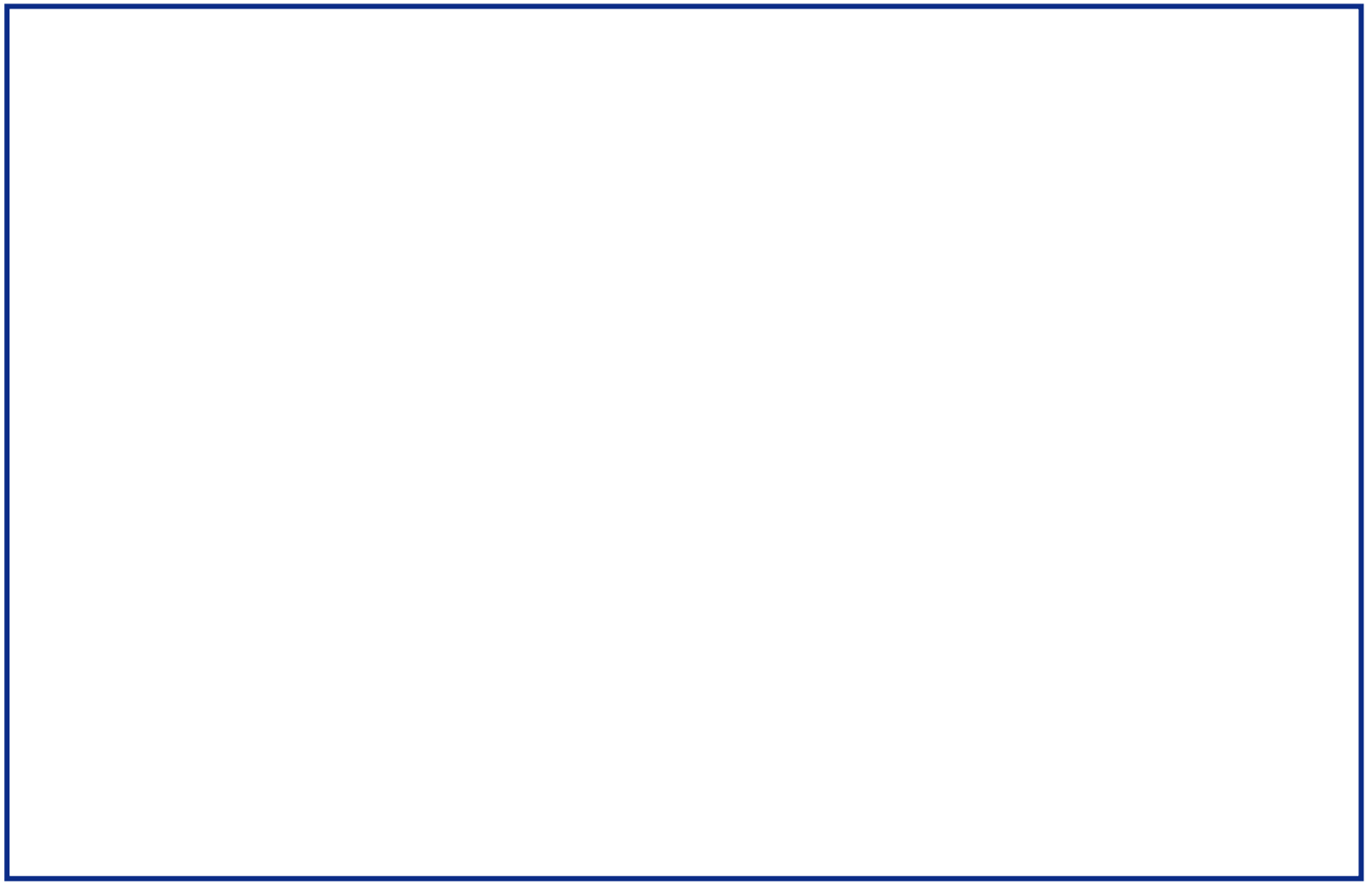
Memtable Bench

Contents:

About the results of the last bench:

- fillseq & fillrandom
- write_buffer_size
- max_file_size
- Summarize

Code flow in memtable.cc (incomplete)



