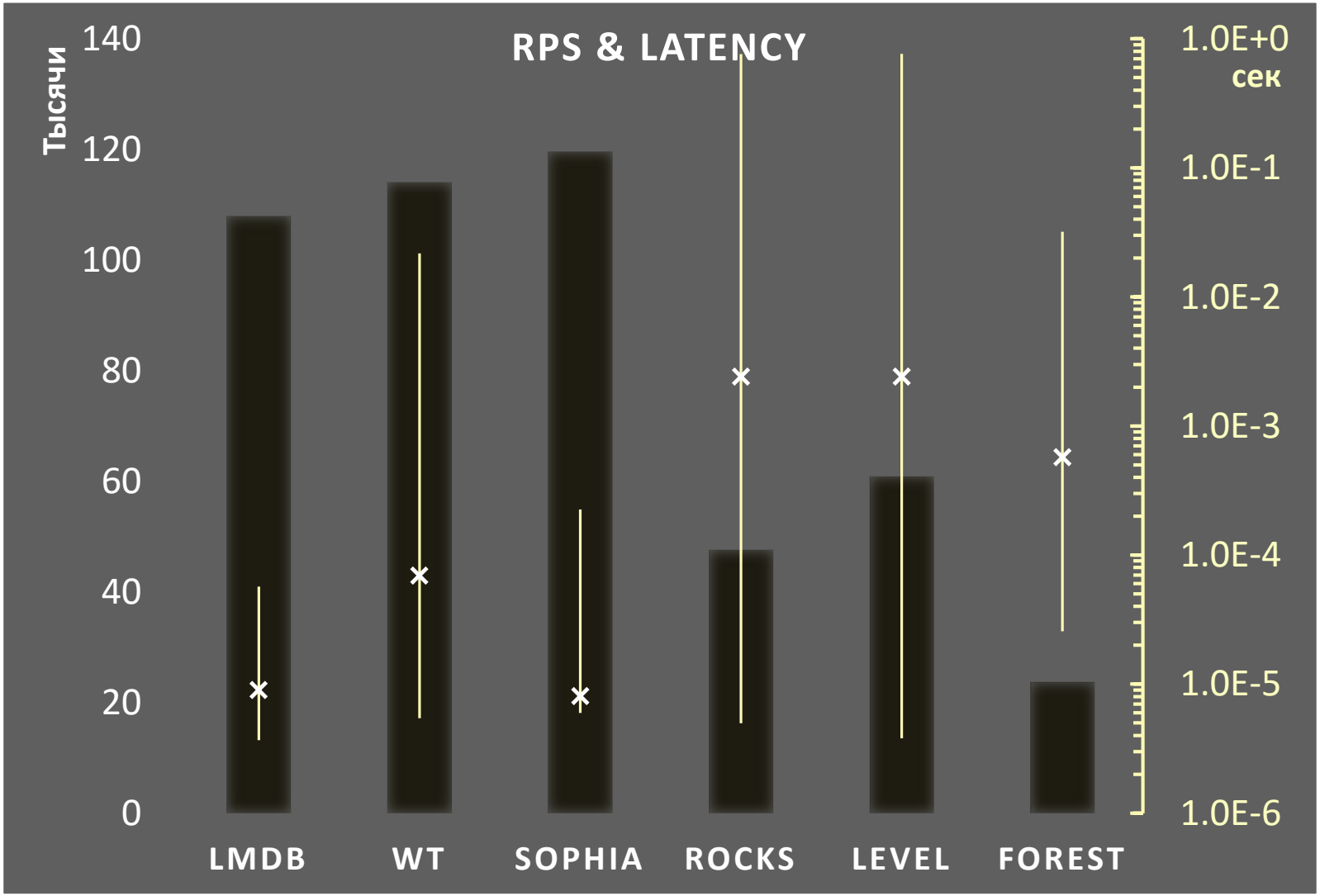




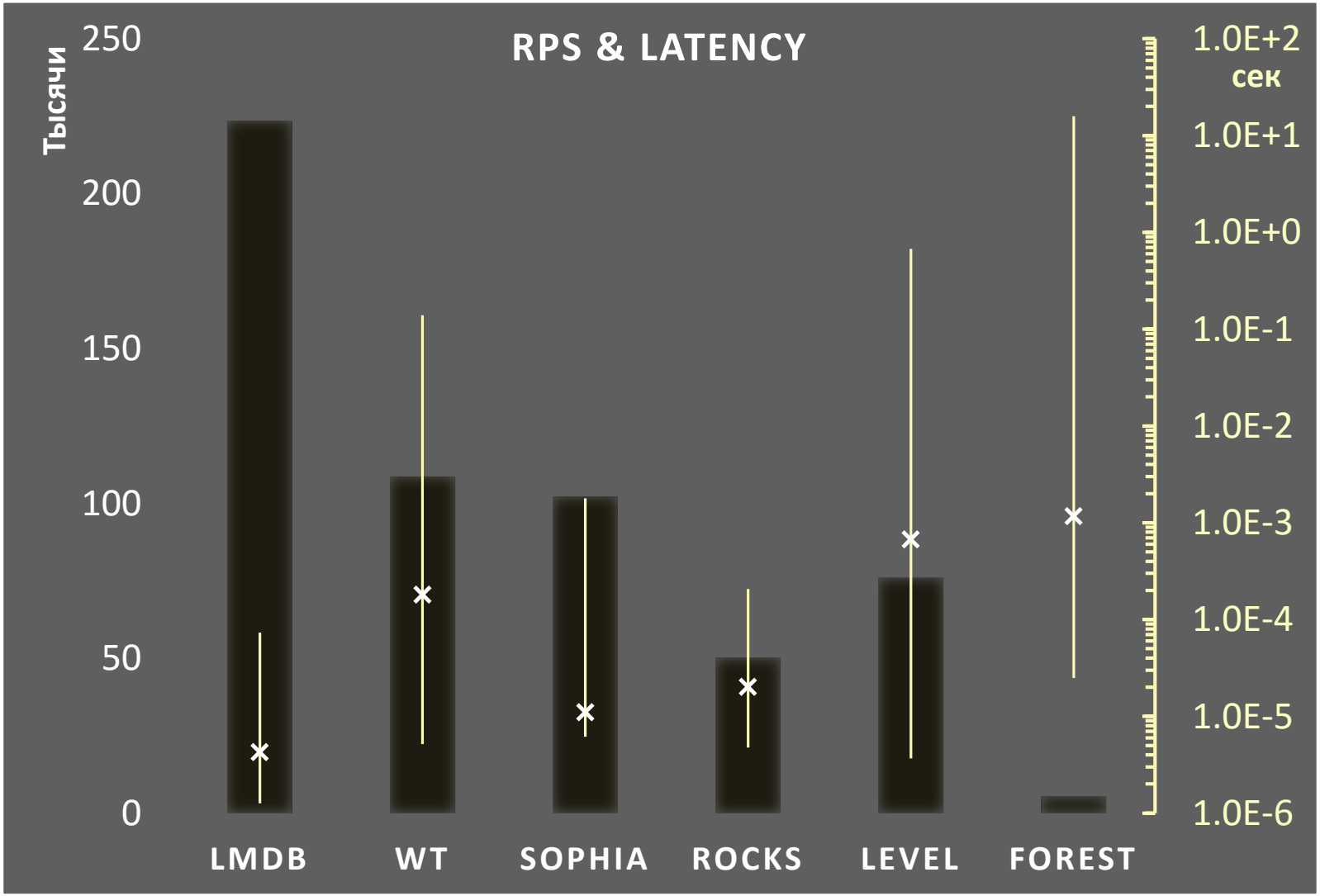
SYNC CRUD×10⁴



LAZY CRUD×5·10^5

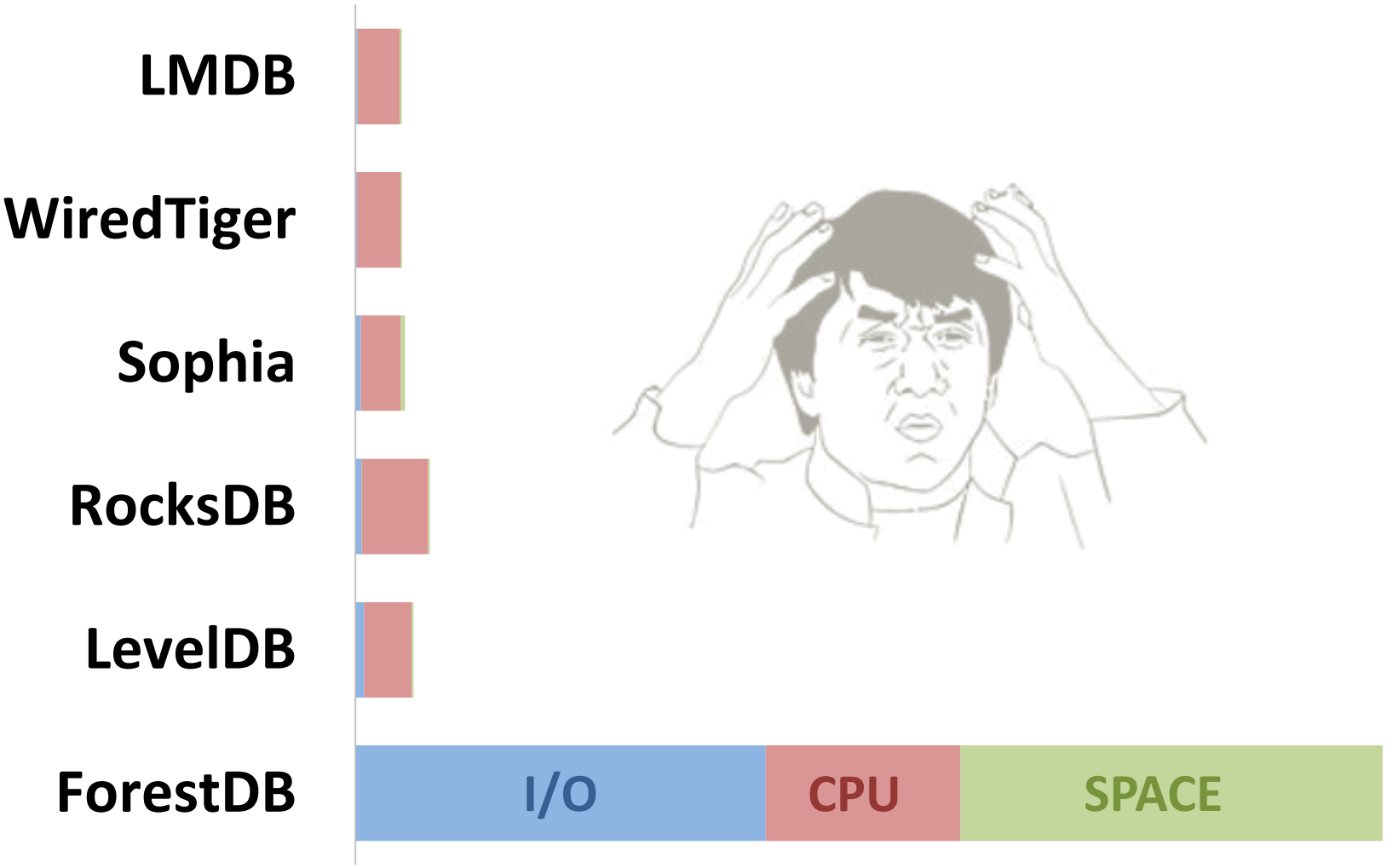


NOSYNC CRUD×10^6

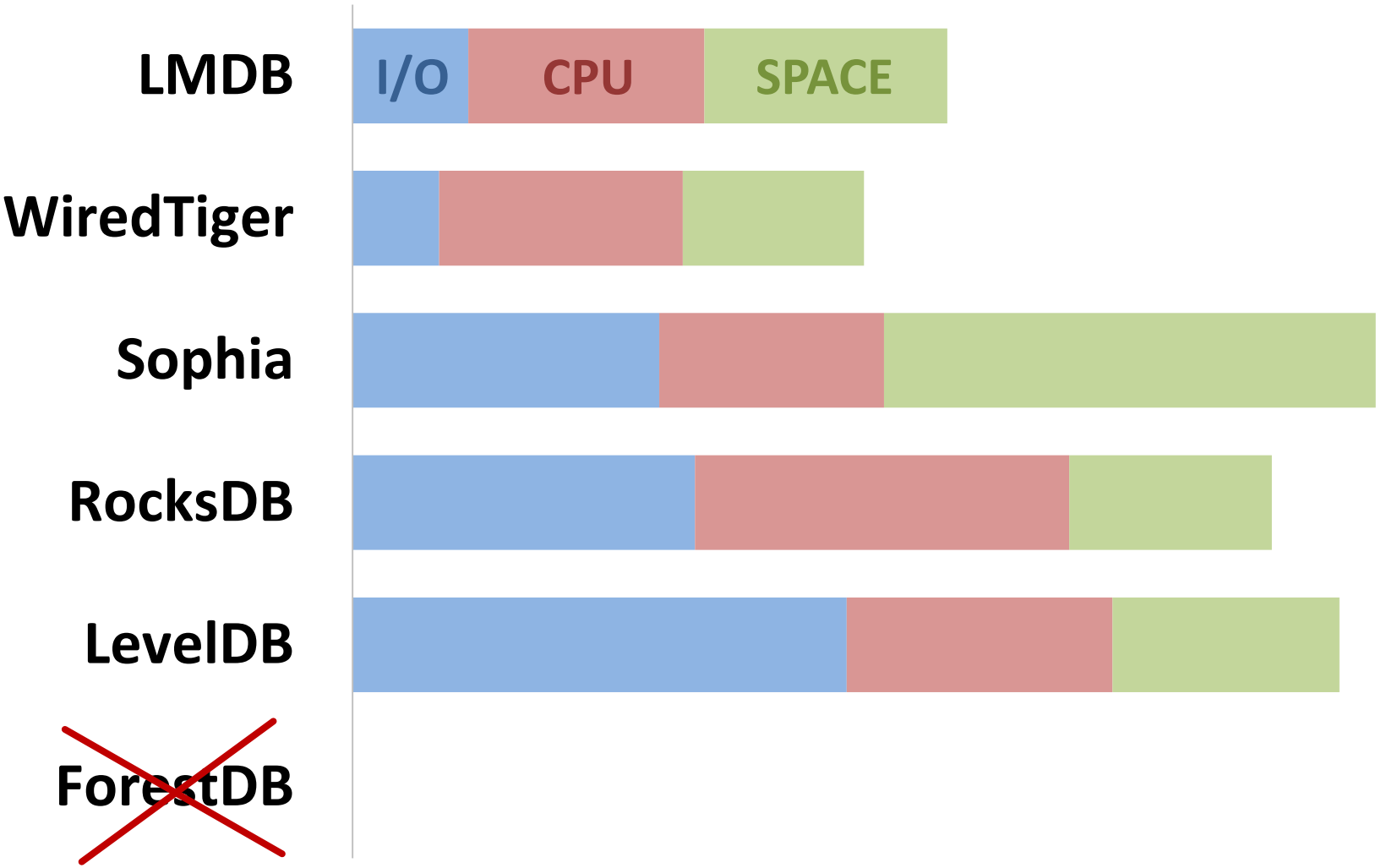


HighLoad++

Стоимость LAZY CRUD×5·10^5



