НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Факультет компьютерных наук,
Прикладная математика и информатика.

КУРСОВАЯ РАБОТА
Программный проект
“Cache-словари на SSD в ClickHouse”

Выполнил студент группы БПМИ-175
Васильев Н. С.
Научный руководитель:
Миловидов А. Н.

Москва, 2020

Предметная область

- ClickHouse — столбцовая аналитическая СУБД.
- Кэширование данных из внешних источников.
- Особенности использования SSD:
  - Читать / записывать можно только страницей целиком.
  - Удалять можно только блок целиком.
  - Несколько операций могут выполняться одновременно.
  - Ограниченное число удалений.
- Особенности работы словарей ClickHouse:
  - Обработка запросов пачками.

Актуальность задачи

Хранение данных кэша на SSD вместо оперативной памяти поможет

- Увеличить объем кэша;
- Уменьшить стоимость использования кэша;
- Уменьшить потребление оперативной памяти.

Цели и задачи работы

Цель работы

Добавление в СУБД ClickHouse cache-словарей, хранящих данные на SSD, которые учитывают особенности работы ClickHouse и SSD.

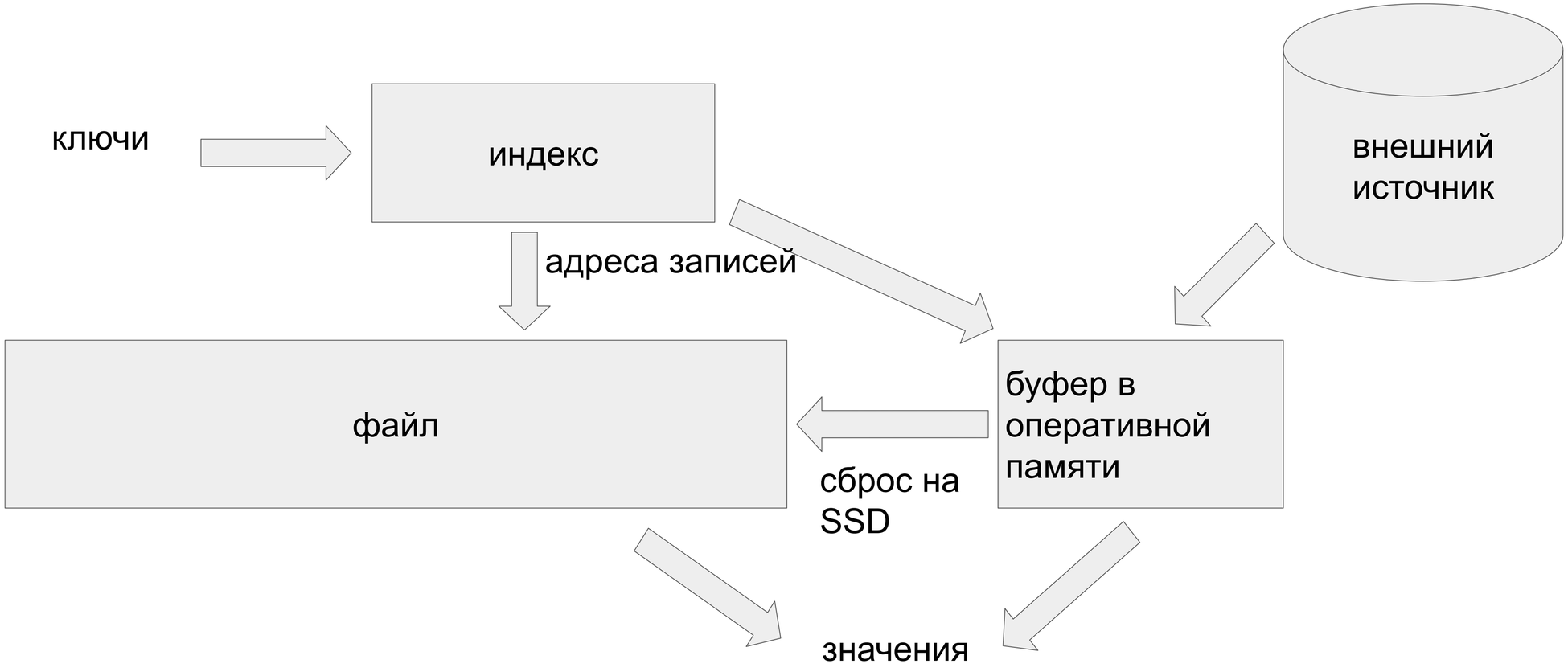
Задачи работы

- Выбор структуры данных для cache-словаря на SSD с учетом особенностей ClickHouse и SSD, которая использует меньше оперативной памяти, чем обычные cache-словари
- Реализация двух cache-словарей на SSD: для произвольных ключей и для ключей типа UInt64.