About the results of the last bench

- These are the key research points

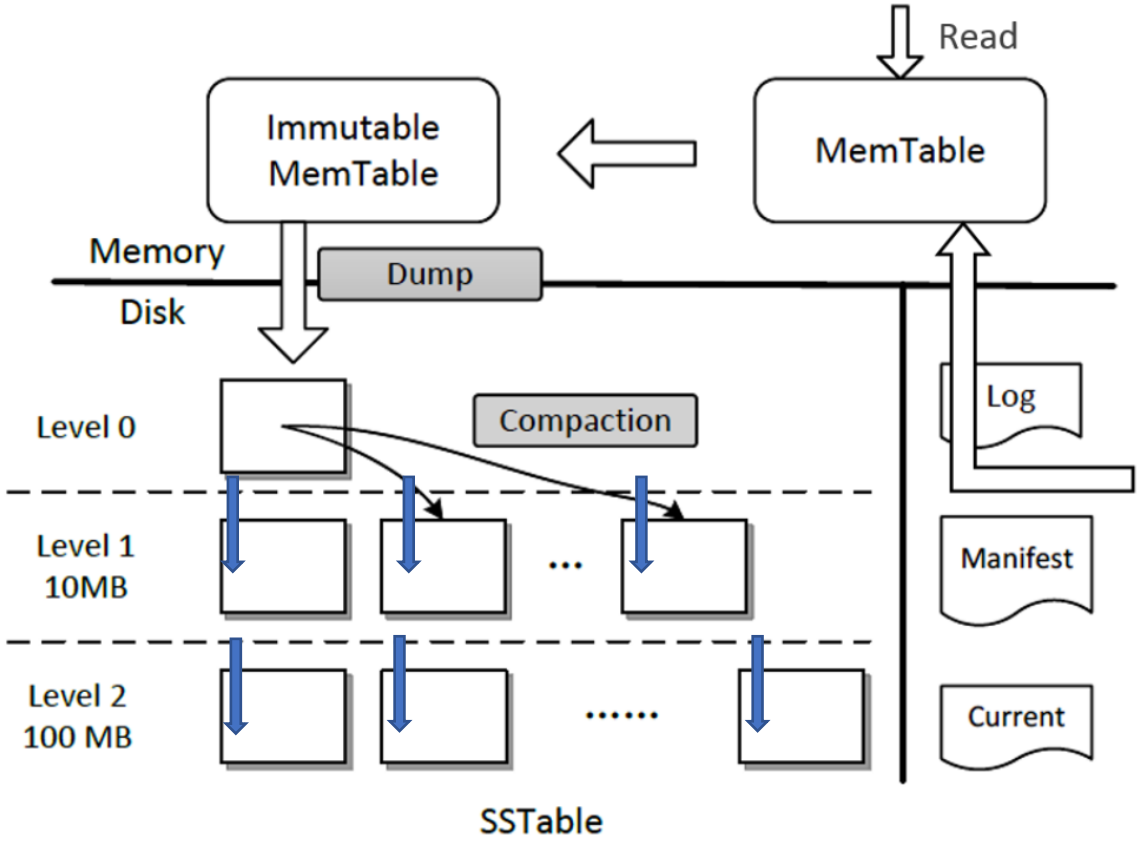
Why is the latency of fillrandom slower than that of fillseq?

Why write_buffer_size gets bigger, the fillrandom latency gets better?

Why max_file_size gets bigger, the fillrandom latency gets better?

