RocksDB Festival

RF3_Team_WAL

Supported by IITP, StarLab.

July 5, 2021

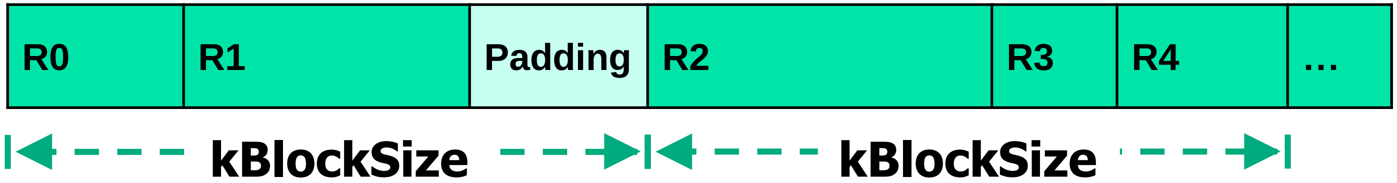
김민준, 이빈
alswnssl0528@naver.com,
32183118@dankook.ac.kr
Team Name

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
  - ✓ Appendix : Type

RocksDB Festival : Log File Format

- WAL Log File Format



R0 | R1 | Padding | R2 | R3 | R4 | …

kBlockSize kBlockSize

Rn : variable size records

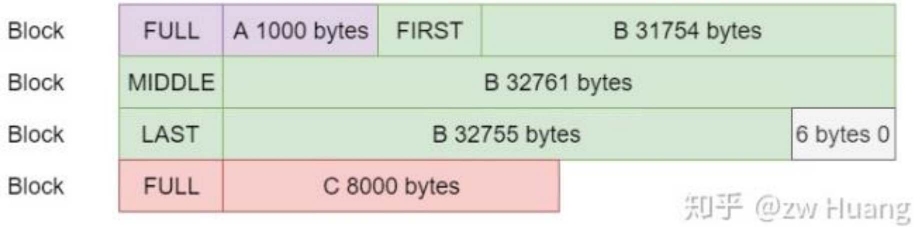
- ✓ Consists of a sequence of **variable** length records.
- ✓ Records are grouped by **kBlockSize**(32k).
- ✓ If a certain record cannot fit into the leftover space (leftover < Rn), then the leftover space is **padded** with empty (null) data.
- ✓ If record is bigger than kBlockSize, record occurs **fragmentation.**

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

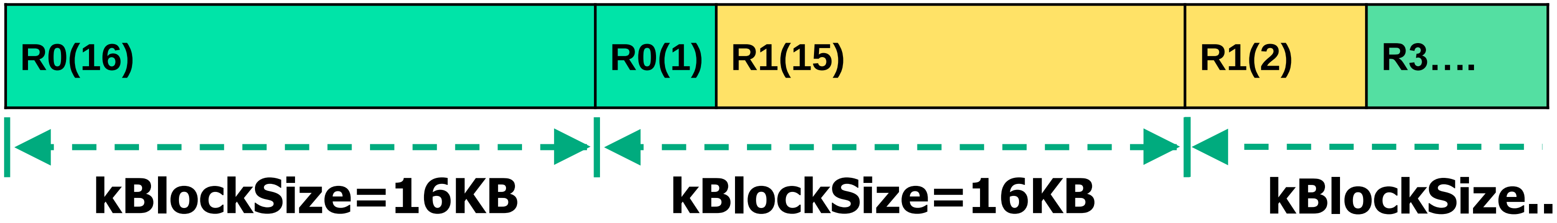
- Experiment Info.
  - ✓ WAL overhead measurement according to the kBlockSize
  - ✓ Because of the kBlockSize affect the size of padding, WAL overhead will change according to the kBlockSize.
  - ✓ Conditions
    - kBlockSize = 4KB, 8KB, 16KB, 32KB(default), 64KB

    [db_bench Option]

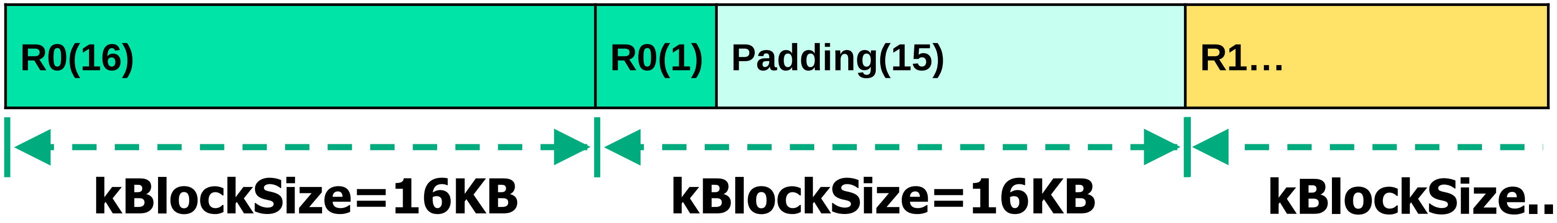
    - benchmarks ="fillseq" , "fillrandom"
    - disable_wal = false, true
    - value_size = 16byte, 32byte, 64byte, 128byte, 256byte, 512byte,
      1024byte, 2048byte, 3072byte, 4096byte, 5120byte,
      6144byte, 7168byte 8192byte 16384byte

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?



