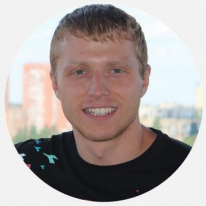


Денис Противенский
Percona

PGDAY'17 RUSSIA
КОНФЕРЕНЦИЯ ПО БАЗАМ ДАННЫХ

Хранилище на LSM-дереве в качестве движка базы данных

Опыт MongoRocks

PERCONA
Server for MongoDB

Содержание

Что есть MongoRocks

Содержание

Что есть MongoRocks

Как устроена RocksDB

Содержание

Что есть MongoRocks

Как устроена RocksDB

Контракты MongoDB для уровня хранения

Содержание

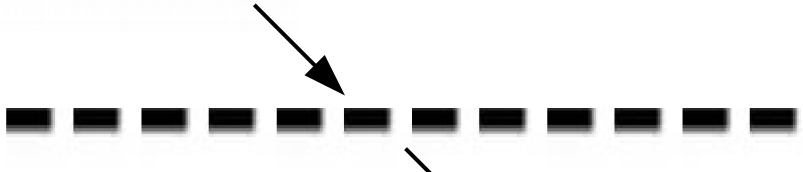
Что есть MongoRocks

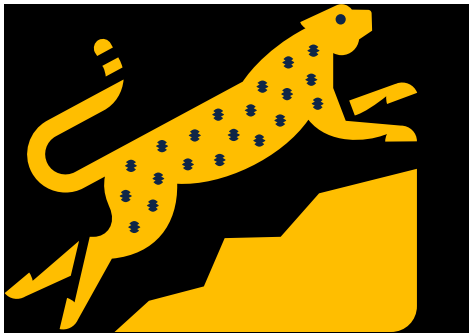
Как устроена RocksDB

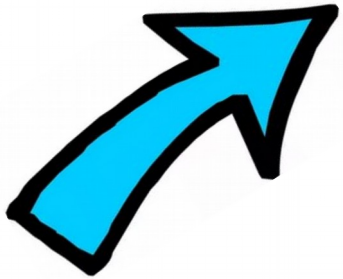
Контракты MongoDB для уровня хранения

О самой проблемной операции

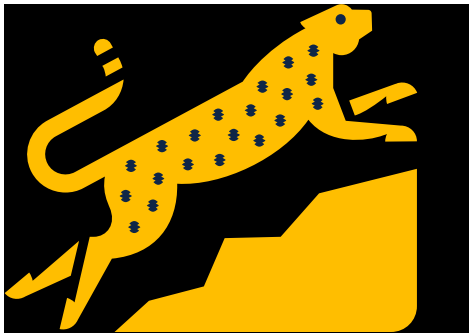
mongoDB

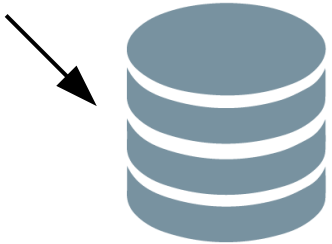




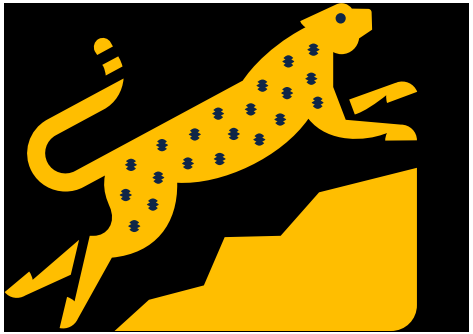


MongoRocks





RocksDB



RocksDB для пользователя

Хранилище ключ-значение:

RocksDB для пользователя

Хранилище ключ-значение:

- $\mathrm{Get(k)} \rightarrow \mathrm{v}$
- $\mathrm{Put(k, v)}$
- $\mathrm{Delete(k)}$

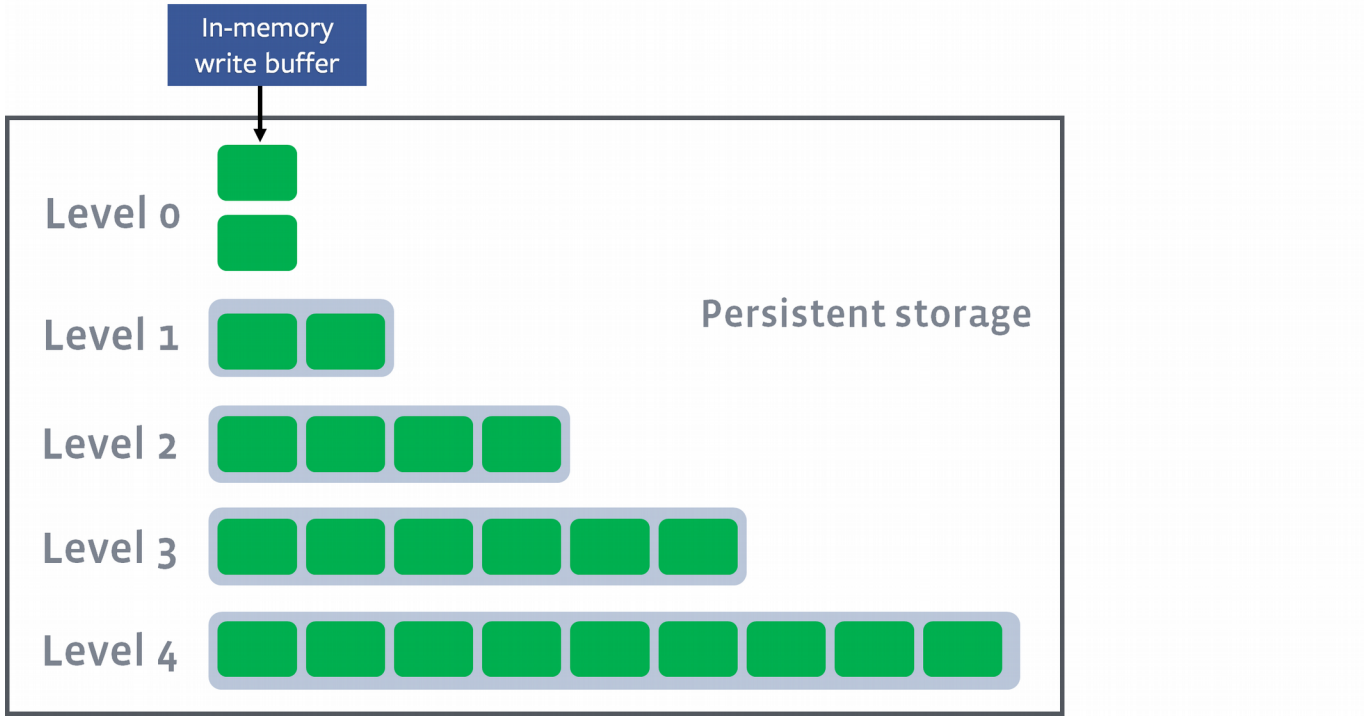
RocksDB для пользователя

Хранилище ключ-значение:

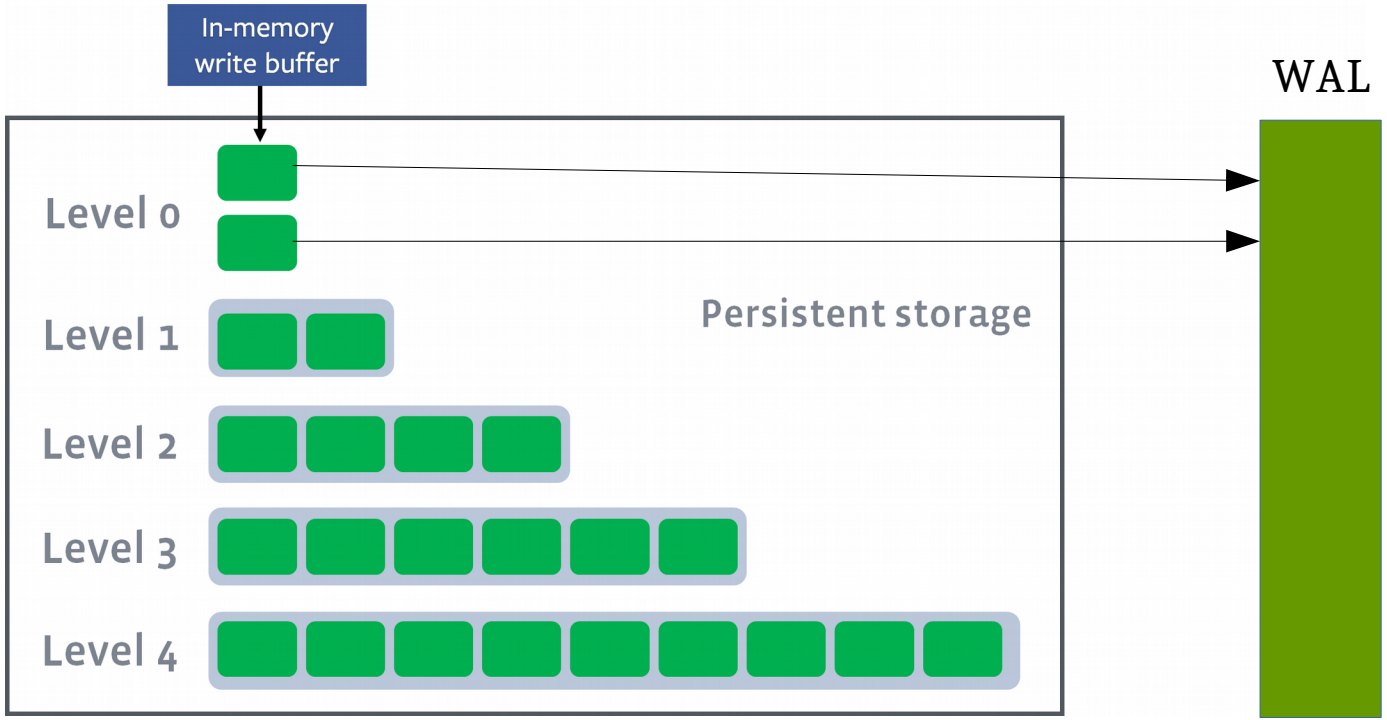
- Get(k) → v
- Put(k, v)
- Delete(k)

- Merge ...