About the results of the last bench

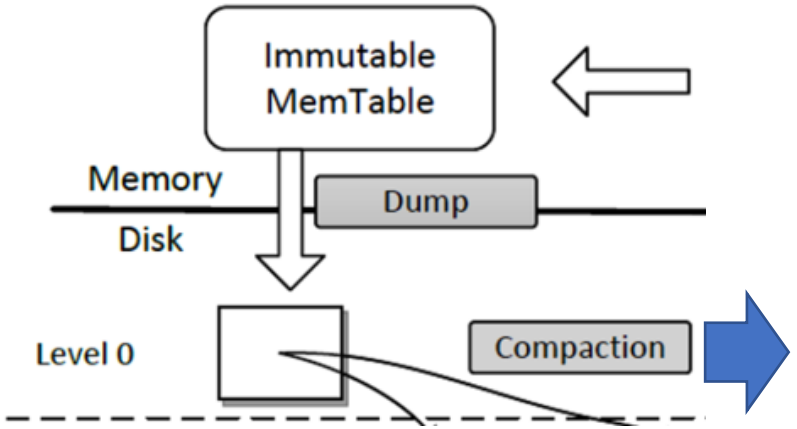
- fillseq & fillrandom



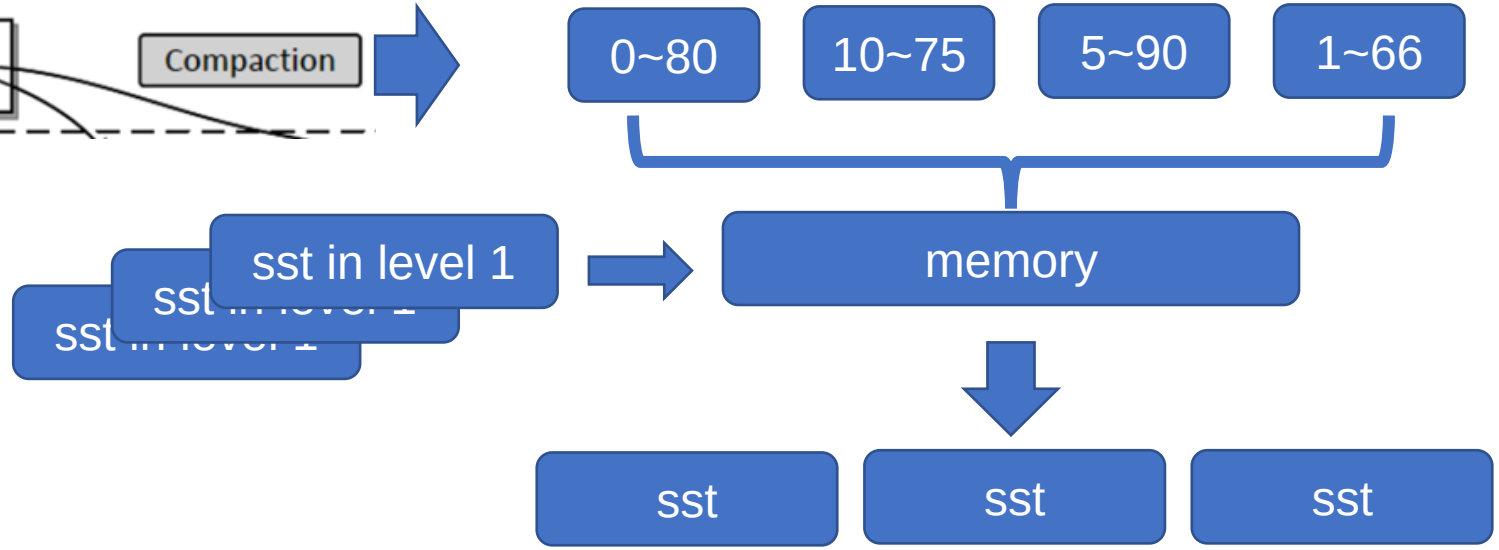
k=0,k=1,k=2,k=3,k=4……

About the results of the last bench

- fillseq & fillrandom

