Memtable bench

Made by Jeyeon Lee

E-Mail: 2reenact@dankook.ac.kr

Contents:

1.실험환경

- LevelDB: version 1.23
- CPU: 4 * Intel(R) Core(TM) i5-4690 CPU @ 3.50GHz
- CPUCache: 6144 KB
- Keys: 16 bytes each
- Values: 100 bytes each (100 bytes after compression)
- Entries: 5000000
- Reads: 1000000
- RawSize: 553.1 MB (estimated)
- FileSize: 553.1 MB (estimated)+
- Storage: (SATA) Samsung 860pro(500G)

2. 실험과정

Topics / Benchmarks / Options

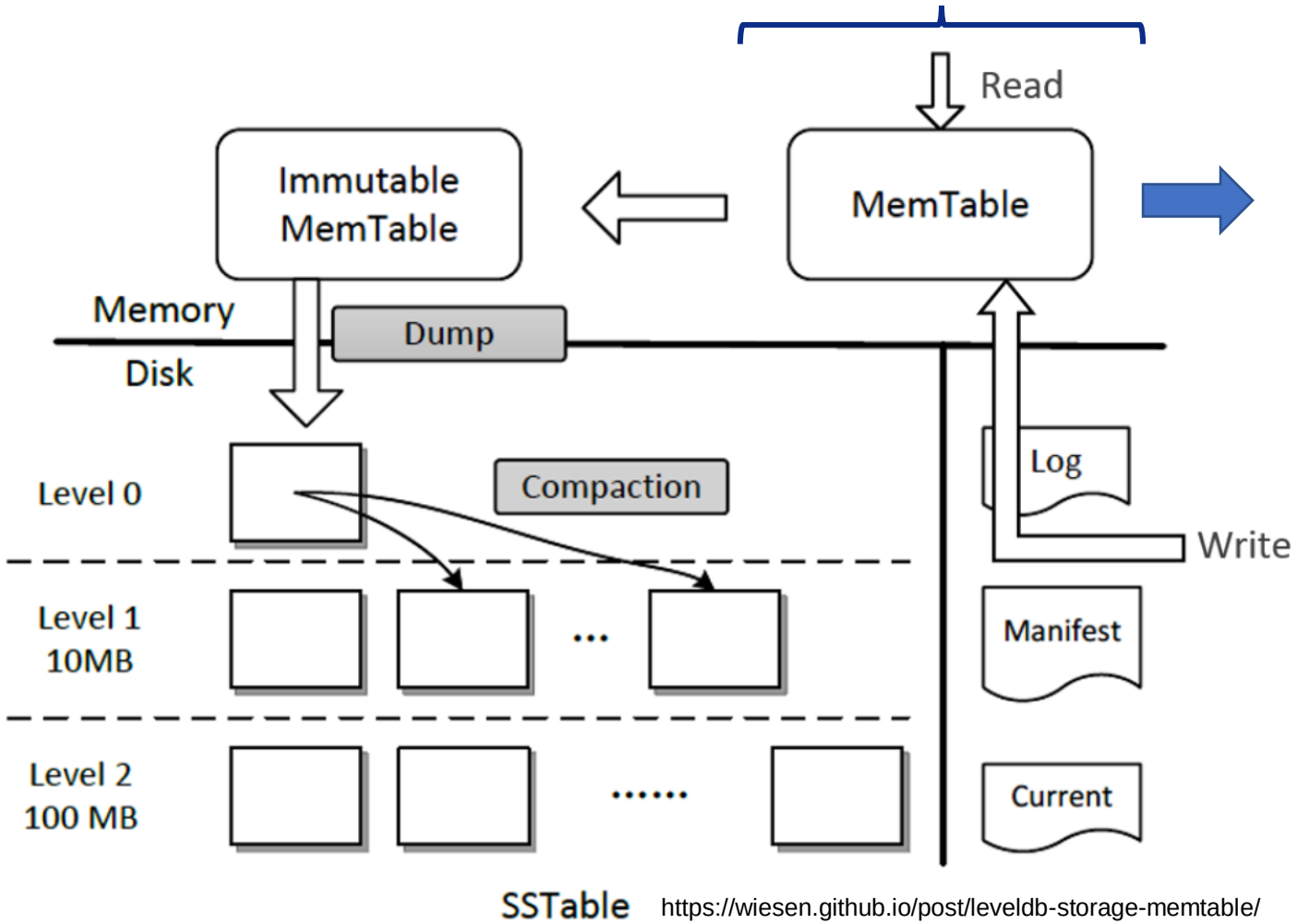
No	Topic	Benchmarks	Options	Result
1	WAL/Manifest	--disable_wal --wal_bytes_per_sync	fillseq/random readrandom	PPT
2	Memtable	--write_buffer_size --max_file_size	fillseq/random readrandom	PPT
3	Compaction	--base_background_compactions --compaction_style	fillseq/random readrandom	PPT
4	SSTable	--write_buffer_size --max_file_size --block_size	fillseq/random readseq/random seekrandom	PPT
5	Bloom Filter	--bloom_bits	readhot/random seekrandom	PPT
6	Cache	--cache_size --block_size	readhot/random seekrandom	PPT