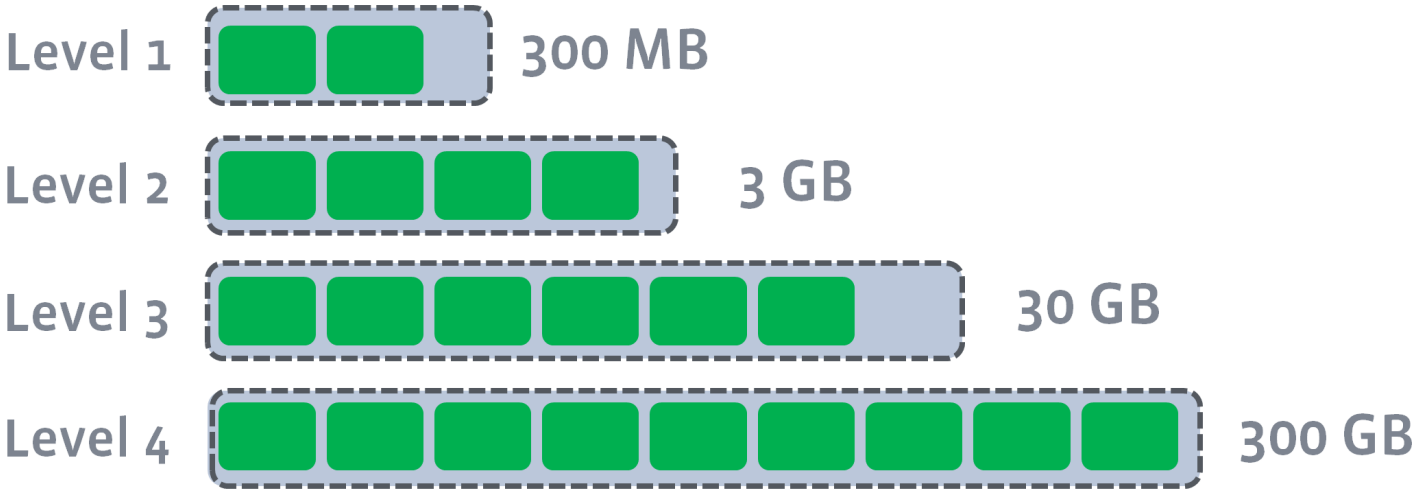
Организация уровней



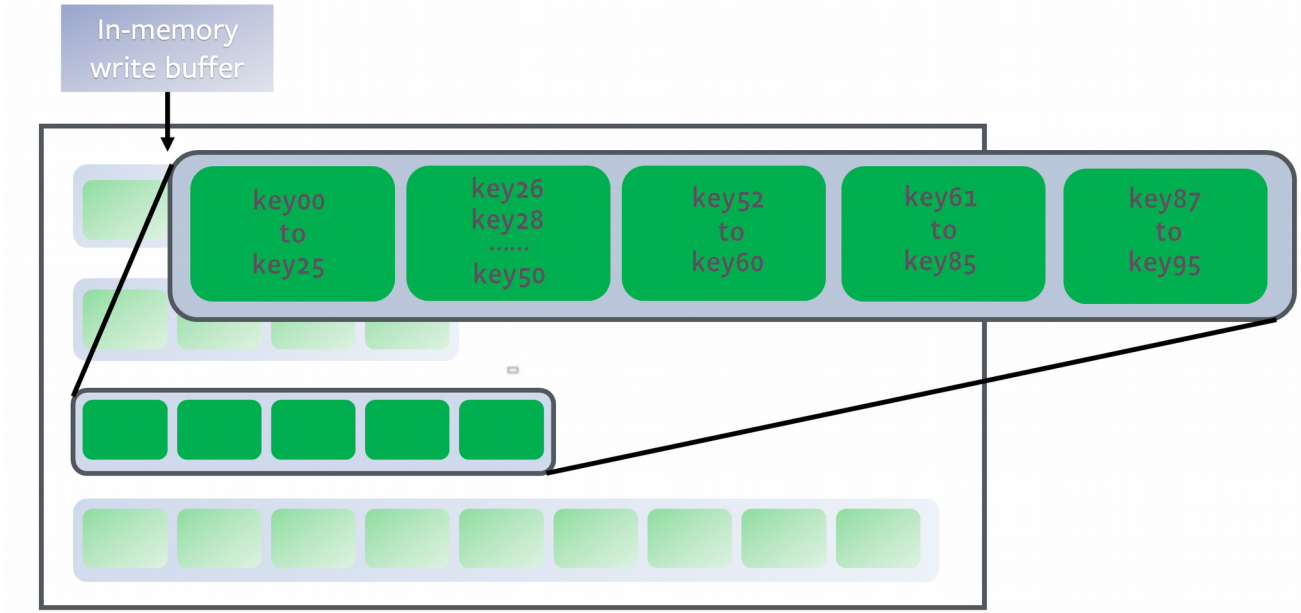
Журнал операций



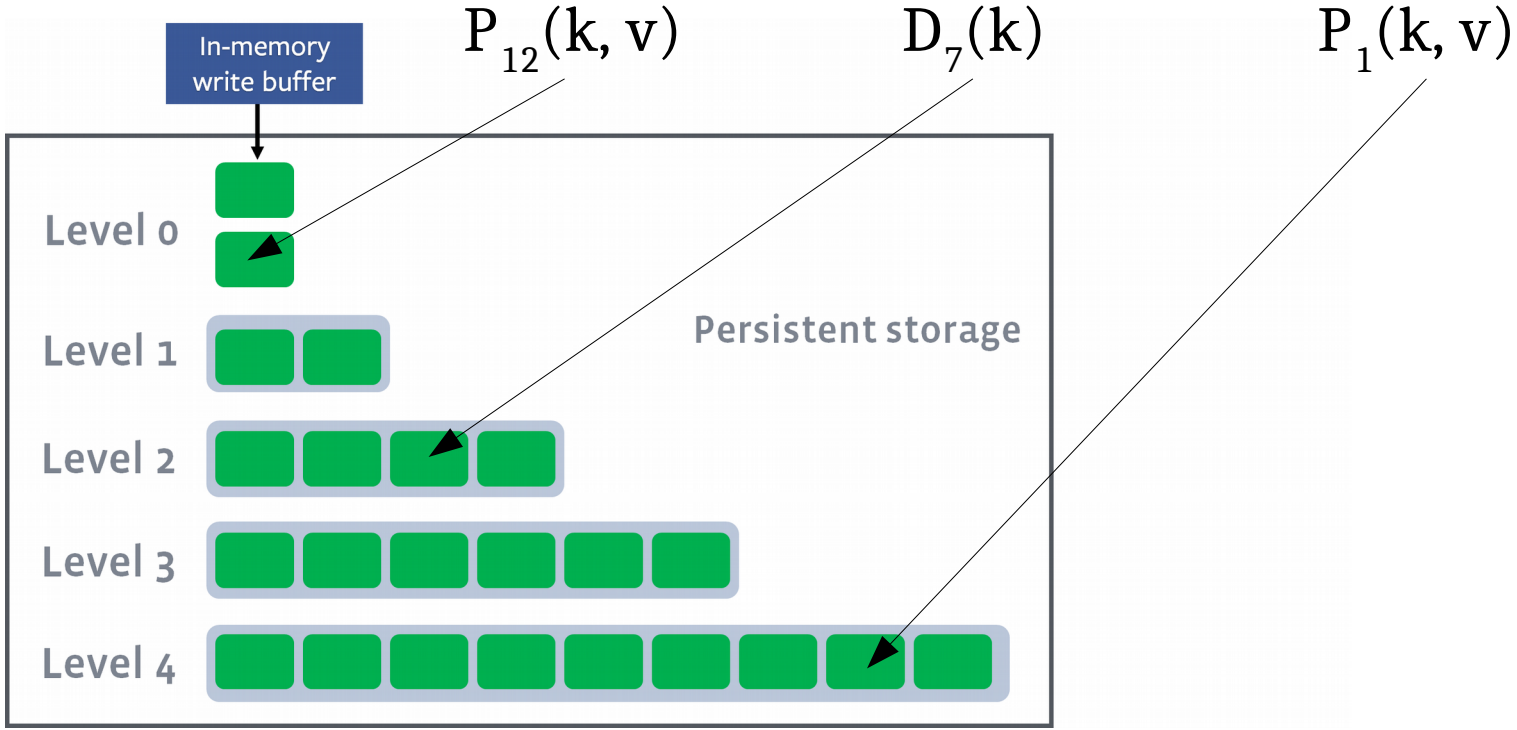
Следующий уровень кратно больше предыдущего



