RocksDB Festival

RF3_Team_WAL

Supported by IITP, StarLab.

July 5, 2021

김민준, 이빈

alswnssl0528@naver.com,
32183118@dankook.ac.kr

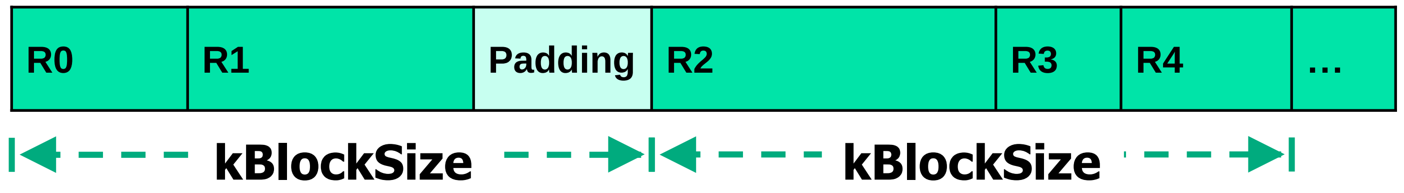
Team Name

RocksDB Festival

- Content
  - ✓ WAL Log File Format
  - ✓ Experiment : is kBlockSize affected performance?
    - Info
    - Hypothesis
    - Result
    - Discussion1
    - Discussion2
  - ✓ Next assignment : WAL performance according to value or key size

RocksDB Festival : Log File Format

- WAL Log File Format



Rn : variable size records

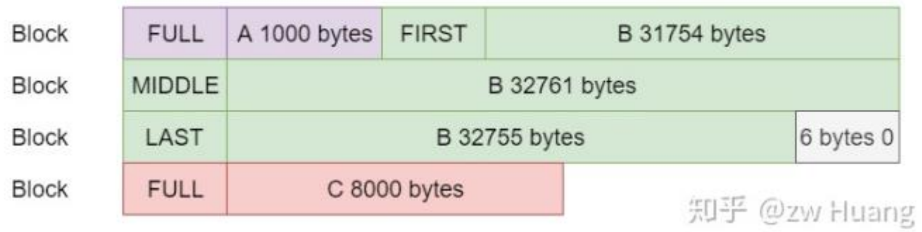
- ✓ Consists of a sequence of **variable** length records.
- ✓ Records are grouped by **kBlockSize**(32k).
- ✓ If a certain record cannot fit into the leftover space (leftover < Rn), then the leftover space is **padded** with empty (null) data.
- ✓ If record is bigger than kBlockSize, record occurs **fragmentation.**

RocksDB Festival : Log File Format

The Legacy Record Format

CRC (4B)	Size(2B)	Type(1B)	Payload

- Record consists of CRC, Size, Type, Payload
  - CRC(Cyclic Redundancy Check) : Verifies the integrity of the WAL
  - Size : Length of the record size
  - Type : kZeroType, kFullType, kFirstType, kLastType, kMiddleType



  - Payload : The actual value of the key-value is written

References : https://zhuanlan.zhihu.com/p/258091002

RocksDB Festival : Log File Format

The Recyclable Record Format

CRC (4B)	**Size(2B)**	**Type(1B)**	**Log number (4B)**	**Payload**

- ✓ Record consists of CRC, Size, Type, Log Number, Payload
  - CRC, Size, Payload : same as the components of the legacy record format.
  - Type : kRecyclableFullType, kRecyclableFirstType,
    kRecyclableMiddleType, kRecyclableLastType
  - Log Number : Distinguish between the previous log writer and the last one. (32bit)