Стоимость LAZY CRUD×5·10^5



Стабильность

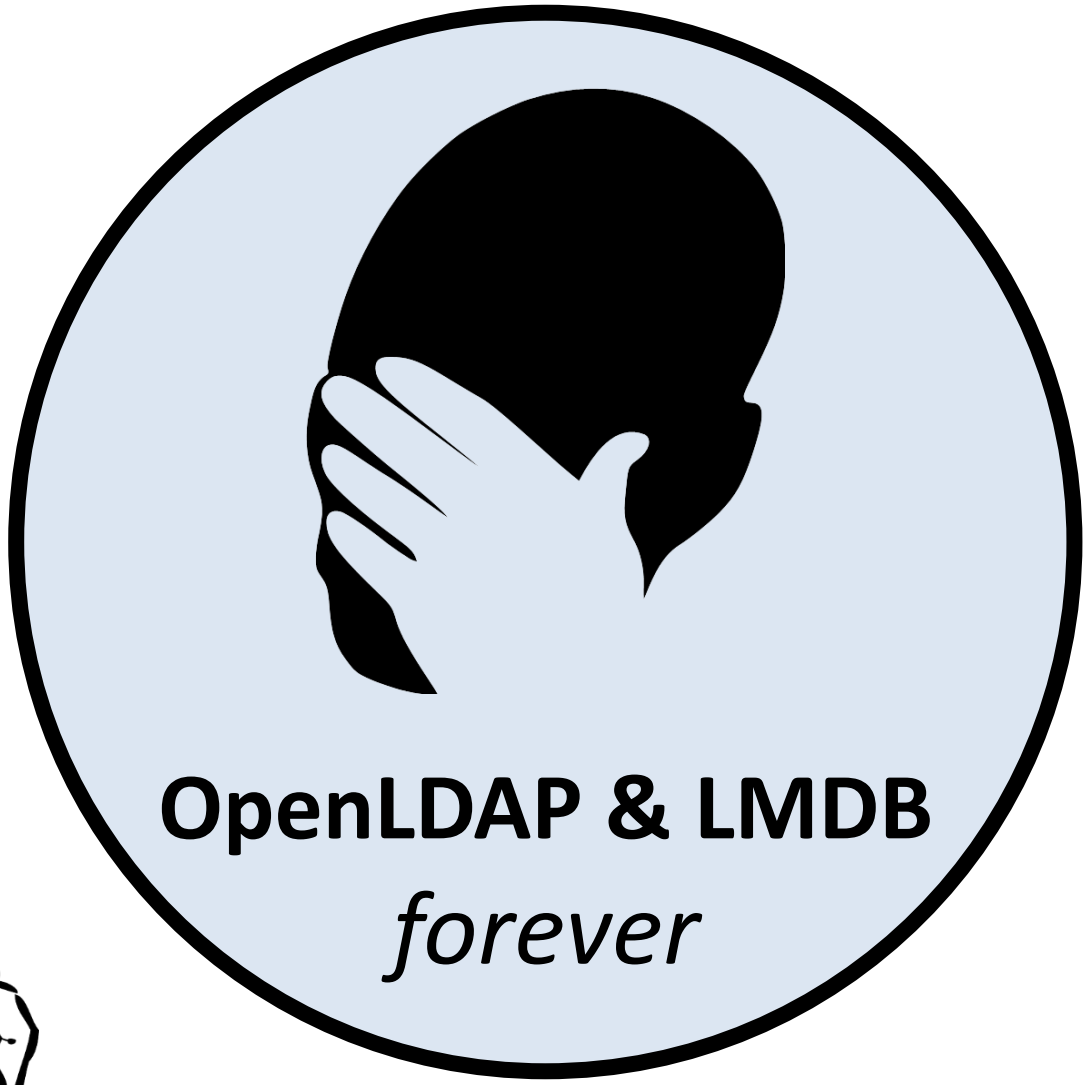
LevelDB
– падения на CRUD с вероятностью 50%

ForestDB
– падения на CRUD с вероятностью **90%**

Не умеем готовить?
– научите нас…
https://github.com/ReOpen/ioarena