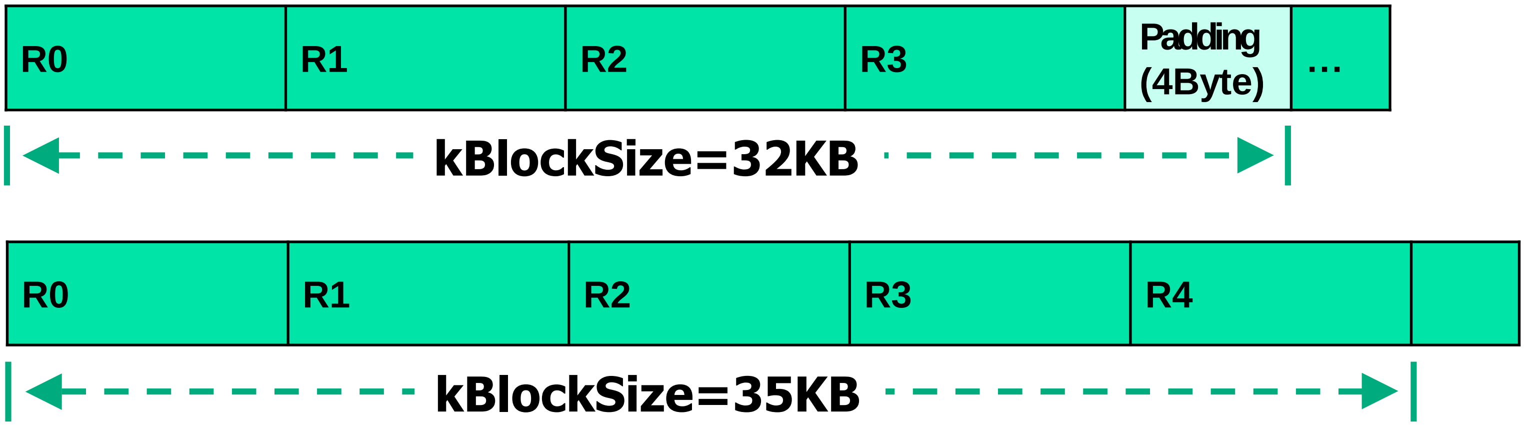
  - ✓ Padding is exist, overhead is too big.

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)



R0 | R1 | R2 | R3 | Padding (4Byte) | …

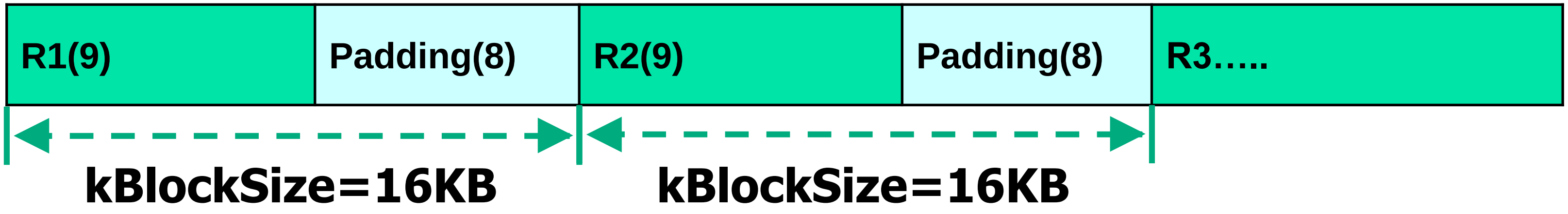
kBlockSize=32KB

R0 | R1 | R2 | R3 | R4 |

kBlockSize=35KB

RocksDB Festival : kBlockSize

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

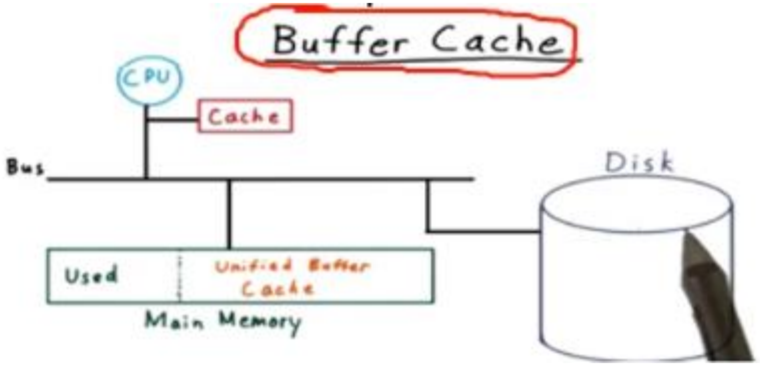


  - ✓ Padding is extremely high size
    - Expected performance degradation
  - ✓ Write in DB = 900KB, Write in Storage = 1600KB
    - Write Amplification is so high