https://github.com/DKU-StarLab/leveldb-study/blob/main/benchmarks/README.md

Benchmarks	Options
--write_buffer_size --max_file_size	fillseq/random readrandom

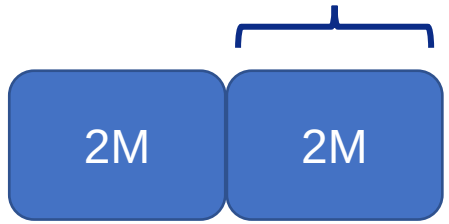
2. 실험과정

write_buffer_size = 4MB(default)



https://wiesen.github.io/post/leveldb-storage-memtable/

max_file_size = 2MB(default)



2. 실험과정

write_buffer_size	max_file_size
2M	2M(default)
4M(default)	4M
8M	8M
16M	16M
32M	32M

3. 실험결과

fillseq & fillrandom

max_file_size →

	SST2M	SST4M	SST8M	SST16M	SST32M
Mem2M	2.192	2.162	2.187	2.172	2.152
Mem4M	2.153	2.192	1.951	1.958	1.983
Mem8M	2.022	2.029	2.001	1.958	2.365
Mem16M	2.129	1.971	2.011	1.961	2.015
Mem32M	2.085	2.219	2.098	2.129	2.161