RocksDB Festival : kBlockSize

- Hardware Environment : D

D	
CPU	1 * AMD Ryzen 5 3500X 6-Core
OS	Ubuntu 20.04.2 LTS
SSD	mx500

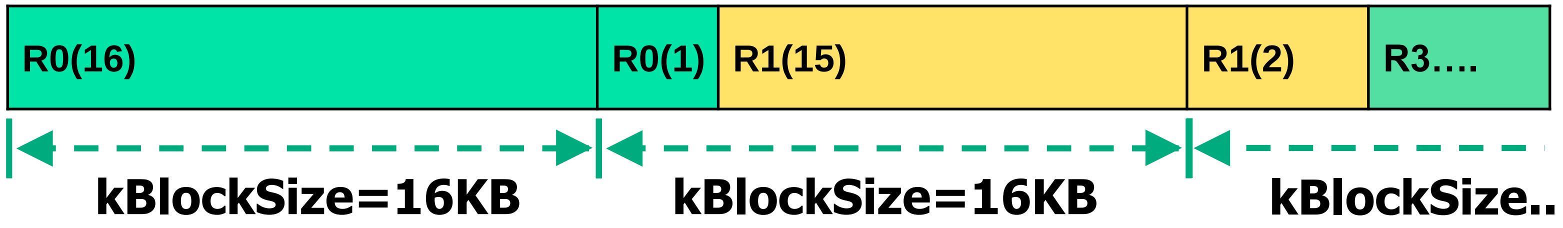
RocksDB Festival : kBlockSize

- Experiment Info.
  - ✓ WAL overhead measurement according to the kBlockSize
  - ✓ Because of the kBlockSize affect the size of padding, WAL overhead will change according to the kBlockSize.
  - ✓ Conditions
    - kBlockSize = 4KB, 8KB, 16KB, 32KB(default), 64KB

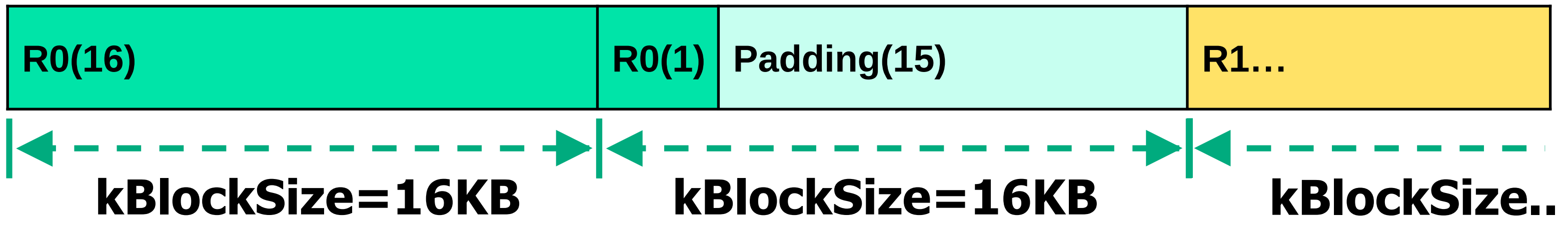
    [db_bench Option]
    - benchmarks ="fillseq" , "fillrandom"
    - disable_wal = false, true
    - value_size = 16byte, 32byte, 64byte, 128byte, 256byte, 512byte, 1024byte, 2048byte, 3072byte, 4096byte, 5120byte, 6144byte, 7168byte 8192byte 16384byte

RocksDB Festival : kBlockSize

- Hypothesis - If record size is bigger than kBlockSize
  - ✓ If kBlockSize = 16KB, record size = 17KB, num=100



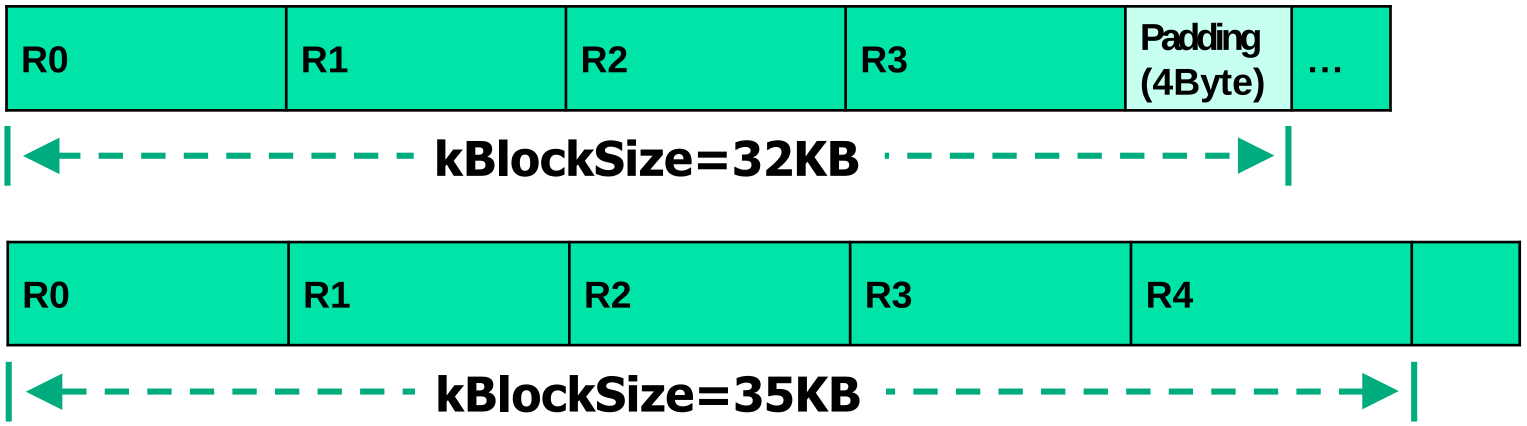
  - ✓ Padding is **not exist**, predict performance improve.
  - ✓ But, Wouldn't fragmentation cause **consistency** issues?