RocksDB Festival : kBlockSize

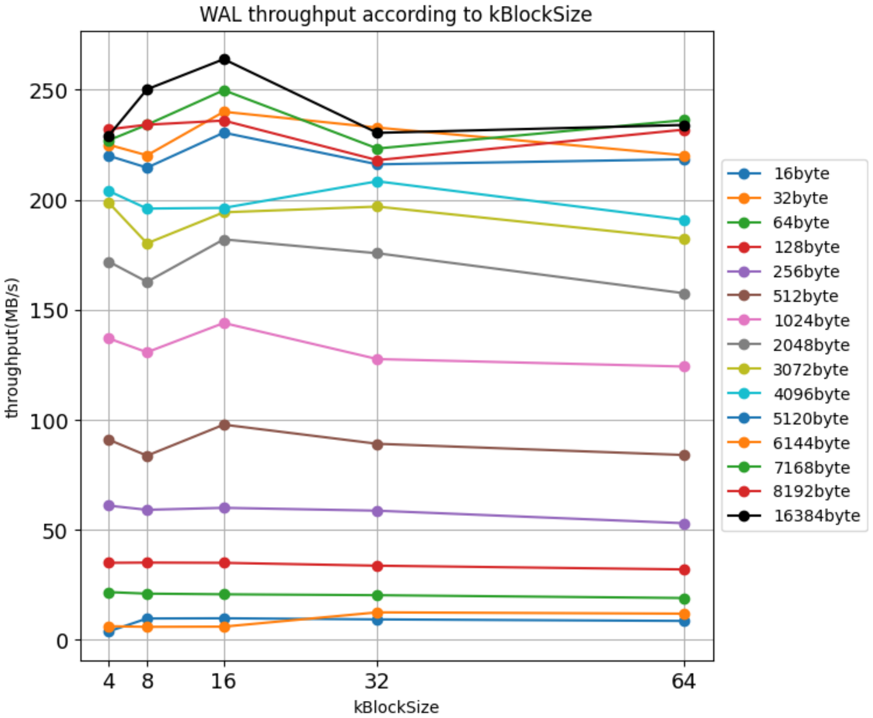
- Method for decreasing padding size
  - ✓ Predict payload size
  - ✓ Determine kBlockSize considering payload and padding size

- 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!!