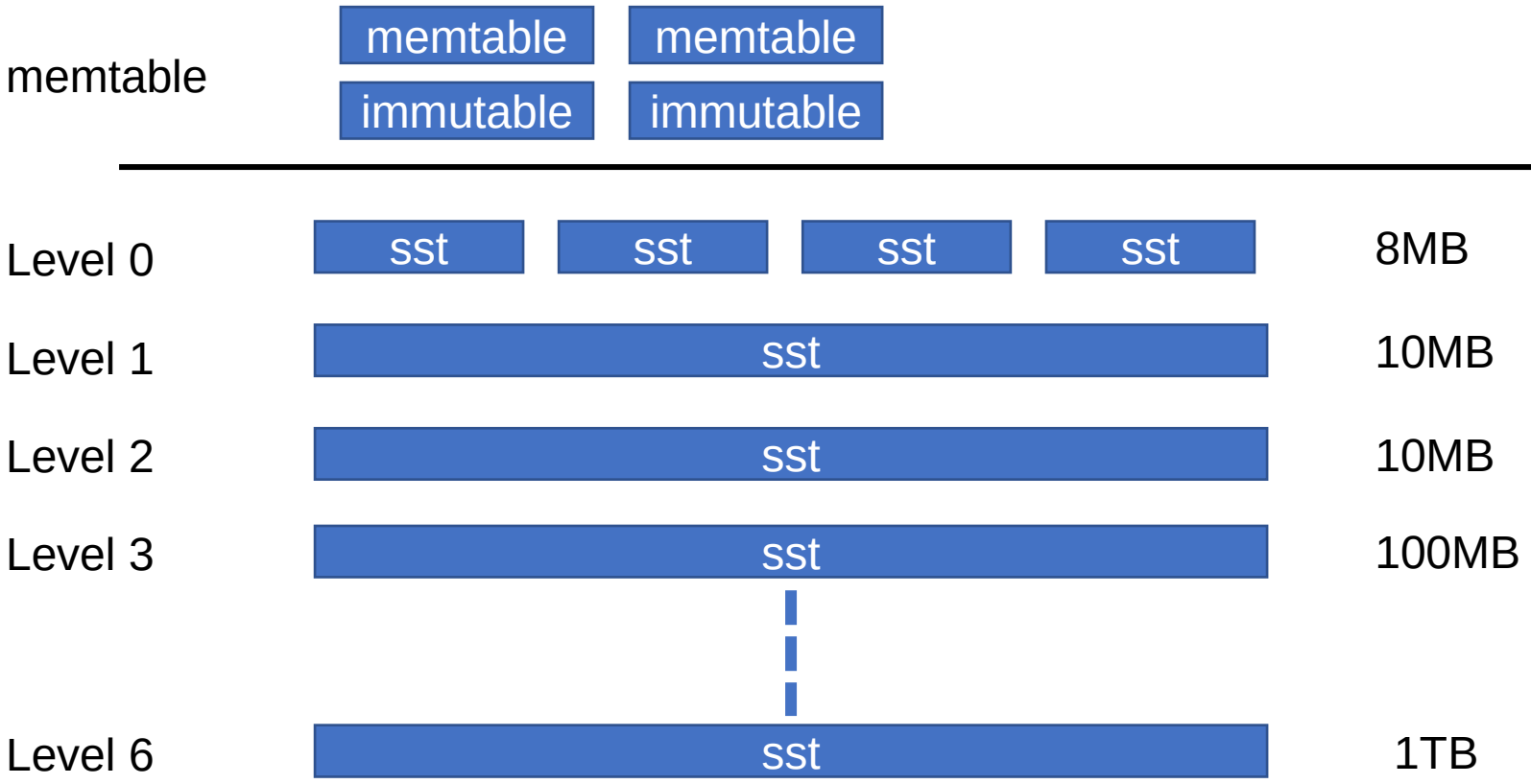


Latency of fillrandom is slower than that of fillseq



About the results of the last bench

- Why write_buffer_size gets bigger, the fillrandom latency gets better?