  - ✓ Padding is exist, overhead is too big.

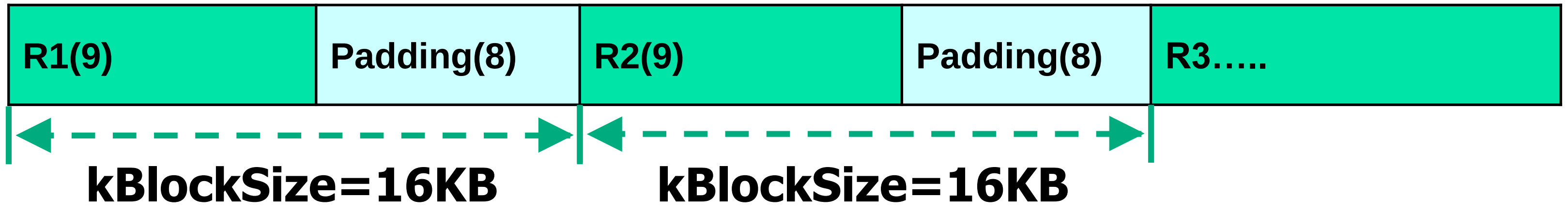
RocksDB Festival : kBlockSize

- Hypothesis - If kBlockSize is bigger than record size
  - ✓ Size of Padding = kBlockSize % Size of Recode(fixed)
  - ✓ Ex. Size of record = 7KB, num=20
    - kBlockSize = 32KB, Size of Padding = 4KB
      → Total 160KB = 140KB + **20KB** (Higher overhead)
    - kBlockSize = 35KB, Size of Padding = 0KB
      → Total 140KB = 140KB + **0KB** (Lower overhead)



RocksDB Festival : kBlockSize

- Hypothesis - Extreme situations
  - ✓ If kBlockSize = 16KB, record size = 9KB, num=100

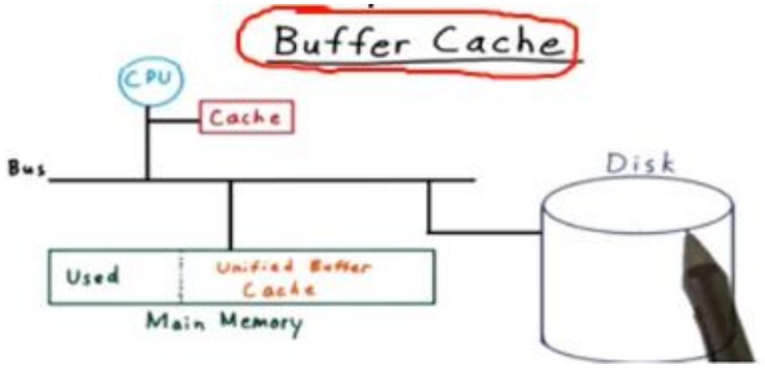


  - ✓ Padding is extremely high size
    - Expected performance degradation
  - ✓ Write in DB = 900KB, Write in Storage = 1600KB
    - Write Amplification is so high

RocksDB Festival : kBlockSize

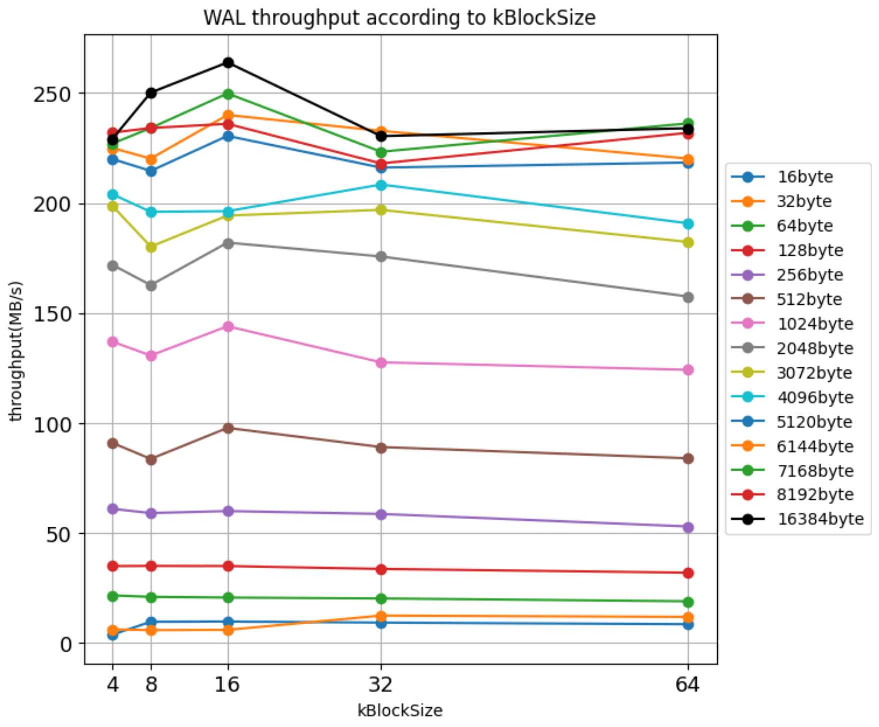
- Method for decreasing padding size
  - ✓ Predict payload size
  - ✓ Determine kBlockSize considering payload and padding size