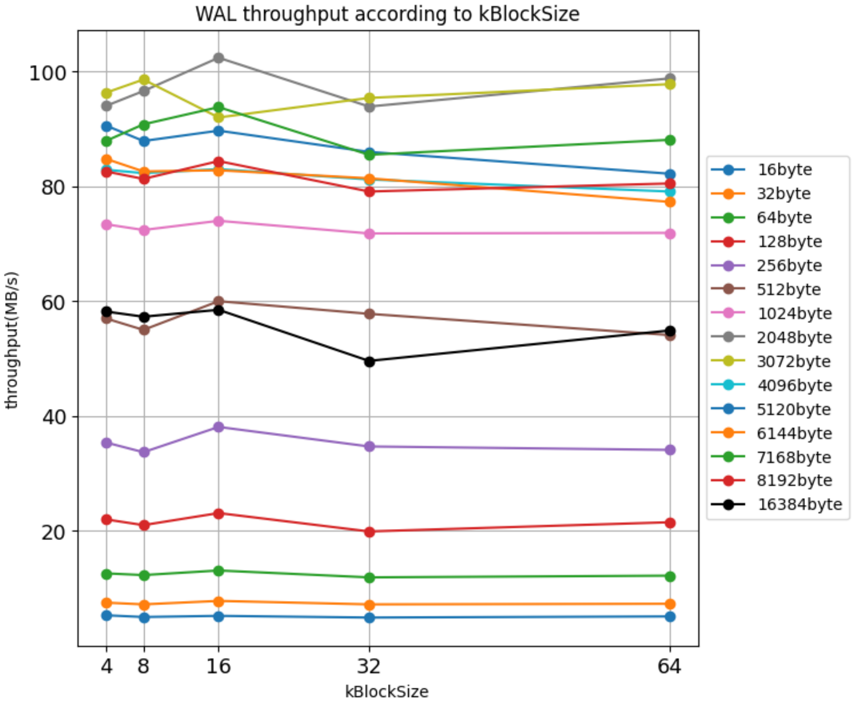


RocksDB Festival : kBlockSize

- Result 1
  - ✓ kBlockSizes do not affect WAL overhead



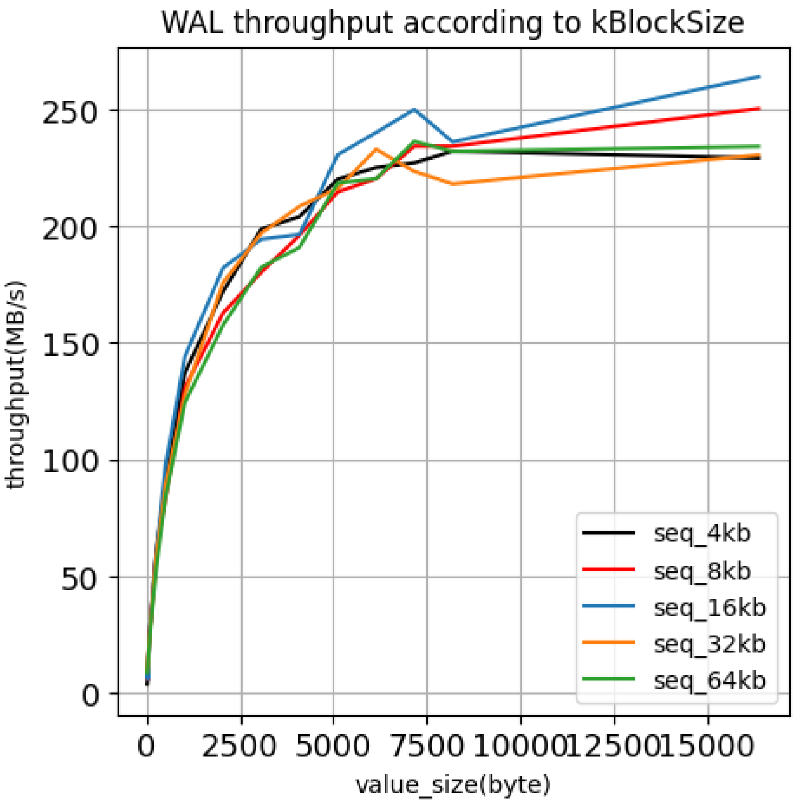
fillseq



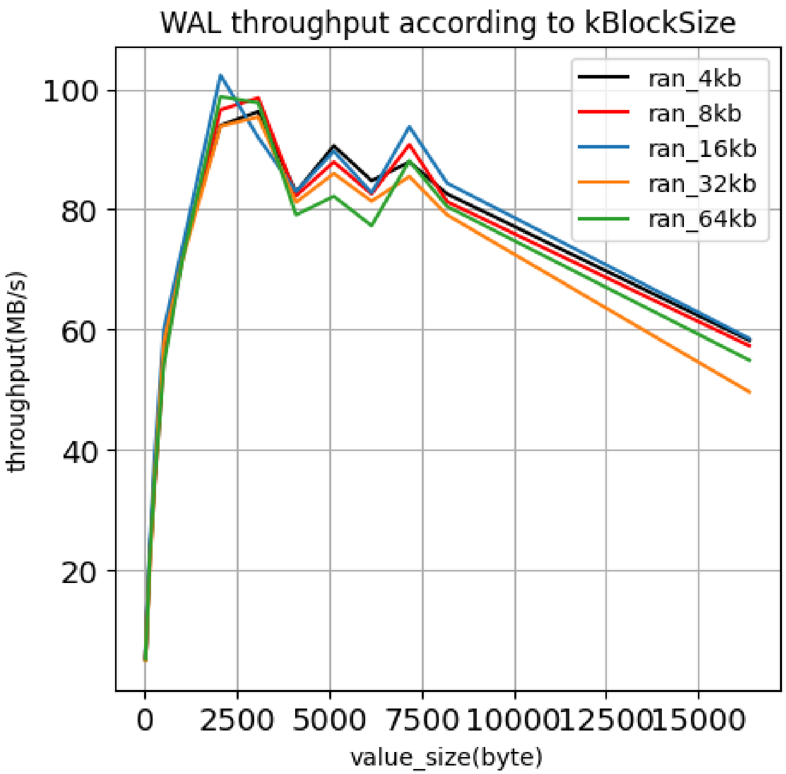
fillrandom

RocksDB Festival : kBlockSize

- Result 2
  - ✓ kBlockSizes do not affect WAL overhead



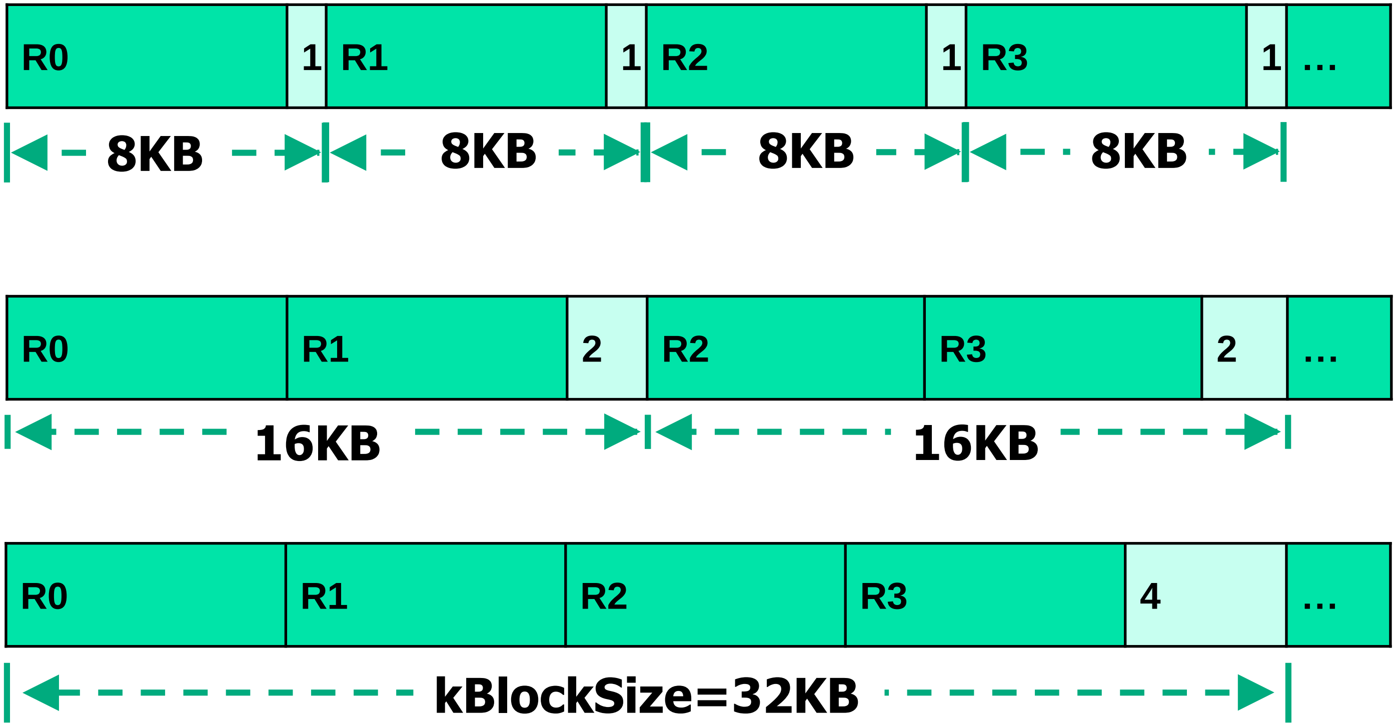
fillseq



fillrandom