About the results of the last bench

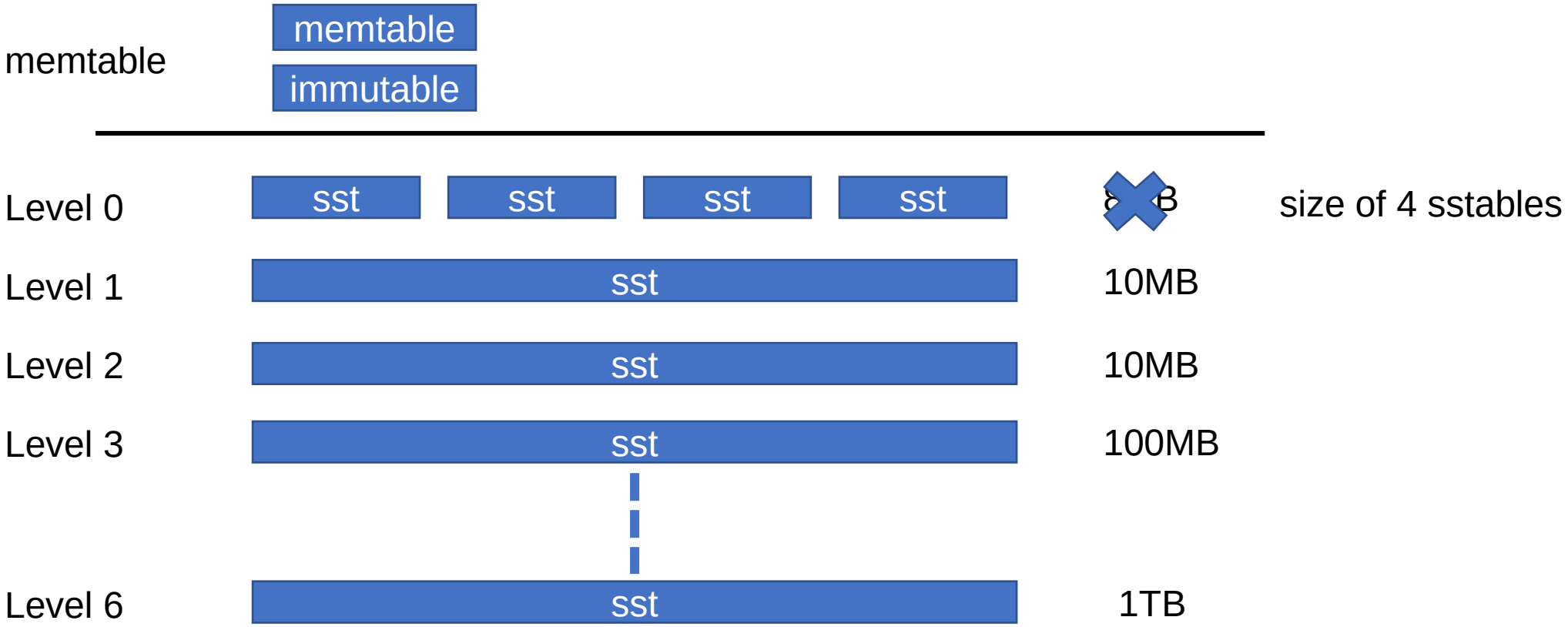
- Why write_buffer_size gets bigger, the fillrandom latency gets better?

```
20  namespace leveldb {
21
22  // Grouping of constants.  We may want to make some of these
23  // parameters set via options.
24  namespace config {
25  static const int kNumLevels = 7;
26
27  // Level-0 compaction is started when we hit this many files.
28  static const int kL0_CompactionTrigger = 4;
29
30  // Soft limit on number of level-0 files.  We slow down writes at this point.
31  static const int kL0_SlowdownWritesTrigger = 8;
32
33  // Maximum number of level-0 files.  We stop writes at this point.
34  static const int kL0_StopWritesTrigger = 12;
35
36  // Maximum level to which a new compacted memtable is pushed if it
37  // does not create overlap.  We try to push to level 2 to avoid the
38  // relatively expensive level 0=>1 compactions and to avoid some
39  // expensive manifest file operations.  We do not push all the way to
40  // the largest level since that can generate a lot of wasted disk
41  // space if the same key space is being repeatedly overwritten.
42  static const int kMaxMemCompactLevel = 2;
43
44  // Approximate gap in bytes between samples of data read during iteration.
45  static const int kReadBytesPeriod = 1048576;
46
47  }  // namespace config
```