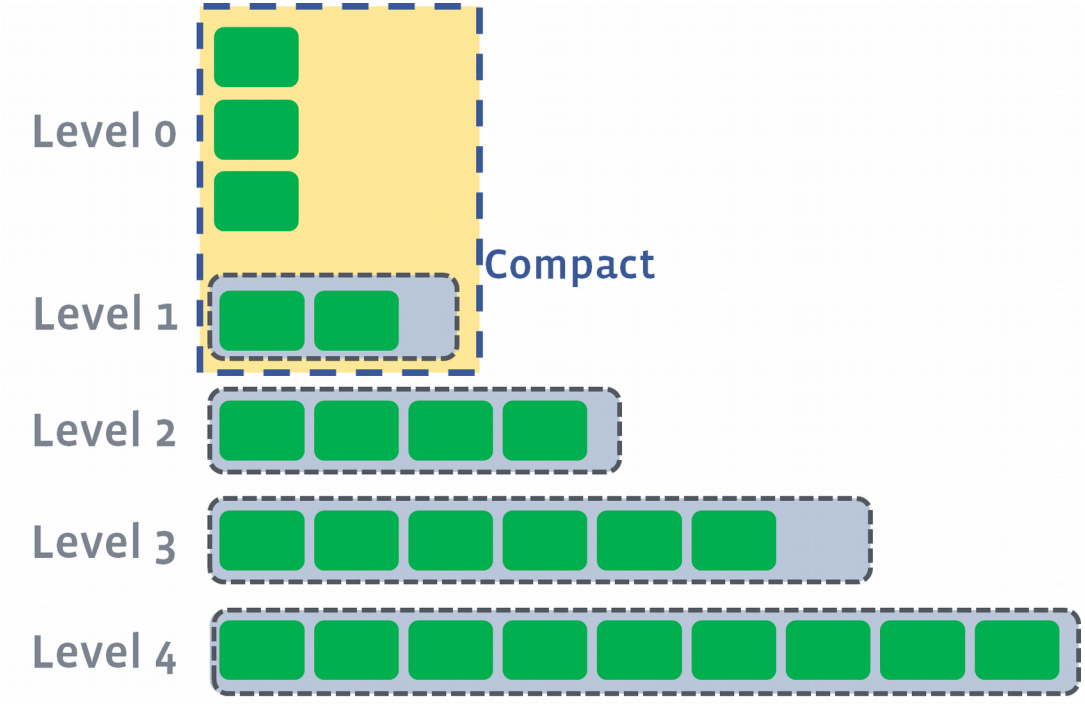
Ключи упорядочены в пределах уровня



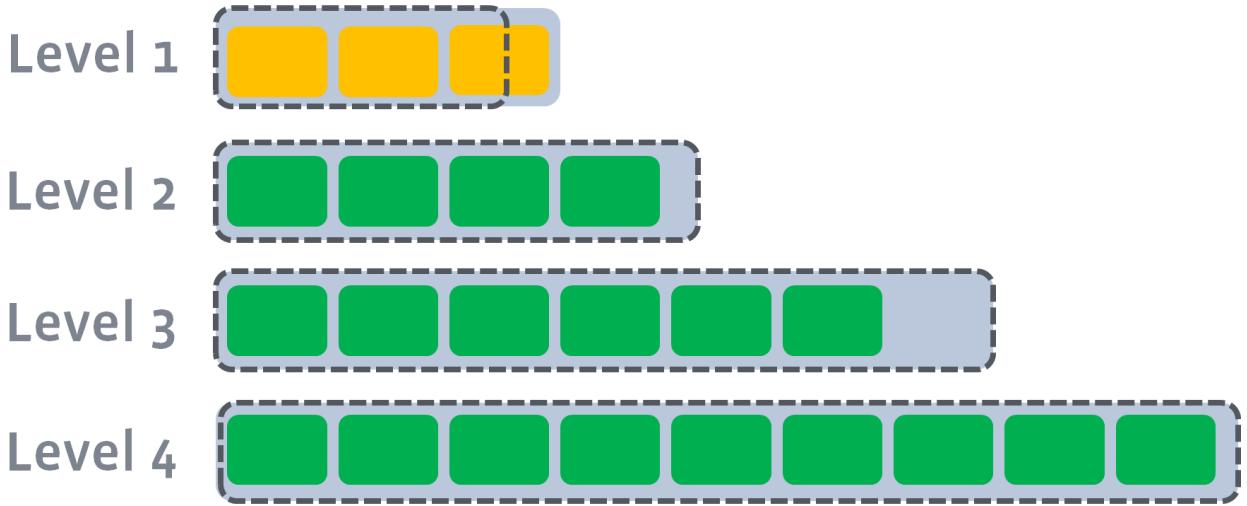
Версионирование ключей



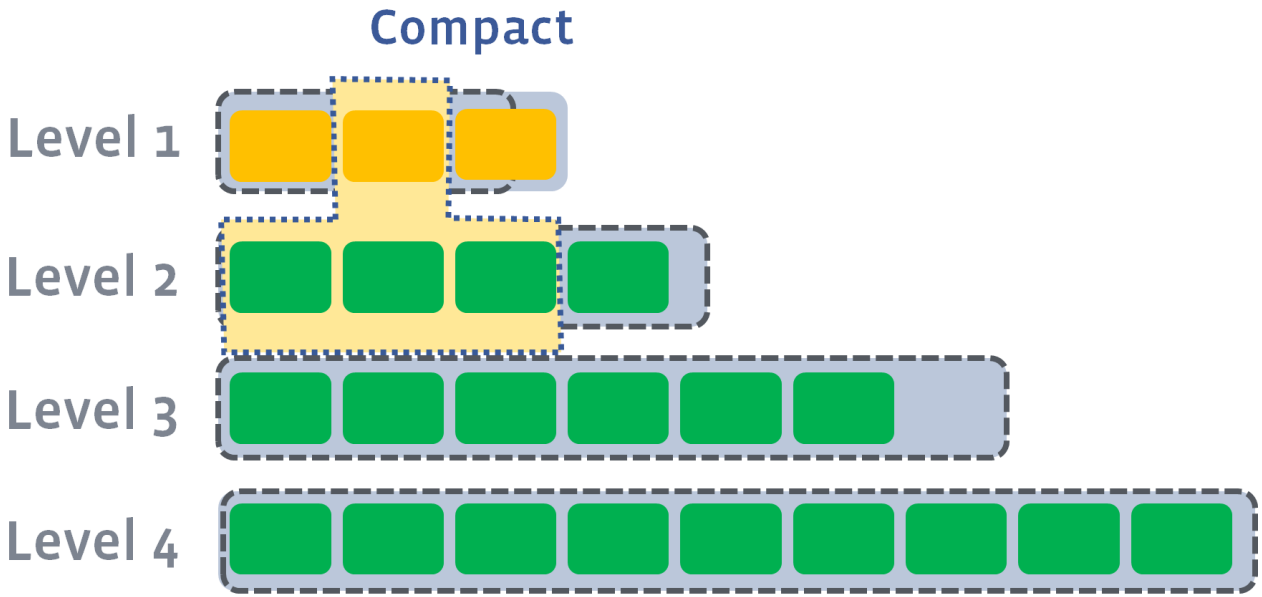
Превышение размера уровнем L0



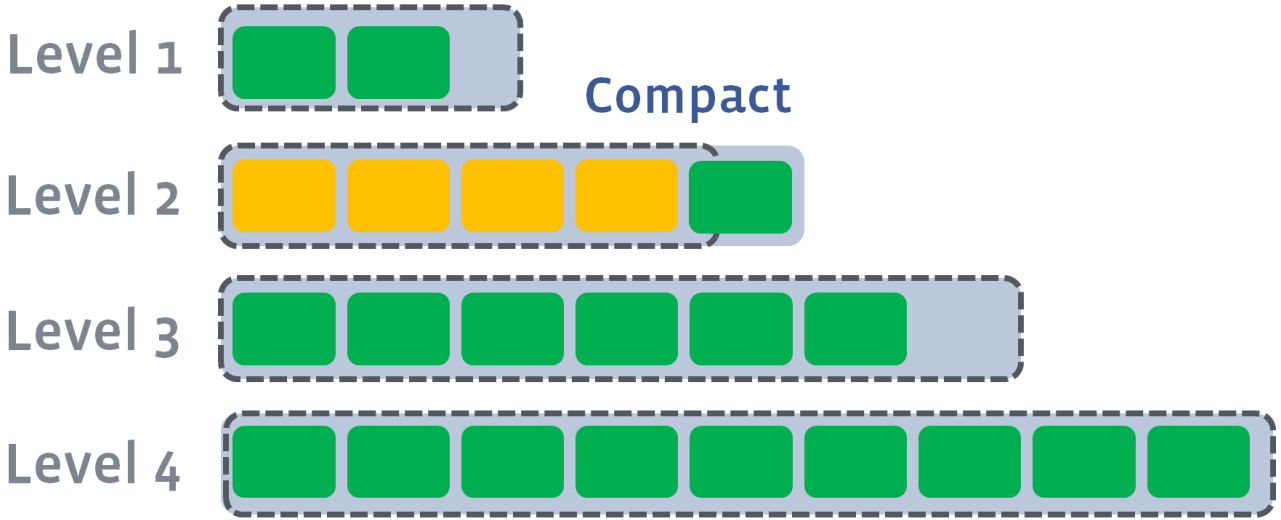
Следующий уровень может превысить порог



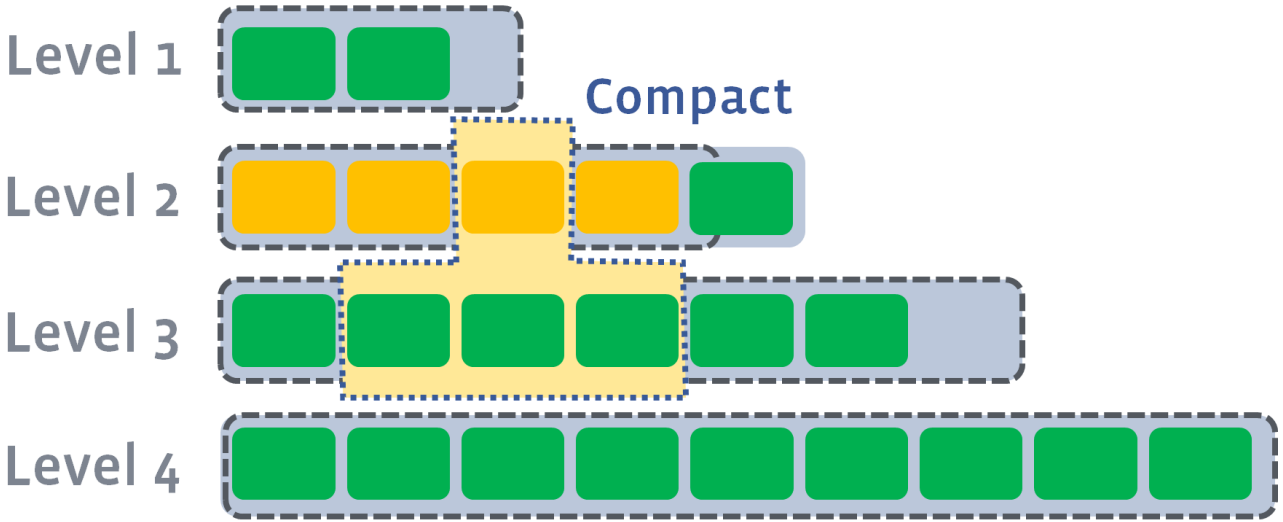
Выбираются файлы нижнего уровня с пересекающимися ключами



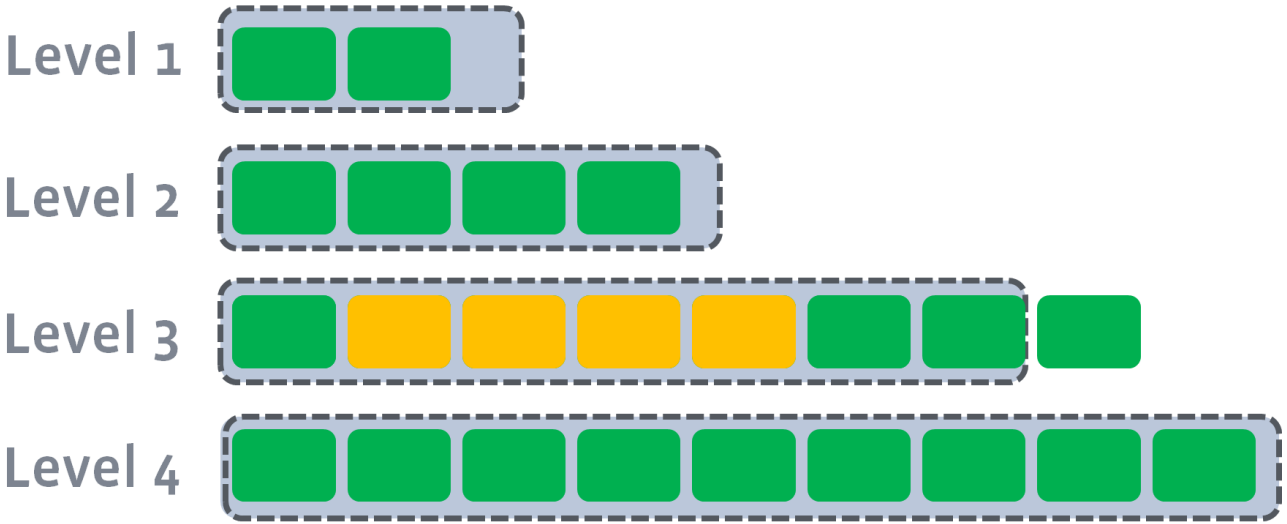
Компактизация может повторяться рекурсивно



Снова выбирается пересекающийся диапазон ниже



Результат