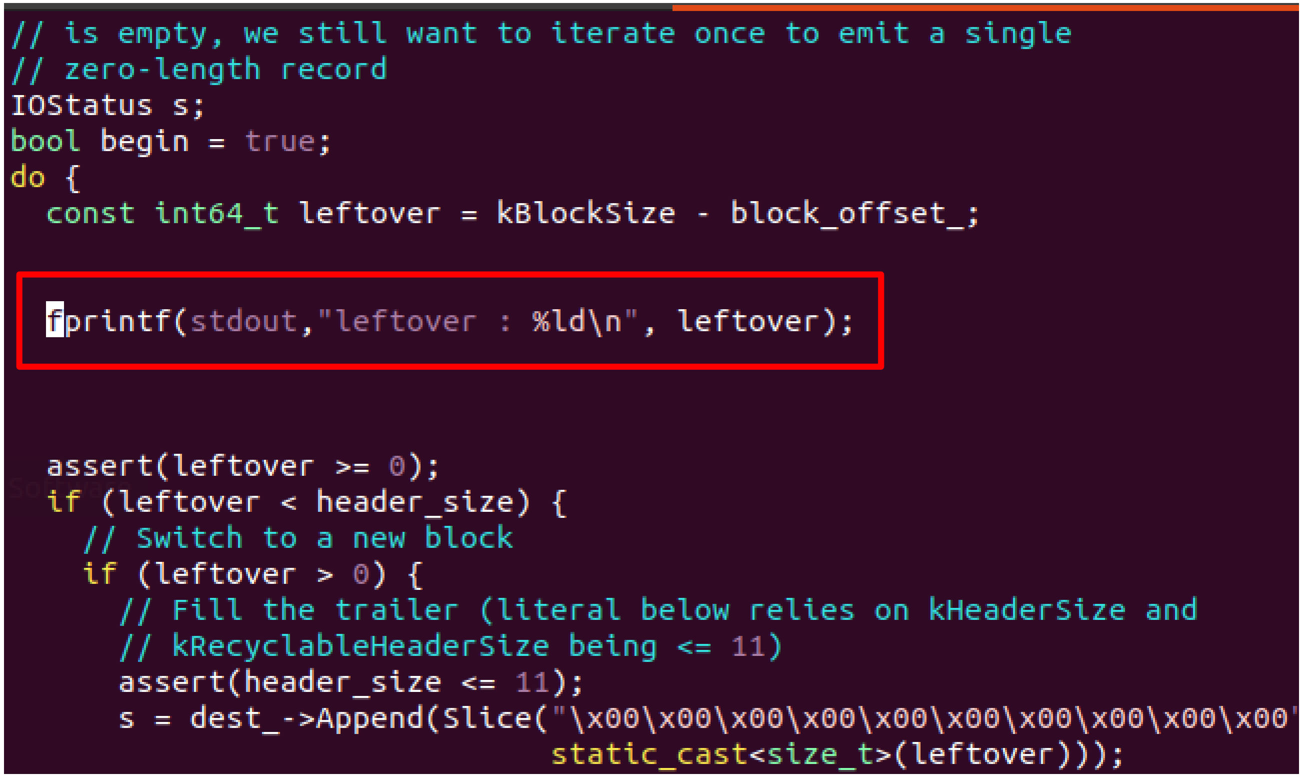
RocksDB Festival : kBlockSize

- 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

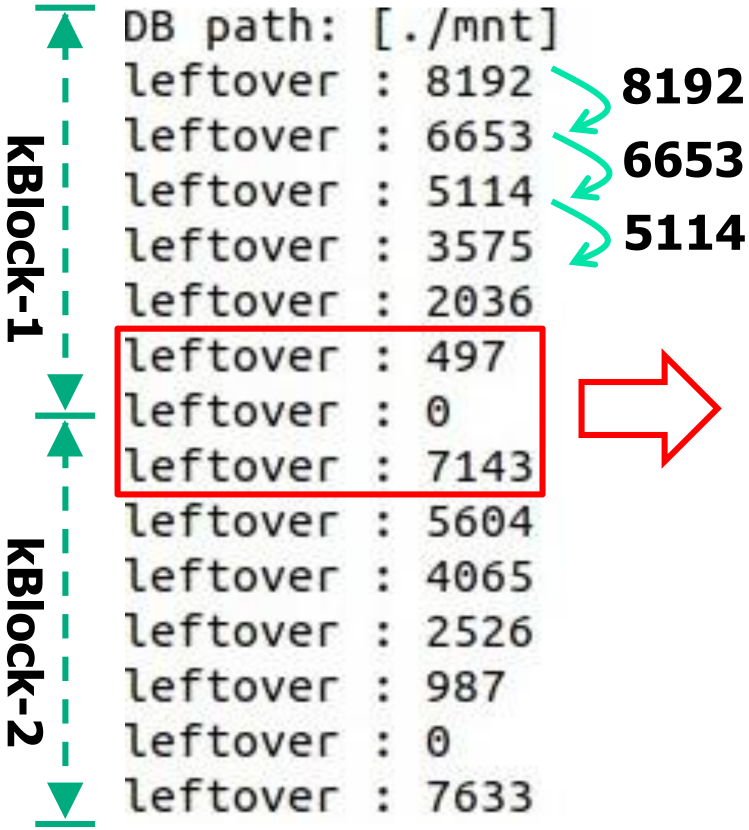


```
// is empty, we still want to iterate once to emit a single
// zero-length record
IOStatus s;
bool begin = true;
do {
  const int64_t leftover = kBlockSize - block_offset_;

  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