- Despite of disadvantage, Why kBlockSize is used in Log File?
  - ✓ For delayed write in OS, managing static size is efficiently for processing (buffer cache)
  - ✓ Like Paging!!

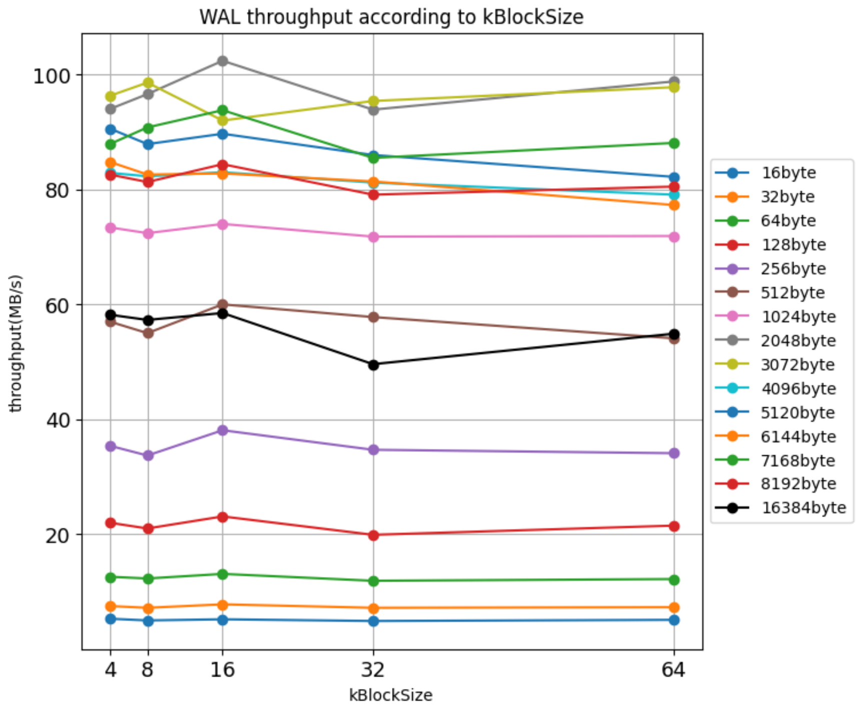


RocksDB Festival : kBlockSize