Существующие решения

	вытеснение из кэша / сборка мусора	поиск	запись	структура на SSD
RocksDB	compaction	бинарный поиск	большими объемами с буфером в ОЗУ	LSM (SST файлы)
Fatcache	FIFO	хеш-таблица в ОЗУ	большими объемами с буфером в ОЗУ	ключ-значение
CaSSanDra	LRU / compaction	хеш-таблица в ОЗУ	большими объемами с буфером в ОЗУ	значения
FlashStore	FIFO с возвращениями	хеш-таблица в ОЗУ	большими объемами с буфером в ОЗУ	ключ-значение
BlueCache	LRU внутри корзины / FIFO	n корзин по 4 элемента в ОЗУ	большими объемами с буфером в ОЗУ	ключ-значение
Flashield	CLOCK / FIFO с возвращениями	хеш-таблица и хеш-функция для адресации внутри блока в ОЗУ	большими объемами с буфером в ОЗУ	ключ-значение внутри блока, адресуемые хеш-функцией

Архитектура cache-словарей на SSD

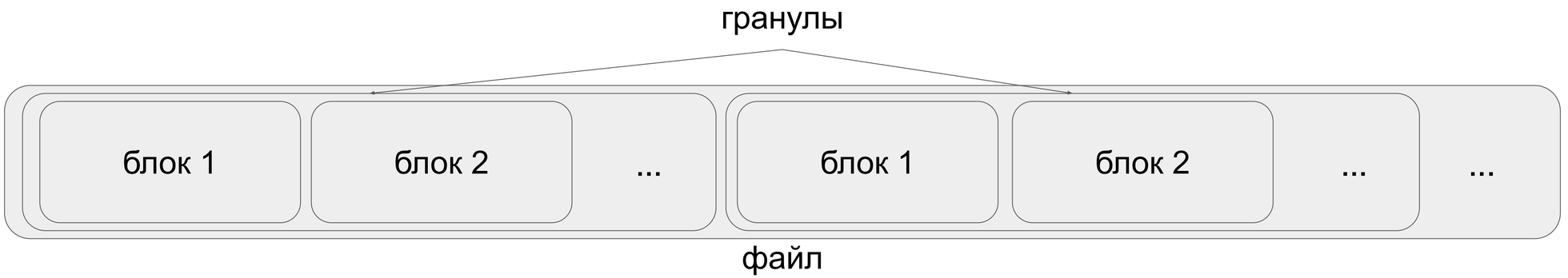


Cache-словари на SSD: хранение данных

- Минимальная единица чтения из файла — блок, который содержит несколько записей.

контрольная сумма	кол-во ключей	ключ	метаданные	значения	...

- Минимальная единица записи — гранула, которая состоит из нескольких блоков. Буфер в ОЗУ состоит из одной гранулы, а файл из нескольких.