dbformat.h

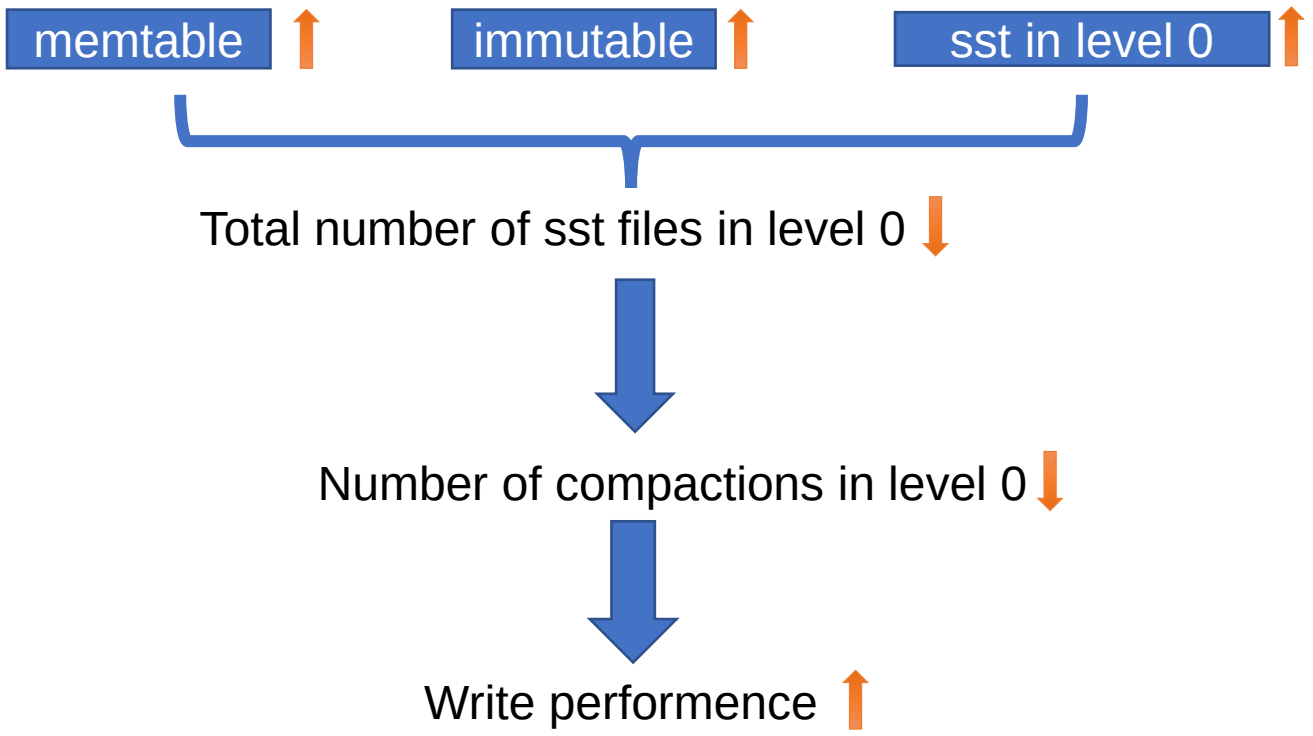
About the results of the last bench

- Why write_buffer_size gets bigger, the fillrandom latency gets better?



About the results of the last bench

- Why write_buffer_size gets bigger, the fillrandom latency gets better?



But what is the demerit?

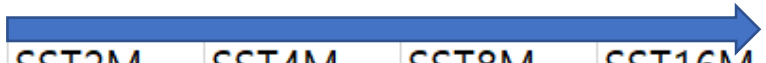
write_buffer_size ↑

Speed of opening database ↓

About the results of the last bench

- Why max_file_size gets bigger, the fillrandom latency gets better?

max_file_size = max size of sstable (not only for memtable)