- **Result 1**
  - ✓ kBlockSizes do not affect WAL overhead



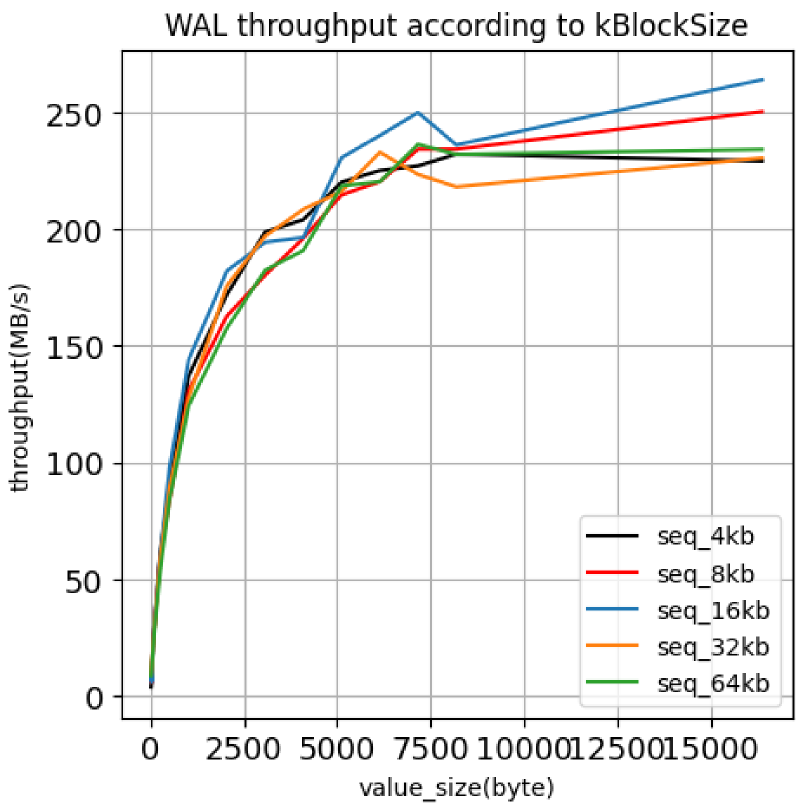
fillseq



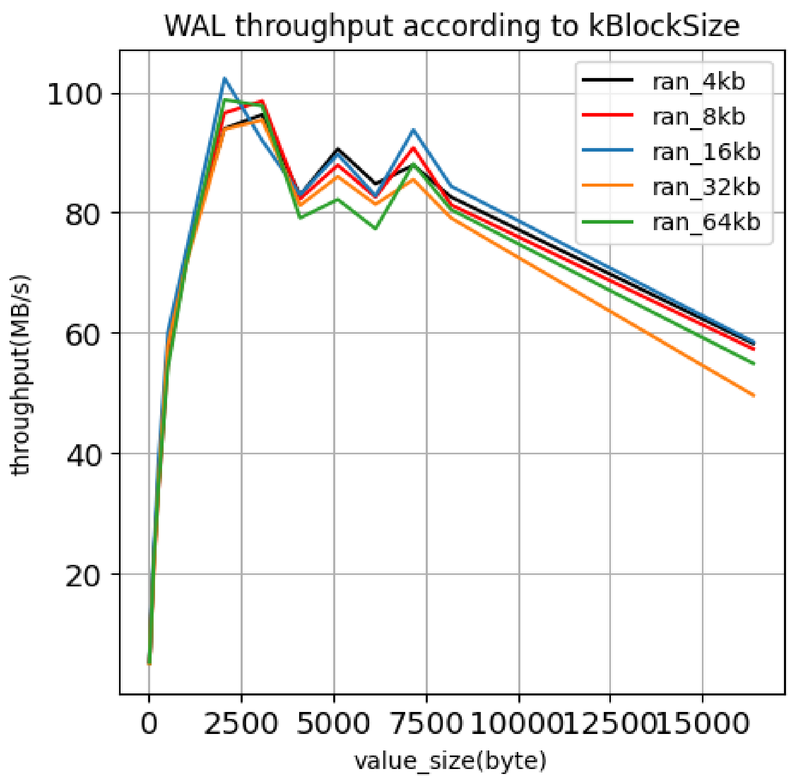
fillrandom

RocksDB Festival : kBlockSize

- Result 2
  - ✓ kBlockSizes do not affect WAL overhead