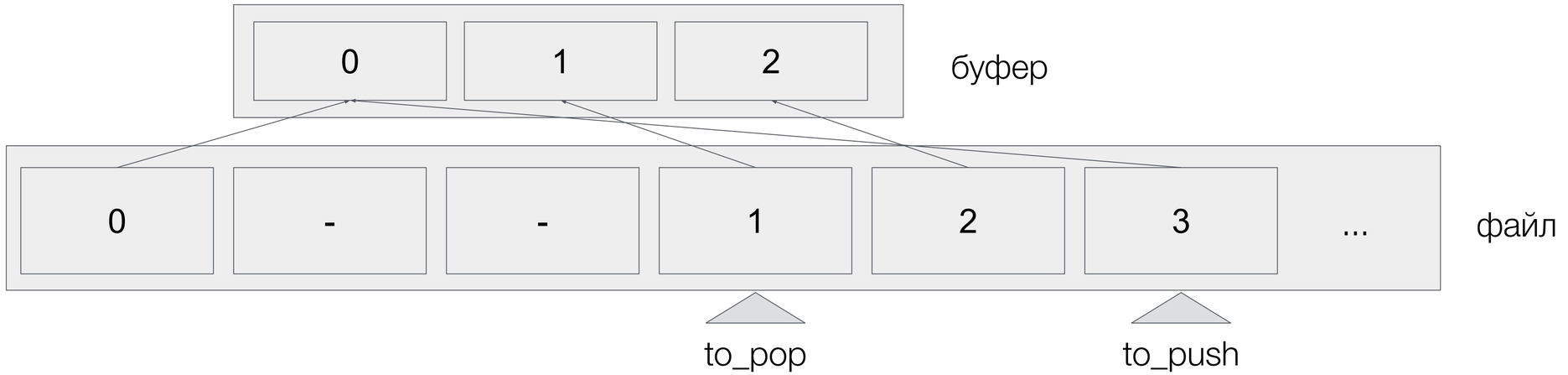


Cache-словари на SSD: запись

- Запись происходит в буфер, который при заполнении сбрасывается в файл на место самой старой гранулы.
- Записываются элементы (ключ, метаданные и значения) таким образом, чтобы каждый лежал целиком в одном блоке.

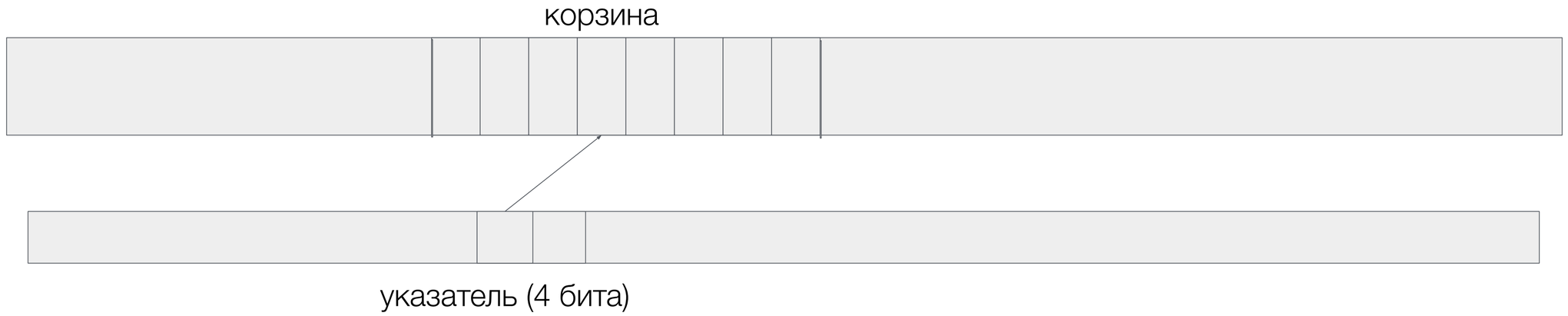
Cache-словари на SSD: чтение

- Чтение из файла сортирует и группирует по блокам адреса записей.
- Блоки читаются целиком.
- Используется асинхронный интерфейс ввода/вывода.
- Позволяет использовать параллелизм SSD.



Cache-словари на SSD: индекс

- Разбит на корзины по 8 элементов.
- Каждый элемент — два 64-битных неотрицательных числа.
- Для вытеснения элементов внутри корзины используется алгоритм FIFO.
- Использует 16.0625 байт для простых ключей и 18.0625 + k байт — для сложных ключей, где k — размер ключа.