Файлы в уровнях неизменяемы

Компактизация создает новые файлы, а старые удаляются в момент, когда перестают использоваться

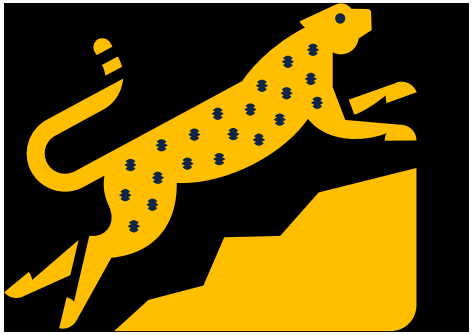
Файлы в уровнях неизменяемы

Компактизация создает новые файлы, а старые удаляются в момент, когда перестают использоваться

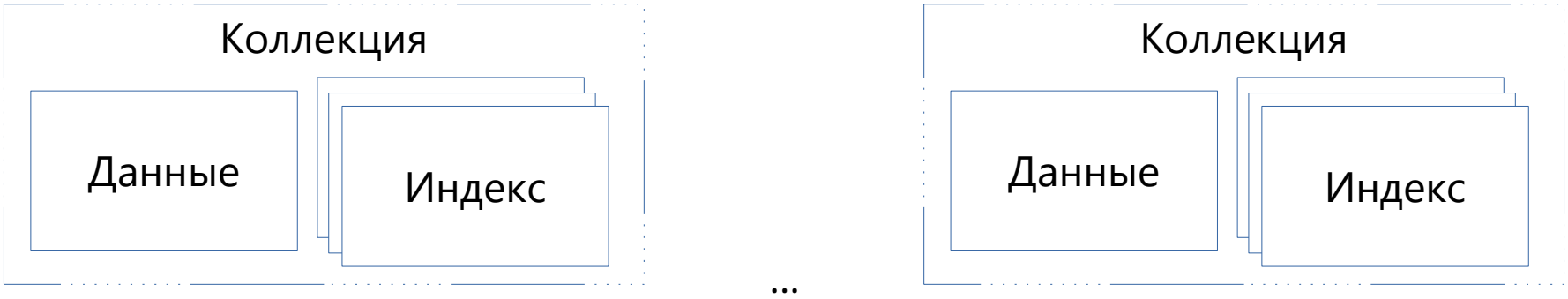
Файлы пишутся на диск последовательно, что ускоряет операции ввода-вывода


mongoDB

+



Данные в представлении MongoDB



Данные в представлении MongoDB

“Контейнеры” данных и индексов получают уникальные строковые идентификаторы ident