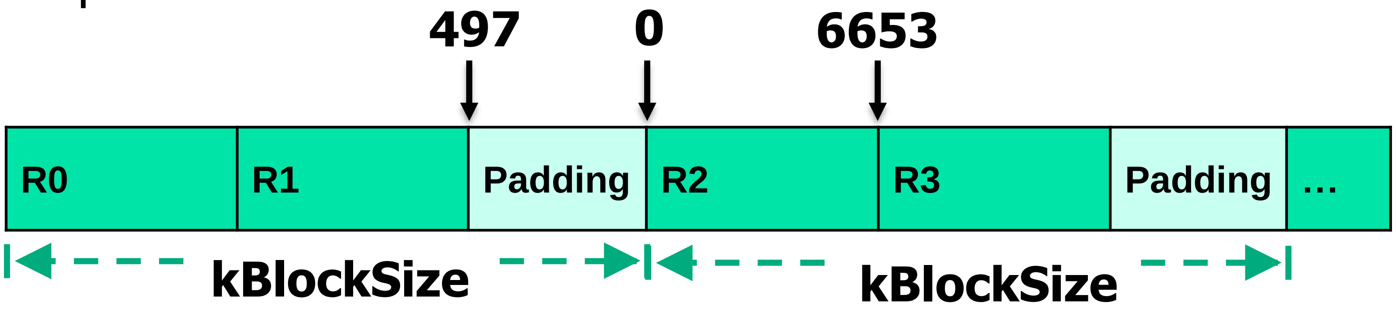
Fragmentation!

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

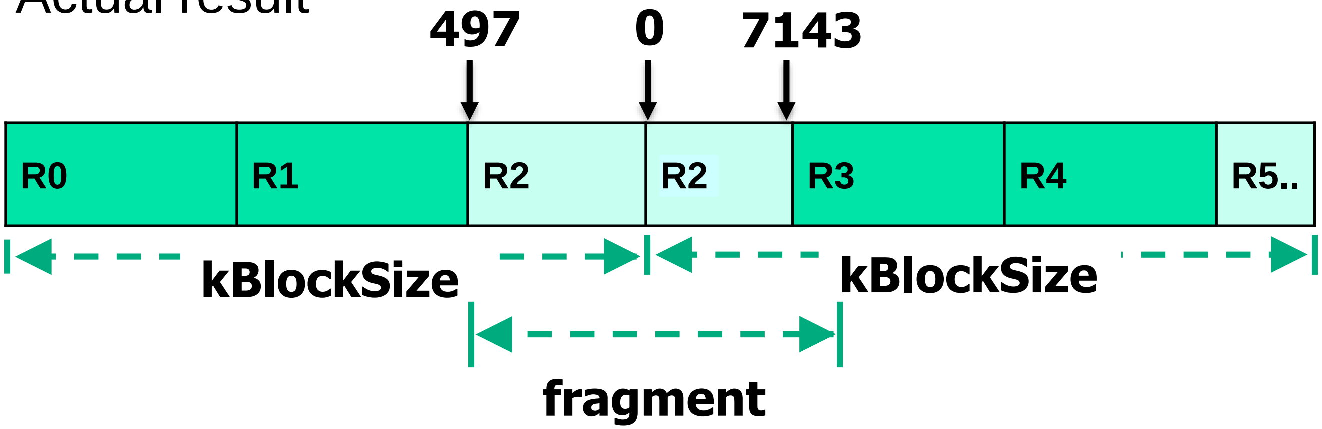
Where is padding??

RocksDB Festival : kBlockSize

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

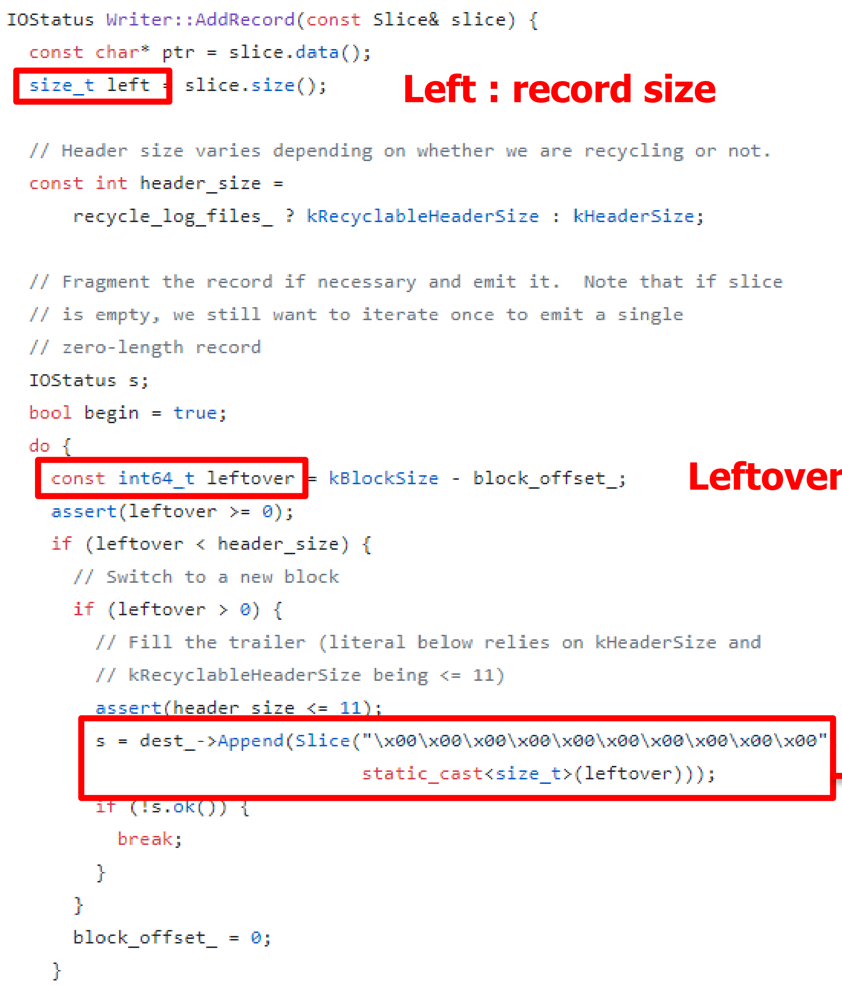


  - ✓ Actual result



RocksDB Festival : kBlockSize

- Source code analysis
  - ✓ AddRecord => log file writer