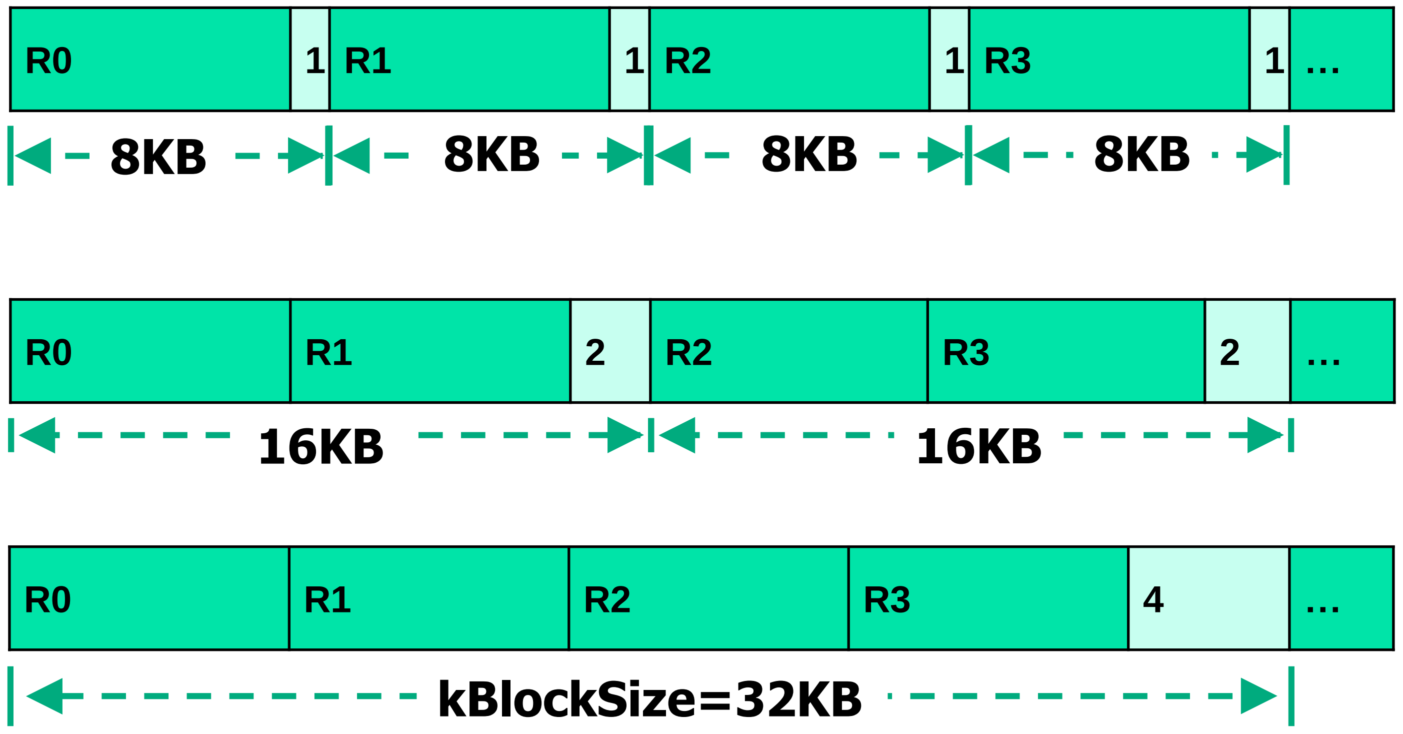
fillseq



fillrandom

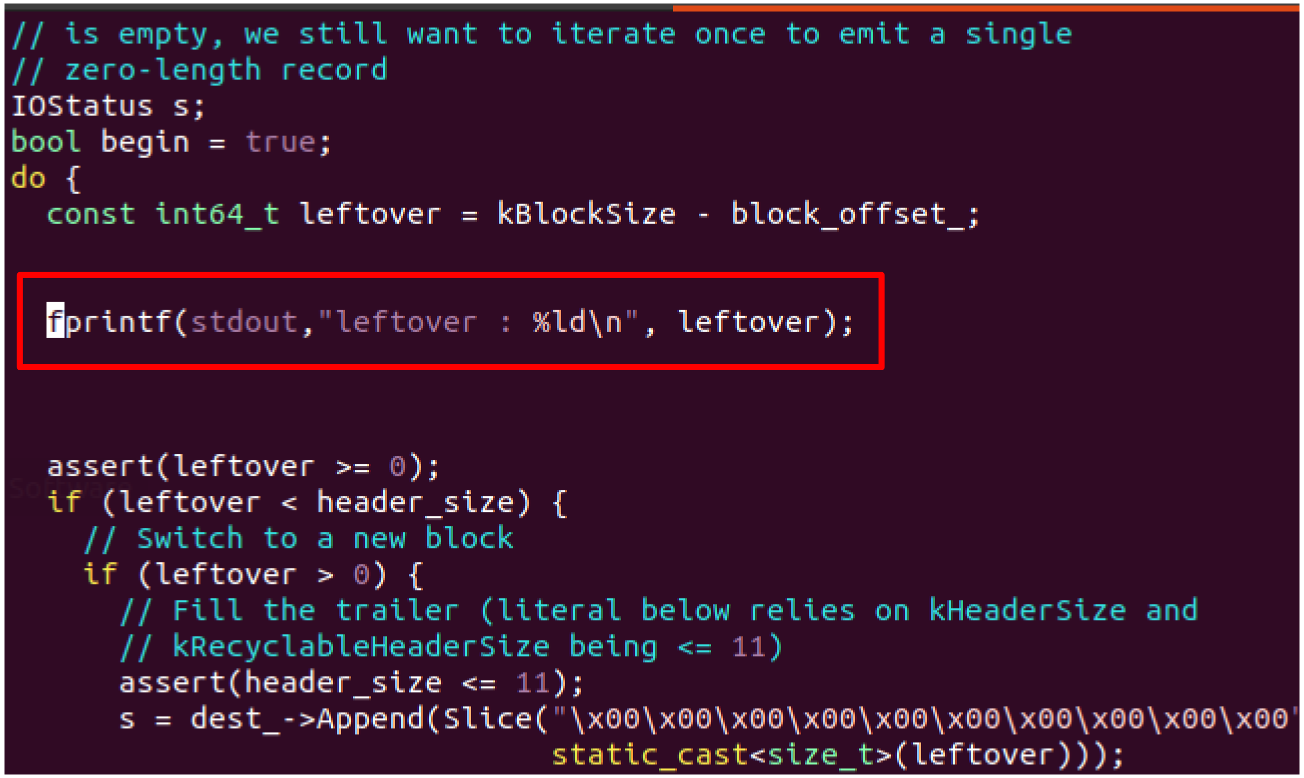
RocksDB Festival : kBlockSize

- Discussion1
  - ✓ The sum of padding sizes is constant.
  - ✓ Size of record = 7KB