Данные в представлении MongoDB

“Контейнеры” данных и индексов получают уникальные строковые идентификаторы ident

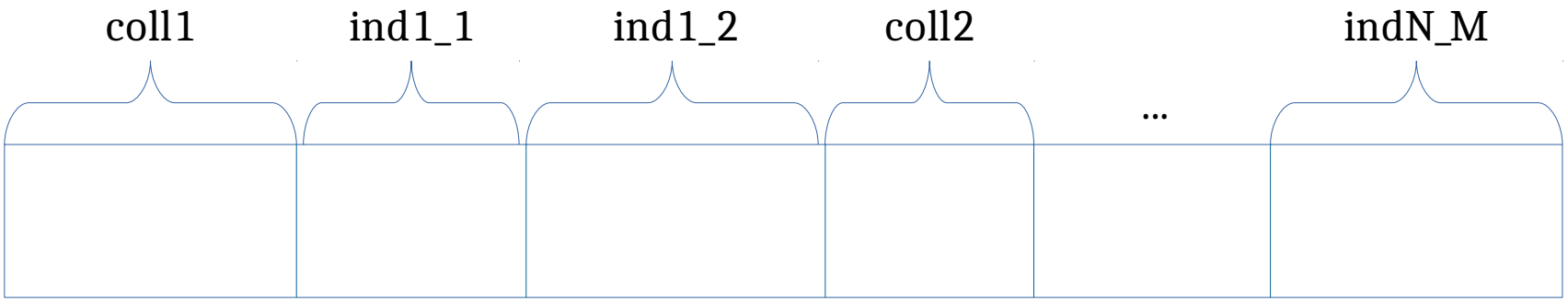
Сами элементы должны иметь уникальный id в пределах контейнера

Данные в представлении RocksDB

Как представить структуру MongoDB в “плоском” хранилище RocksDB?

Данные в представлении RocksDB

<ident + id> для элемента контейнера



Данные в представлении RocksDB

ident > 20 символов - лишние расходы на каждый элемент данных

Данные в представлении RocksDB

ident > 20 символов - лишние расходы на каждый элемент данных

Длина ident вызвана использованием его как имени файла для WiredTiger и mmapv1

Как сэкономить на ident правильно?

Данные в представлении RocksDB

Хэш от ident – плохо, т.к. возможны коллизии при малой длине хэша

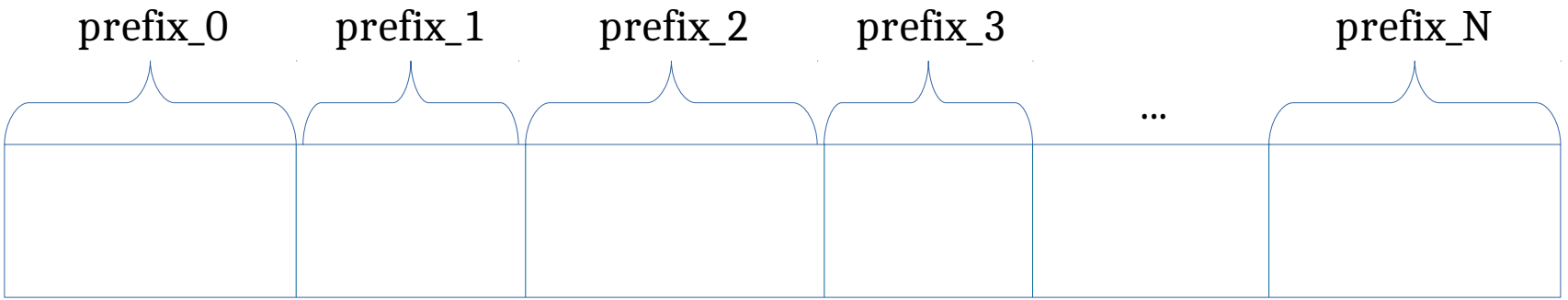
Данные в представлении RocksDB

Хэш от ident – плохо, т.к. возможны коллизии при малой длине хэша

Автоинкрементный счетчик (назв. prefix) и карта соответствия ident → prefix

Данные в представлении RocksDB

<prefix + id> для элемента контейнера



Формат индекса в RocksDB

K = <prefix + value + order + **id (loc)**>

V = <typeof value>

Формат индекса в RocksDB

K = <prefix + value + order + **id (loc)**>

Навязано MongoDB

V = <typeof value>

Как искать id, если он требуется для формирования ключа?

Формат индекса в RocksDB

Хранилище должно поддерживать одну из операций поиска вида **lower_bound** | **upper_bound**

Формат индекса в RocksDB

Хранилище должно поддерживать одну из операций поиска вида **lower_bound** | **upper_bound**

Позволяет спозиционироваться на ближайшее значение и декодировать его

Формат индекса в RocksDB

Хранилище должно поддерживать одну из операций поиска вида **lower_bound** | **upper_bound**

Позволяет спозиционироваться на ближайшее значение и декодировать его

В RocksDB задача решается с помощью курсоров

Самая проблемная операция над данными

Удаление данных в RocksDB

Удаление элемента (документа, индекса) – помещение операции D в LSM-дерево

Удаление данных в RocksDB

Удаление элемента (документа, индекса) – помещение операции D в LSM-дерево

В результате, в дереве скапливается мусор из старых данных и операций удаления, увеличивающий время обхода

Решение есть

Удаление данных в RocksDB

Запрашивать статистику курсора после итерирования по диапазону

Удаление данных в RocksDB

Запрашивать статистику курсора после итерирования по диапазону

Если количество пропущенных данных превысило порог – запустить компактизацию для данного диапазона

Удаление данных в RocksDB

Запрашивать статистику курсора после итерирования по диапазону

Если количество пропущенных данных превысило порог – запустить компактизацию для данного диапазона

Диапазон всегда изолирован в пределах prefix

Это оказалось меньшей проблемой ...