```
IOStatus Writer::AddRecord(const Slice& slice) {
  const char* ptr = slice.data();
  size_t left = slice.size();

  // Header size varies depending on whether we are recycling or not.
  const int header_size =
      recycle_log_files_ ? kRecyclableHeaderSize : kHeaderSize;

  // Fragment the record if necessary and emit it.  Note that if slice
  // 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_;
    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)));
        if (!s.ok()) {
          break;
        }
      }
      block_offset_ = 0;
    }
```

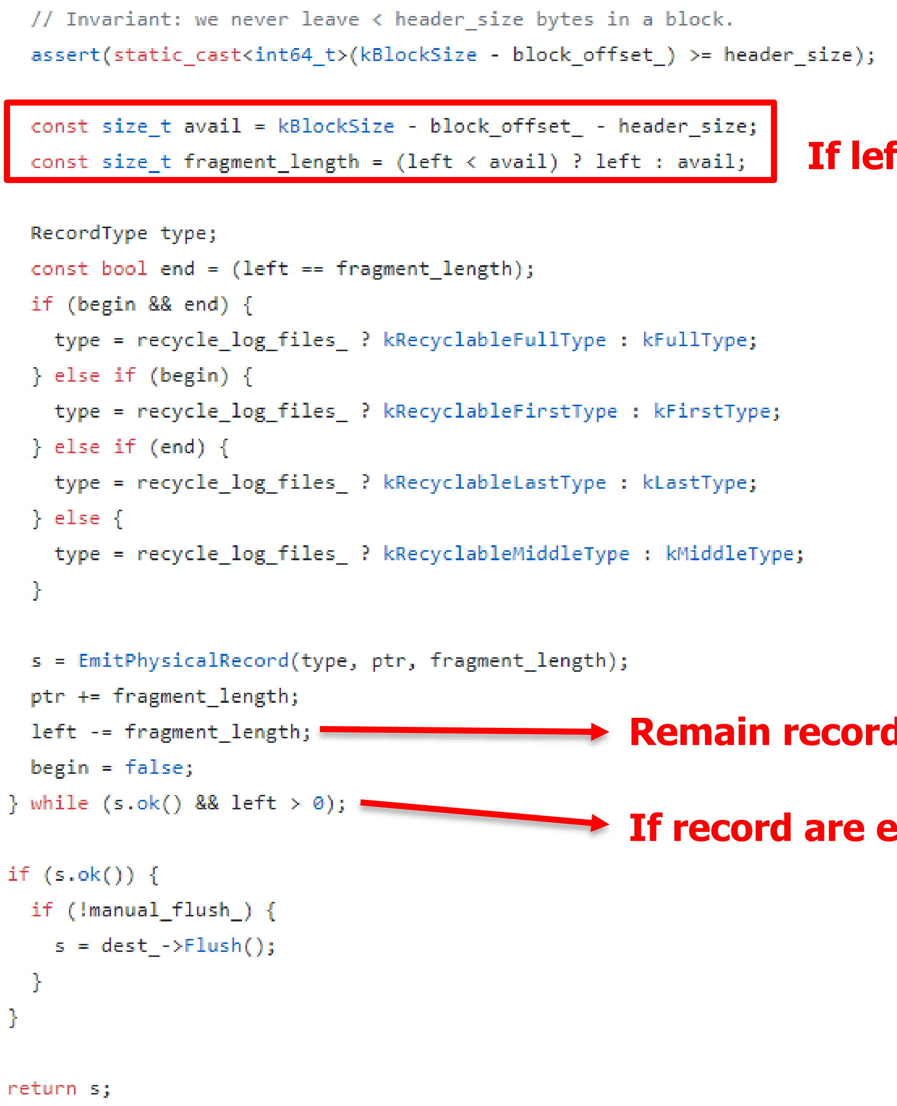
Left : record size

Leftover : empty space of kBlock

If leftover smaller than headersize, padding insert in kBlock Switch to next kBlock

RocksDB Festival : kBlockSize

- Source code analysis



```
  // Invariant: we never leave < header_size bytes in a block.
  assert(static_cast<int64_t>(kBlockSize - block_offset_) >= header_size);

  const size_t avail = kBlockSize - block_offset_ - header_size;
  const size_t fragment_length = (left < avail) ? left : avail;