RocksDB Festival : kBlockSize

- Discussion2
  - ✓ Padding is **not exist!**
  - ✓ Additional experiments to observe padding

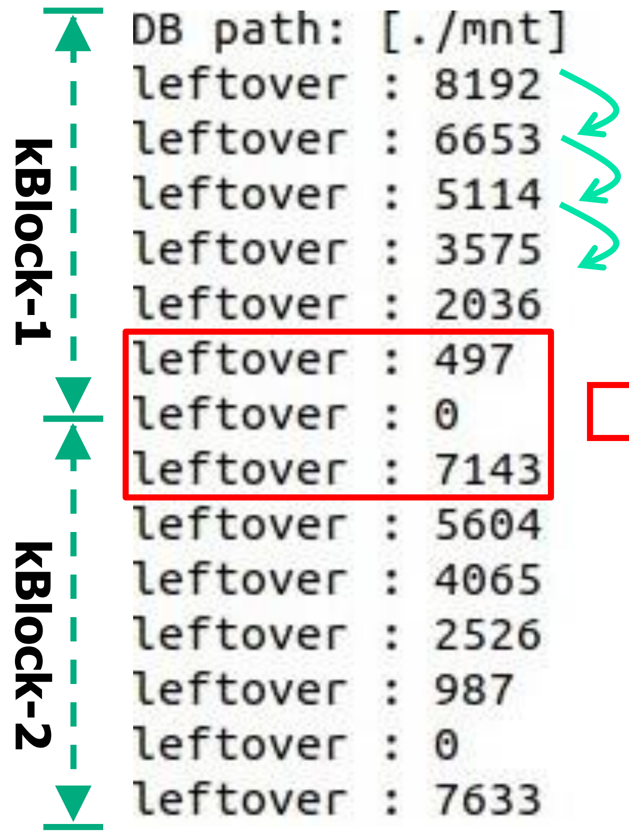


```
// is empty, we still want to iterate once to emit a single
// zero-length record
IOStatus s;
bool begin = true;
do {
  const int64_t leftover = kBlockSize - block_offset_;

  fprintf(stdout,"leftover : %ld\n", leftover);


  assert(leftover >= 0);
  if (leftover < header_size) {
    // Switch to a new block
    if (leftover > 0) {
      // Fill the trailer (literal below relies on kHeaderSize and
      // kRecyclableHeaderSize being <= 11)
      assert(header_size <= 11);
      s = dest_->Append(Slice("\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
                              static_cast<size_t>(leftover)));
```

log_writer.cc

RocksDB Festival : kBlockSize

- Discussion2
  - ✓ Kblocksize : 8KB, Key Size : 16byte, Value Size : 1500byte
  - ✓ leftover = kBlockSize - block_offset_;



```
DB path: [./mnt]
leftover : 8192
leftover : 6653
leftover : 5114
leftover : 3575
leftover : 2036
leftover : 497
leftover : 0
leftover : 7143
leftover : 5604
leftover : 4065
leftover : 2526
leftover : 987
leftover : 0
leftover : 7633
```