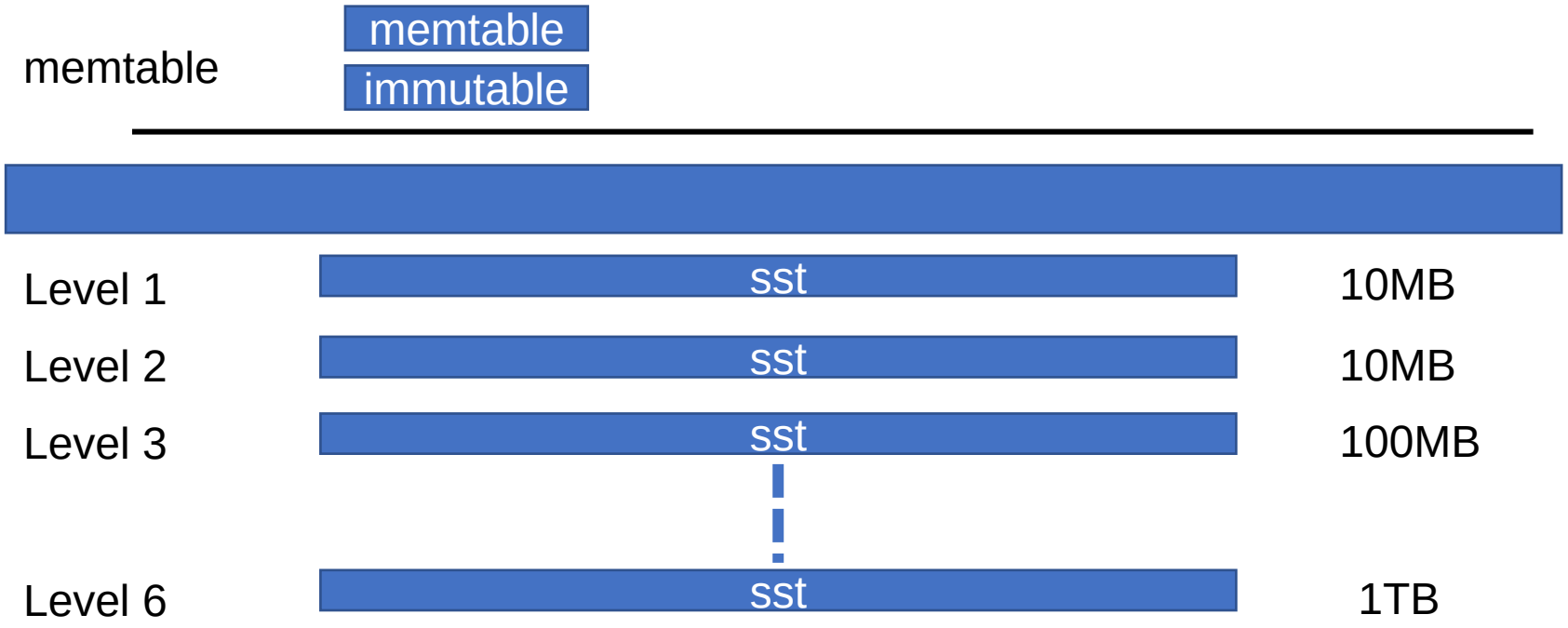


	SST2M	SST4M	SST8M	SST16M	SST32M
Mem2M	10.208	9.746	8.956	9.323	9.435
Mem4M	7.934	7.225	6.243	6.383	7.632
Mem8M	6.293	5.409	4.865	4.69	4.578
Mem16M	4.443	3.916	3.863	3.56	3.642
Mem32M	3.828	3.177	3.174	3.142	3.906

About the results of the last bench

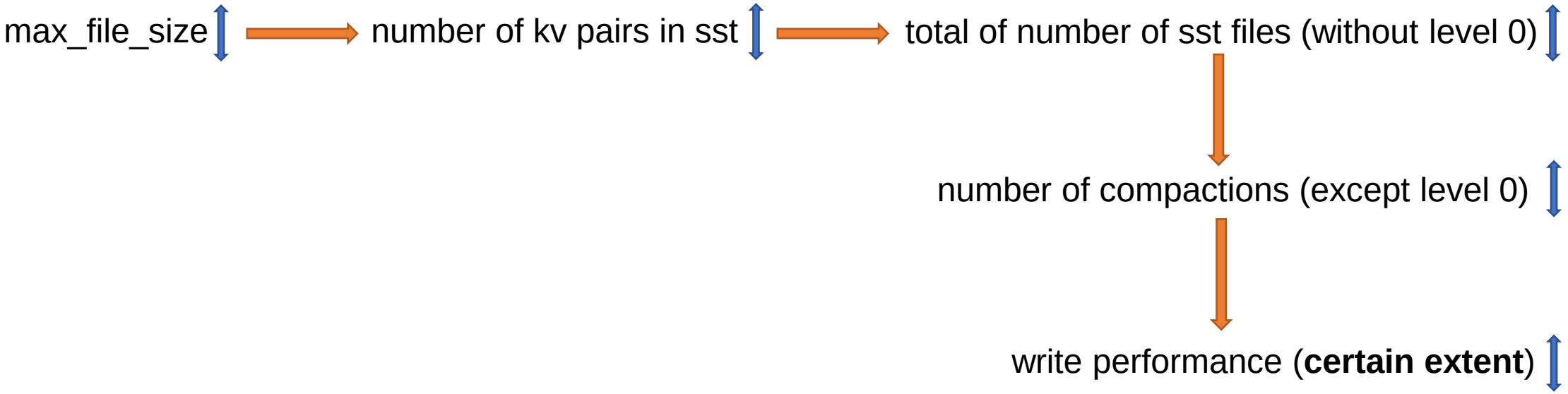
- Why max_file_size gets bigger, the fillrandom latency gets better?

max_file_size = max size of sstable (not only for memtable)