Удаление коллекций в RocksDB

Нужно обойти все данные и индексы коллекции и вызвать для них операции удаления

Удаление коллекций в RocksDB

Нужно обойти все данные и индексы коллекции и вызвать для них операции удаления

Сильно замусоривается хранилище

Удаление коллекций в RocksDB

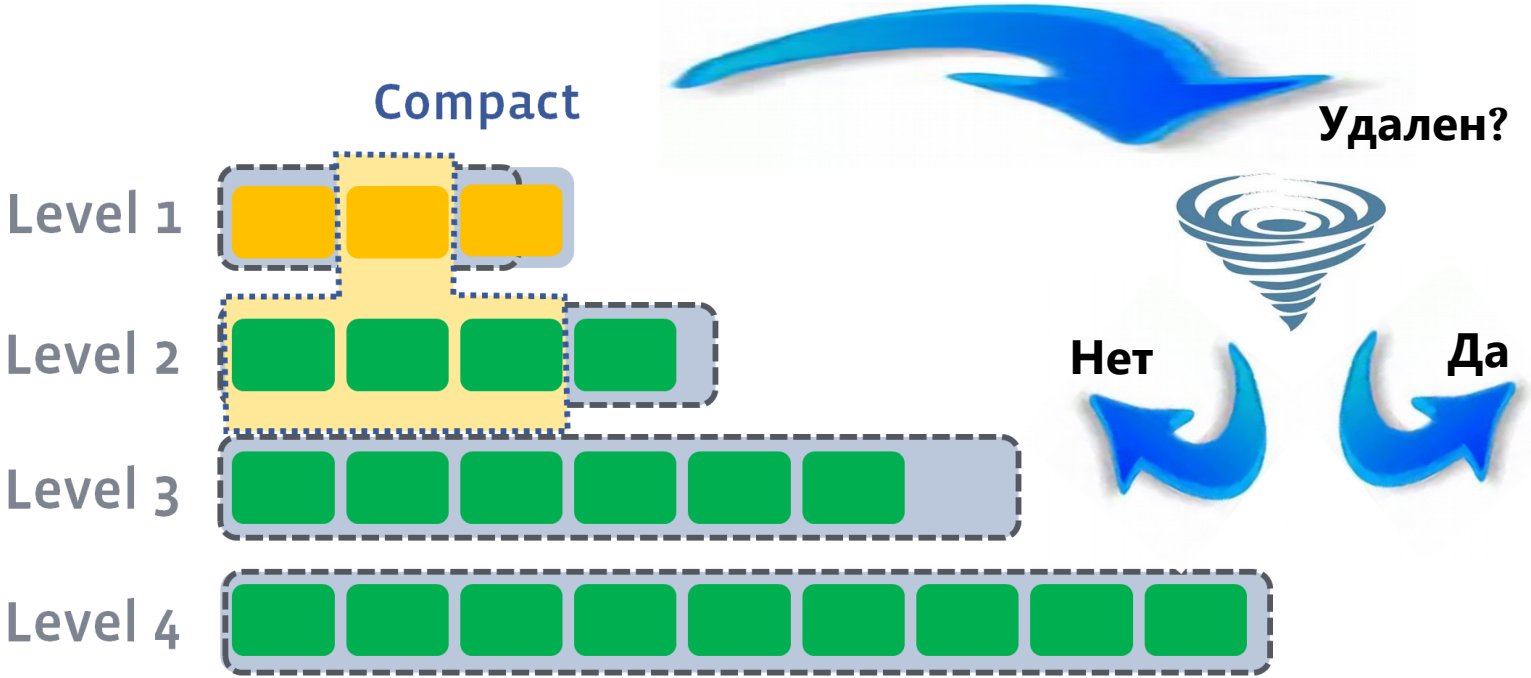
Нужно обойти все данные и индексы коллекции и вызвать для них операции удаления

Сильно замусоривается хранилище

Дорого и бессмысленно в сравнении с движками, где одному дереву соответствует один файл, и достаточно просто удалить группу файлов

Фильтры компактизации

Удаление коллекций в RocksDB



Удаление коллекций в RocksDB

Создается фильтр, в который помещаются prefix всех удаленных контейнеров

Удаление коллекций в RocksDB

Создается фильтр, в который помещаются prefix всех удаленных контейнеров

Запускается компактизация для нужных prefix

Удаление коллекций в RocksDB

Создается фильтр, в который помещаются prefix всех удаленных контейнеров

Запускается компактизация для нужных prefix

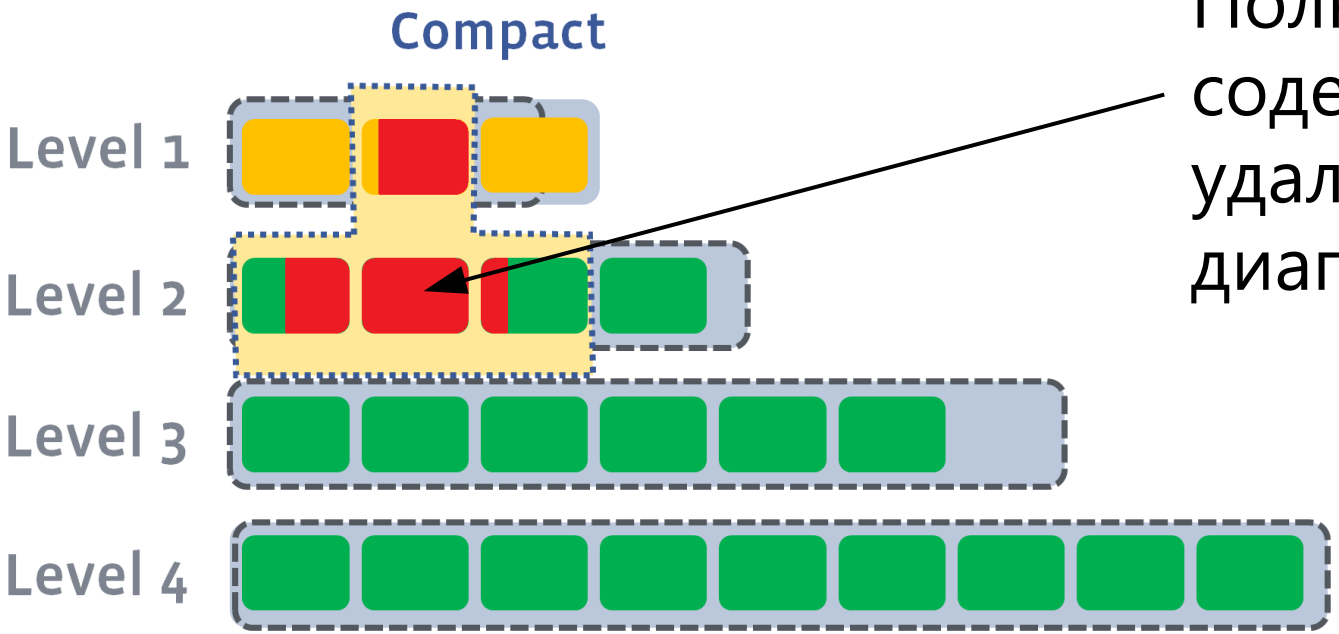
Алгоритм компактизации вызывает фильтр и определяет, нужно ли удалять каждый конкретный ключ из диапазона

Удаление коллекций в RocksDB

Для продолжения очистки в случае перезапуска после аварийного завершения в хранилище записывается маркер о каждом удаляемом prefix до окончания его очистки

Можно еще лучше

Удаление коллекций в RocksDB



Полностью содержит удаляемый диапазон

Удаление коллекций в RocksDB

DeleteFilesInRange позволяет удалить файлы, ключи в которых полностью входят в указанный диапазон

Удаление коллекций в RocksDB

DeleteFilesInRange позволяет удалить файлы, ключи в которых полностью входят в указанный диапазон

Удаляет “живые” ключи, не отмеченные операцией `D`, поэтому требует осторожности

Чего не хватает

Удаление коллекций в RocksDB

MongoDB не посылает нотификаций о логическом удалении коллекции или базы целиком

Удаление коллекций в RocksDB

MongoDB не посылает нотификаций о логическом удалении коллекции или базы целиком

Причина: в WiredTiger и mmapv1 удаляются файлы, и им такая нотификация не требуется