Что дальше?

HighLoad++



Что дальше с MDBX ?

1. **Фоновая фиксация**
   – автоматически weak → steady

2. **Порядок/Оптимизация**
   – лишние проверки
   – узкие места

3. **Release**
   – документирование
   – пакетирование

Что потом с MDBX ?

50/50

MDBX → 389DS
– Red Hat Directory Server (RHDS)

Merkle Tree → MDBX
– полная потеря совместимости
+ 1Hippeus

OpenDJ → Петер-Сервис

Да или Нет ?

HighLoad++

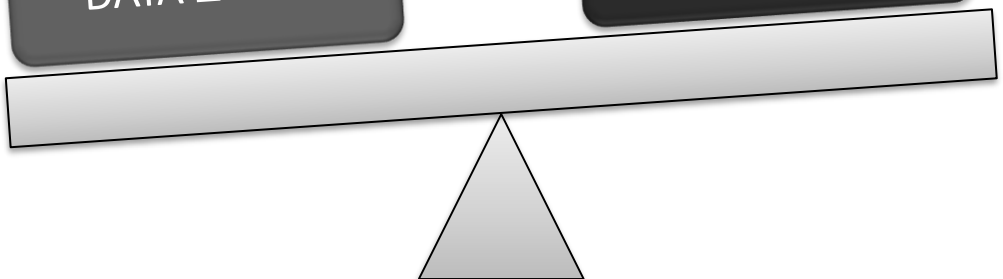
LMDB или НЕТ ?

Подходит

- много дубликатов
- много читателей
- ∅ восстановления
- DATA ≤ RAM

НЕ подходит

- **много писателей**
- показан **WAL**
- DATA > RAM



Спасибо !

LMDB
http://symas.com/mdb/

MDBX
https://github.com/ReOpen/libmdbx

IOARENA[2]
https://github.com/ReOpen/ioarena