Использование: создание словаря

```
ThinkPad-E570 :) CREATE DICTIONARY database_for_dict.ssd_dict_complex_key (     k1 String,     k2 Int32,     a UInt64 DEFA
ULT 0,     b Int32 DEFAULT -1,     c String DEFAULT 'none' ) PRIMARY KEY k1, k2 SOURCE(CLICKHOUSE(HOST 'localhost' PORT 90
00 USER 'default' TABLE 'table_for_dict' PASSWORD '' DB 'database_for_dict')) LIFETIME(MIN 1000 MAX 2000) LAYOUT(COMPLEX_K
EY_SSD_CACHE(FILE_SIZE 1073741824 MAX_STORED_KEYS 2097152 BLOCK_SIZE 4096 WRITE_BUFFER_SIZE 1048576 READ_BUFFER_SIZE 10485
76 PATH '/var/lib/clickhouse/clickhouse_dicts/0d'));

CREATE DICTIONARY database_for_dict.ssd_dict_complex_key
(
    `k1` String,
    `k2` Int32,
    `a` UInt64 DEFAULT 0,
    `b` Int32 DEFAULT -1,
    `c` String DEFAULT 'none'
)
PRIMARY KEY k1, k2
SOURCE(CLICKHOUSE(HOST 'localhost' PORT 9000 USER 'default' TABLE 'table_for_dict' PASSWORD '' DB 'database_for_dict'))
LIFETIME(MIN 1000 MAX 2000)
LAYOUT(COMPLEX_KEY_SSD_CACHE(FILE_SIZE 1073741824 MAX_STORED_KEYS 2097152 BLOCK_SIZE 4096 WRITE_BUFFER_SIZE 1048576 READ_B
UFFER_SIZE 1048576 PATH '/var/lib/clickhouse/clickhouse_dicts/0d'))

Ok.

0 rows in set. Elapsed: 0.064 sec.
```

Использование: запросы к словарю

```
ThinkPad-E570 :)  SELECT dictGetUInt64('database_for_dict.ssd_dict_complex_key', 'a', tuple('test', toInt32(3))) AS number, dictGetString('data
base_for_dict.ssd_dict_complex_key', 'c', tuple('test', toInt32(3))) AS string, dictHas('database_for_dict.ssd_dict_complex_key', tuple('test',
 toInt32(3))) AS has

SELECT
    dictGetUInt64('database_for_dict.ssd_dict_complex_key', 'a', ('test', toInt32(3))) AS number,
    dictGetString('database_for_dict.ssd_dict_complex_key', 'c', ('test', toInt32(3))) AS string,
    dictHas('database_for_dict.ssd_dict_complex_key', ('test', toInt32(3))) AS has

┌─number─┬─string─────┬─has─┐
│    100 │ clickhouse │   1 │
└────────┴────────────┴─────┘

1 rows in set. Elapsed: 1.230 sec.

ThinkPad-E570 :)  SELECT dictGetUInt64('database_for_dict.ssd_dict_complex_key', 'a', tuple('test', toInt32(3))) AS number, dictGetString('data
base_for_dict.ssd_dict_complex_key', 'c', tuple('test', toInt32(3))) AS string, dictHas('database_for_dict.ssd_dict_complex_key', tuple('test',
 toInt32(3))) AS has

SELECT
    dictGetUInt64('database_for_dict.ssd_dict_complex_key', 'a', ('test', toInt32(3))) AS number,
    dictGetString('database_for_dict.ssd_dict_complex_key', 'c', ('test', toInt32(3))) AS string,
    dictHas('database_for_dict.ssd_dict_complex_key', ('test', toInt32(3))) AS has

┌─number─┬─string─────┬─has─┐
│    100 │ clickhouse │   1 │
└────────┴────────────┴─────┘

1 rows in set. Elapsed: 0.008 sec.
```

Тестирование

словарь	запись (300 тыс. ключей), с.	чтение (30 тыс. ключей), с.	чтение (100 тыс. ключей), с.	использование оперативной памяти, байт
ram_dict	11.462	0.007	0.026	100659280
ssd_dict_small (4Кбайт)	103.015	0.173	0.346	34611185
ssd_dict_dict (16Кбайт)	26.369	0.135	0.299	34623473
ssd_dict1m (1Мбайт)	1.207	0.11	0.261	35655665
ssd_dict_large (1Гб)	0.457	0.012	0.029	1108348913

Заключение

- Для cache-словарей на SSD выбрана достаточно производительная структура данных. При её выборе учтены особенности работы ClickHouse и SSD.
- Реализованы два вида cache-словарей на SSD.

Дальнейшее развитие

- Гранулярные блокировки.
- Асинхронный сброс буфера.
- Улучшение алгоритмов вытеснения кэша и сборки мусора.
- Сжатие блоков словаря.
- Использование io_uring.
- Перенос составных ключей на SSD.
- …

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