8192 - 6653 = 1539
6653 - 5114 = 1539
5114 - 3575 = 1539

Size of record : 1539byte

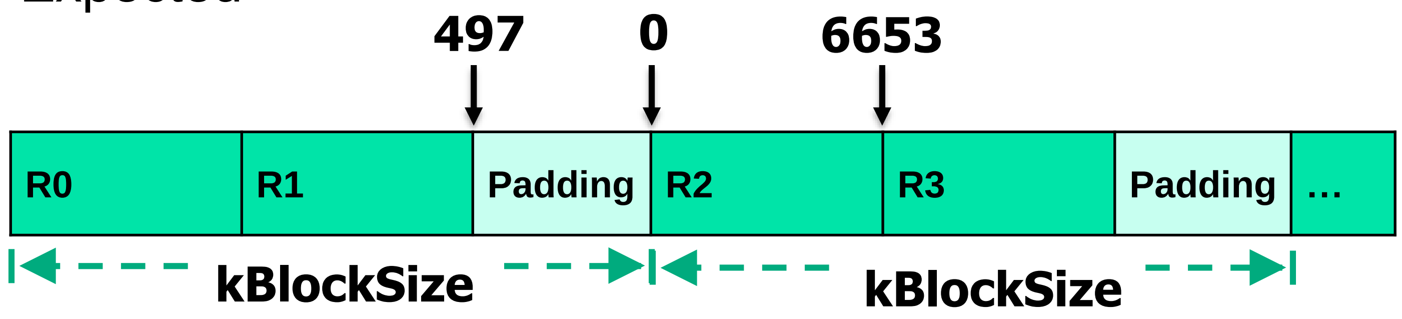
Fragmentation!

8192 – 7143 = 1049
497 + 1049 = 1539 + 7 = 1546 byte

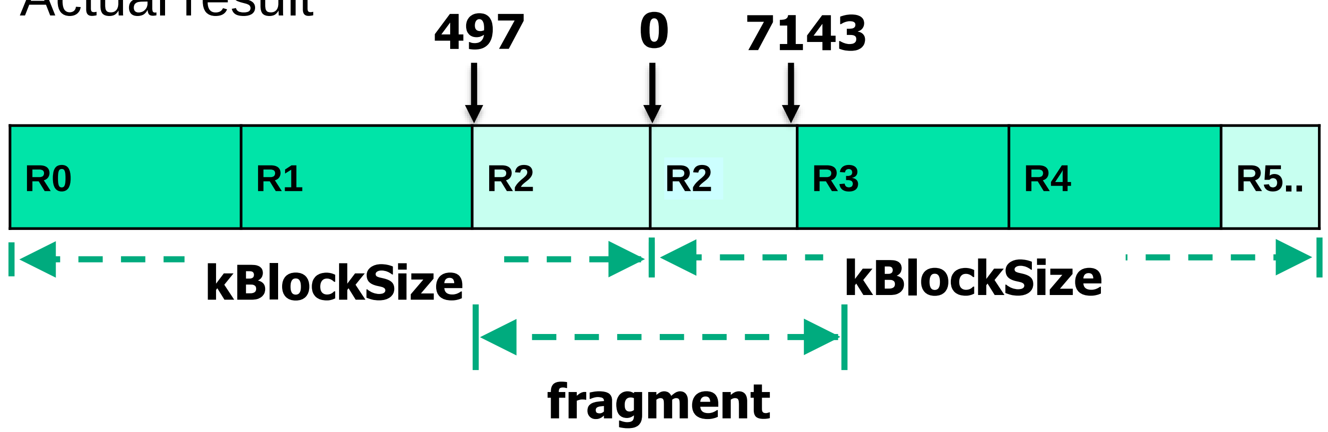
Where is padding??

RocksDB Festival : kBlockSize

- Discussion2 - Is padding existed in kBlock?
  - ✓ Expected

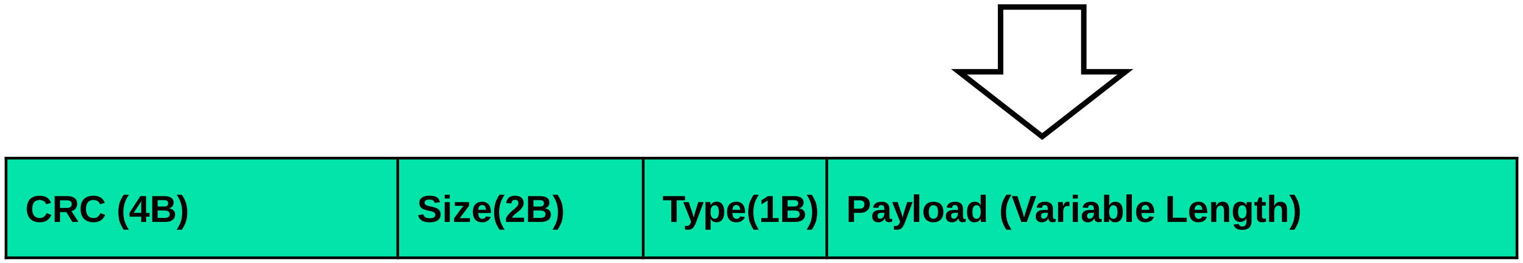


  - ✓ Actual result



RocksDB Festival : key / value size

- Next assignment
  - ✓ WAL performance according to **value** or **key size**



  - ✓ db_bench options
    - --disable_wal=[boolean]
    - --key_size=[int value]
    - --value_size=[int value]

Discussion