  RecordType type;
  const bool end = (left == fragment_length);
  if (begin && end) {
    type = recycle_log_files_ ? kRecyclableFullType : kFullType;
  } else if (begin) {
    type = recycle_log_files_ ? kRecyclableFirstType : kFirstType;
  } else if (end) {
    type = recycle_log_files_ ? kRecyclableLastType : kLastType;
  } else {
    type = recycle_log_files_ ? kRecyclableMiddleType : kMiddleType;
  }

  s = EmitPhysicalRecord(type, ptr, fragment_length);
  ptr += fragment_length;
  left -= fragment_length;
  begin = false;
} while (s.ok() && left > 0);

if (s.ok()) {
  if (!manual_flush_) {
    s = dest_->Flush();
  }
}

return s;
```

If left bigger than avail, it is fragmented by avail

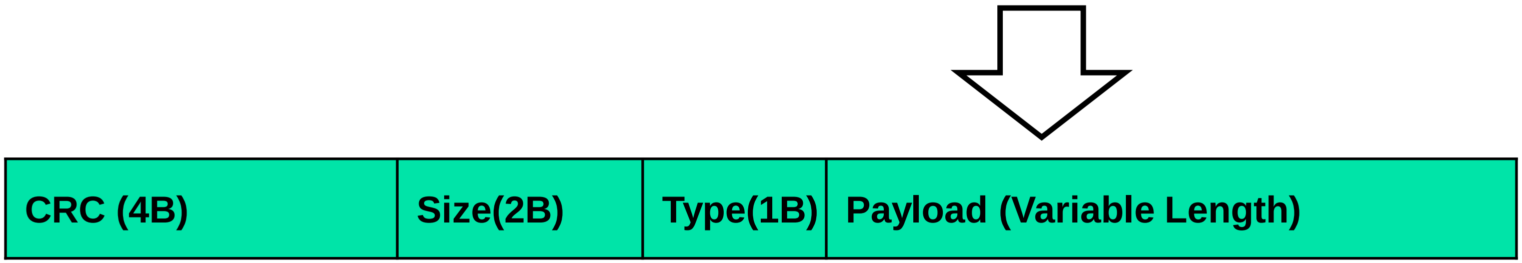
Padding is created only when the header cannot be stored

Remain record size

If record are entirely stored in kBlock, the loop ends

RocksDB Festival : key / value size

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



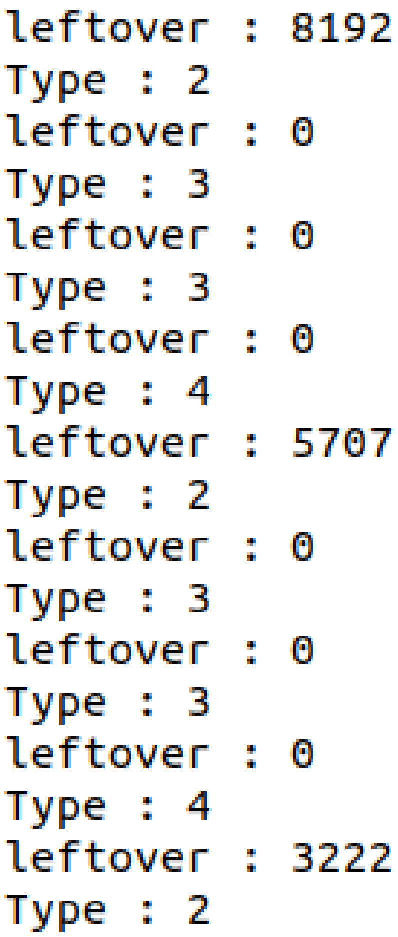
CRC (4B)	Size(2B)	Type(1B)	Payload (Variable Length)

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

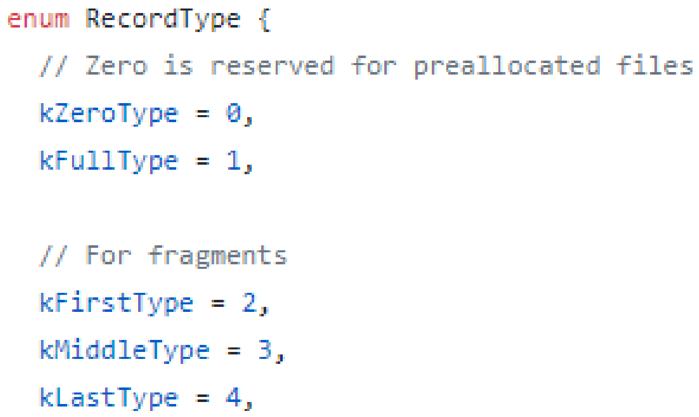
Discussion

Appendix : Type

- RecordType - 27061 bytes Records



```
leftover : 8192
Type : 2
leftover : 0
Type : 3
leftover : 0
Type : 3
leftover : 0
Type : 4
leftover : 5707
Type : 2
leftover : 0
Type : 3
leftover : 0
Type : 3
leftover : 0
Type : 4
leftover : 3222
Type : 2
```



```
enum RecordType {
  // Zero is reserved for preallocated files
  kZeroType = 0,
  kFullType = 1,

  // For fragments
  kFirstType = 2,
  kMiddleType = 3,
  kLastType = 4,
```

Block0	Record0 (0) kFirstType(2) 8192 bytes	
Block1	Record0 (1) kMiddleType(3) 8192 bytes	
Block2	Record0 (2) kMiddleType(3) 8192 bytes	
Block3	Record0 (3) kLastType(4) 2485 bytes	Record1 (0)
Block4	Record1 (1) kMiddleType(3) …	