About the results of the last bench

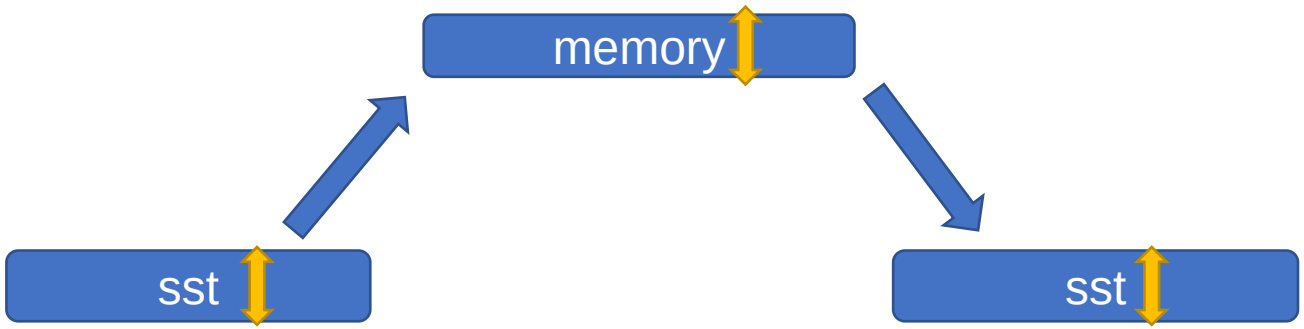
- Why max_file_size gets bigger, the fillrandom latency gets better?



About the results of the last bench

- Why max_file_size gets bigger, the fillrandom latency gets better?

certain extent ? **write/read amplification when compaction**

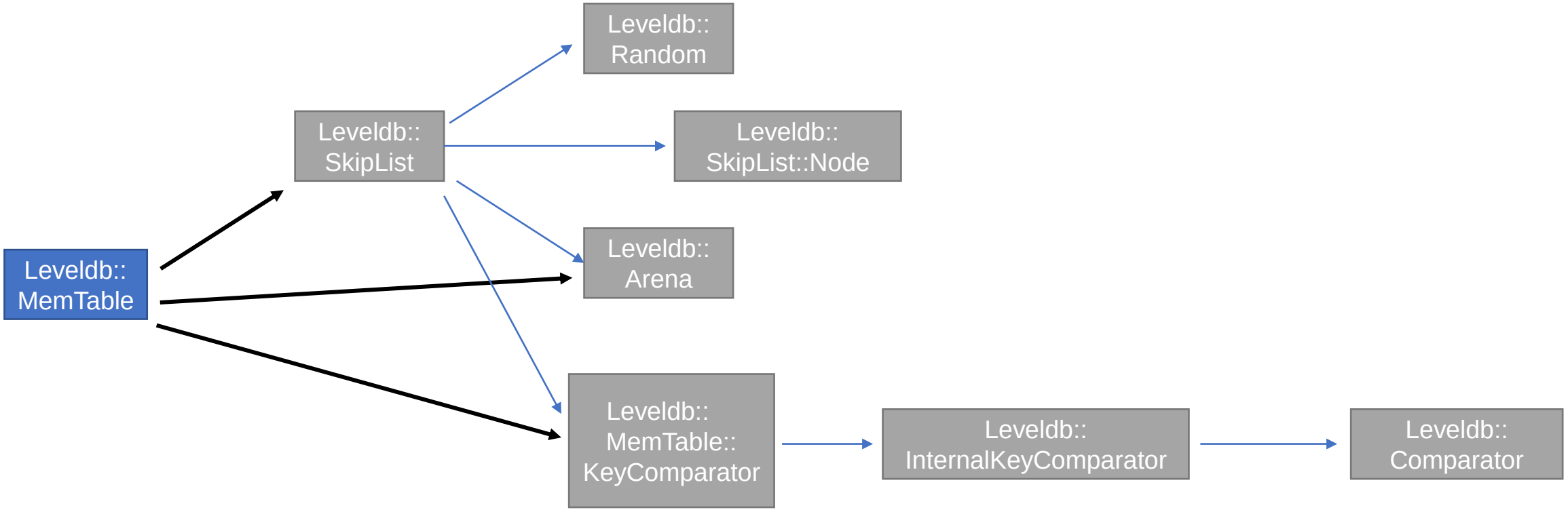


Summarize the bench results

- Increasing the write buffer size can improve the write performance, but also consider the time-consuming restart of DB.

- Max file size should be determined according to the size of KV. Although it will improve the write performance, it will also cause amplification.

Code flow in memtable.cc



Q&A

감사합니다

Thank you!