Удаление коллекций в RocksDB

MongoDB не посылает нотификаций о логическом удалении коллекции или базы целиком

Причина: в WiredTiger и mmapv1 удаляются файлы, и им такая нотификация не требуется

В MongoRocks приходится запускать компактизацию на каждый контейнер (prefix) в отдельности

oplog

Capped-коллекции в RocksDB

MongoDB имеет специальный тип коллекций, реализованных по принципу кольцевого буфера

Capped-коллекции в RocksDB

MongoDB имеет специальный тип коллекций, реализованных по принципу кольцевого буфера

Создавались исключительно для работы oplog – лога репликиции

Capped-коллекции в RocksDB

Размер oplog зачастую очень большой (5% от размера диска, не более 50 Гб по умолчанию)

Capped-коллекции в RocksDB

Размер oplog зачастую очень большой (5% от размера диска, не более 50 Гб по умолчанию)

Из-за большого числа перезаписей в oplog скапливается много мусора, что влияет на работу всего хранилища

Capped-коллекции в RocksDB

Отдельный код для мониторинга размера и “замусоренности” oplog

Capped-коллекции в RocksDB

Отдельный код для мониторинга размера и “замусоренности” oplog

Повышенный приоритет операций компактизации для oplog (очередь по типам операций в коде MongoRocks)

Радикальное решение

Колоночные семейства в RocksDB

В классических движках на каждый контейнер (данные или индекс) имеется по одному дереву

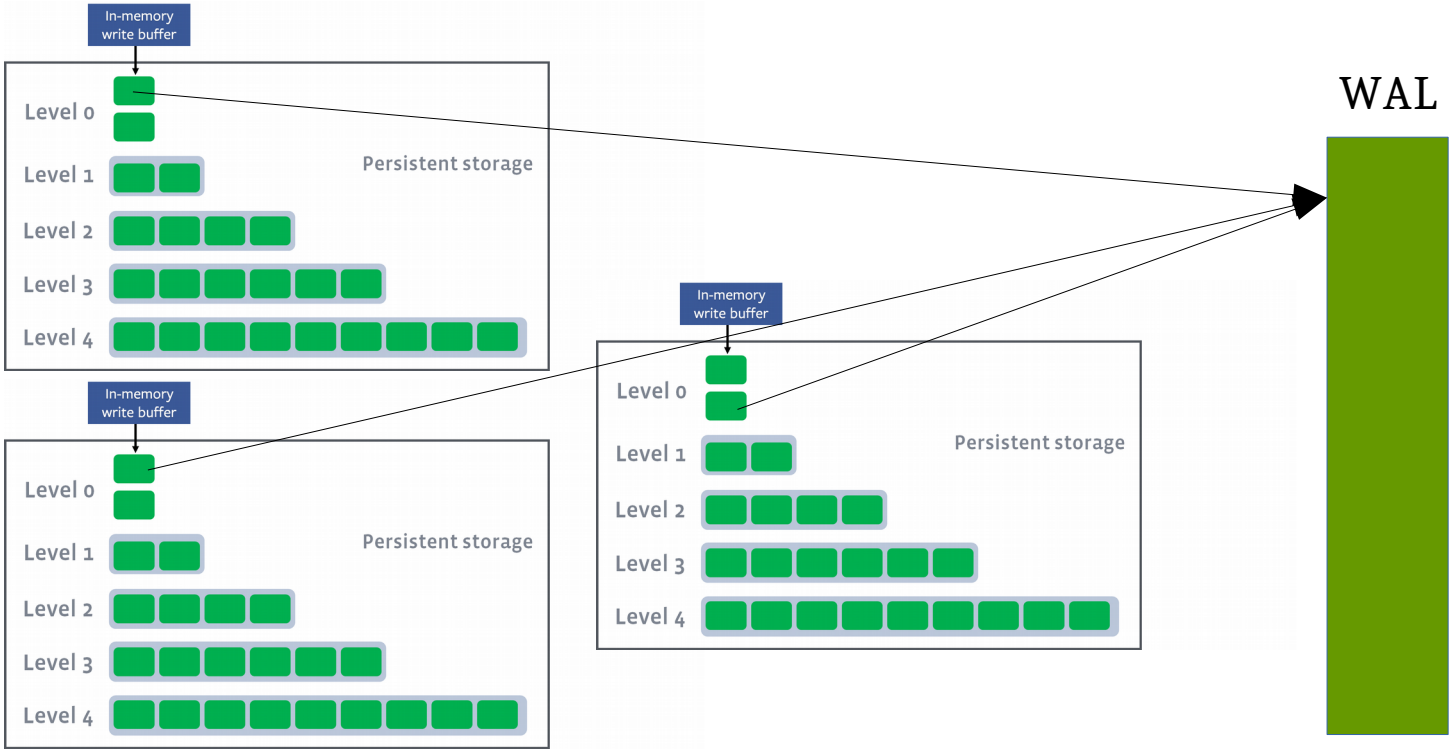
Колоночные семейства в RocksDB

В классических движках на каждый контейнер (данные или индекс) имеется по одному дереву

В MongoRocks одно LSM-дерево на всех

Больше LSM-деревьев!

Колоночные семейства в RocksDB



Колоночные семейства в RocksDB

В RocksDB поддерживается множество LSM-деревьев (колоночных семейств), объединенных журналом WAL для обеспечения транзакционности

Колоночные семейства в RocksDB

В RocksDB поддерживается множество LSM-деревьев (колоночных семейств), объединенных журналом WAL для обеспечения транзакционности

Изначально сделаны и применены с расчетом на MySQL

Колоночные семейства в RocksDB

В RocksDB поддерживается множество LSM-деревьев (колоночных семейств), объединенных журналом WAL для обеспечения транзакционности

Изначально сделаны и применены с расчетом на MySQL

MongoRocks должен иметь одно LSM-дерево на один prefix

Выводы

Контракты MongoDB все еще содержат типовые детали реализации, неприменимые к RocksDB

Выводы

Контракты MongoDB все еще содержат типовые детали реализации, неприменимые к RocksDB

Важно упорядочивать ключи для упрощения работы с ними

Выводы

Контракты MongoDB все еще содержат типовые детали реализации, неприменимые к RocksDB

Важно упорядочивать ключи для упрощения работы с ними

Удаление групп ключей можно решить с применением различных оптимизаций

Выводы

Контракты MongoDB все еще содержат типовые детали реализации, неприменимые к RocksDB

Важно упорядочивать ключи для упрощения работы с ними

Удаление групп ключей можно решить с применением различных оптимизаций

Идея многих LSM-деревьев – шаг вперед

Percona Live Europe Call for Papers & Registration are Open!

Championing Open Source Databases

- MySQL, MongoDB, Open Source Databases
- Time Series Databases, PostgreSQL, RocksDB
- Developers, Business/Case Studies, Operations
- September 25-27th, 2017
- Radisson Blu Royal Hotel, Dublin, Ireland

Submit Your Proposal by July 17th!

www.percona.com/live/e17

Вопросы?

Спасибо за внимание!