	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

↓ write_buffer_size

3. 실험결과

fillseq & fillrandom

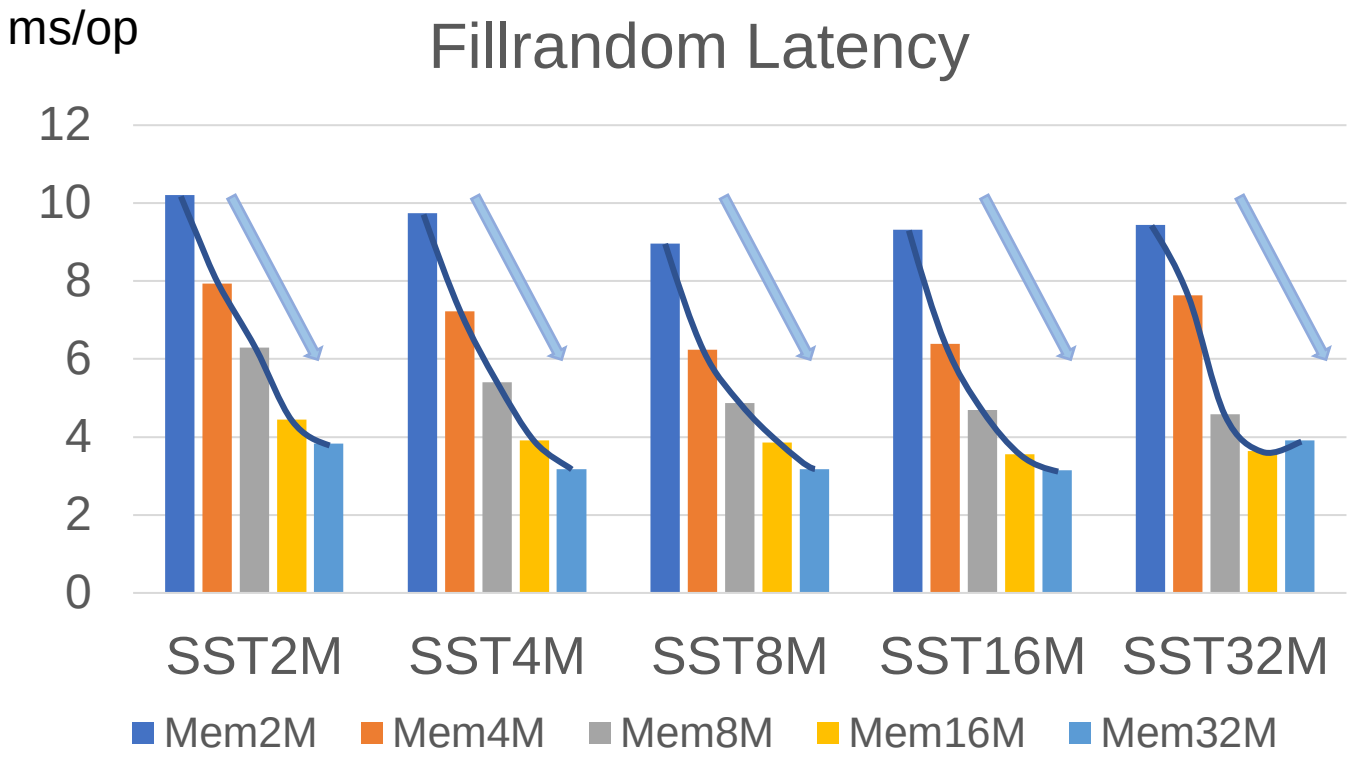
max_file_size

	SST2M	SST4M	SST8M	SST16M	SST32M
Mem2M	2.192	2.162	2.187	2.172	2.152
Mem4M	2.153	2.192	1.951	1.958	1.983
Mem8M	2.022	2.029	2.001	1.958	2.365
Mem16M	2.129	1.971	2.011	1.961	2.015
Mem32M	2.085	2.219	2.098	2.129	2.161

	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

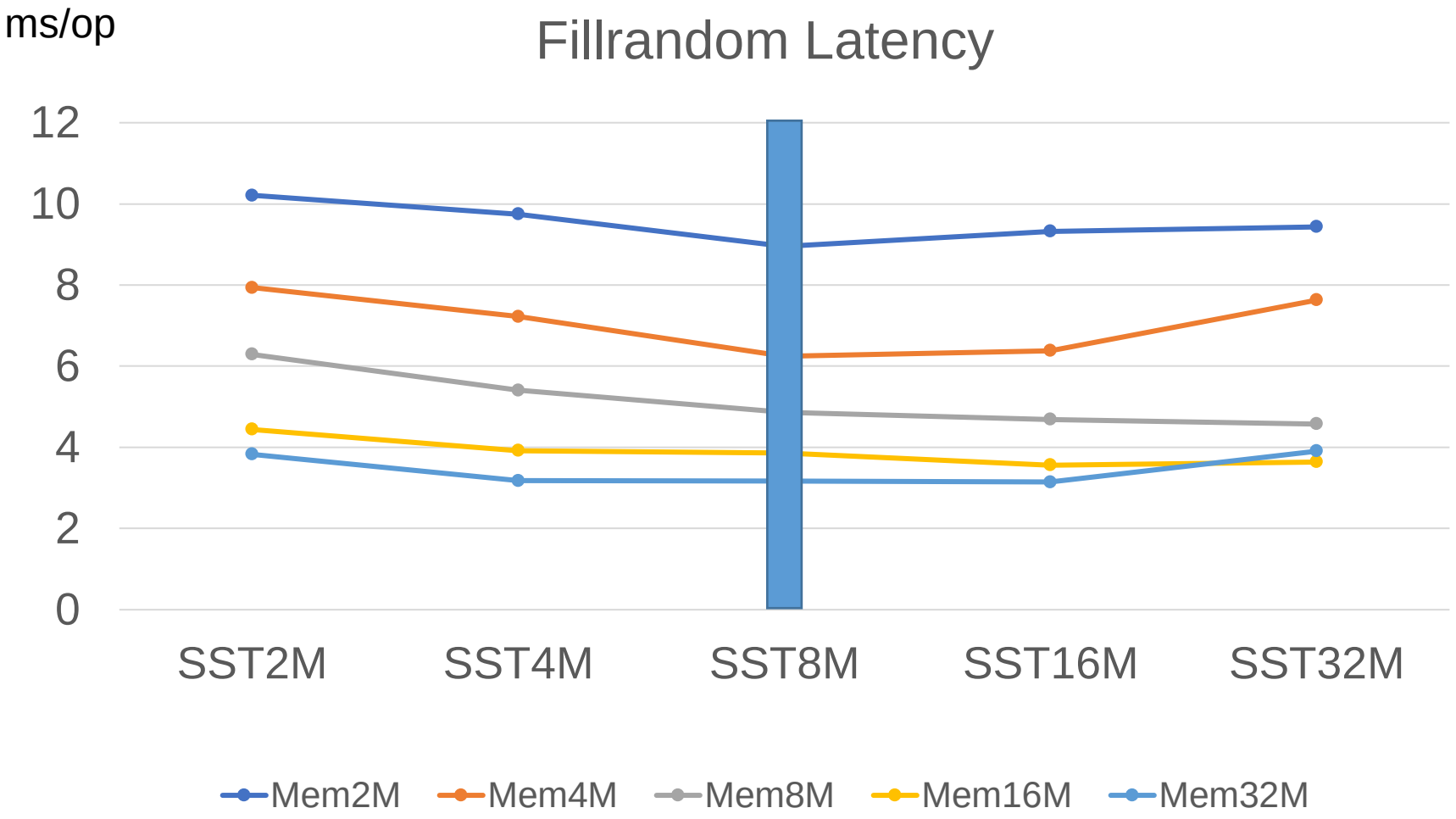
write_buffer_size

3. 실험결과



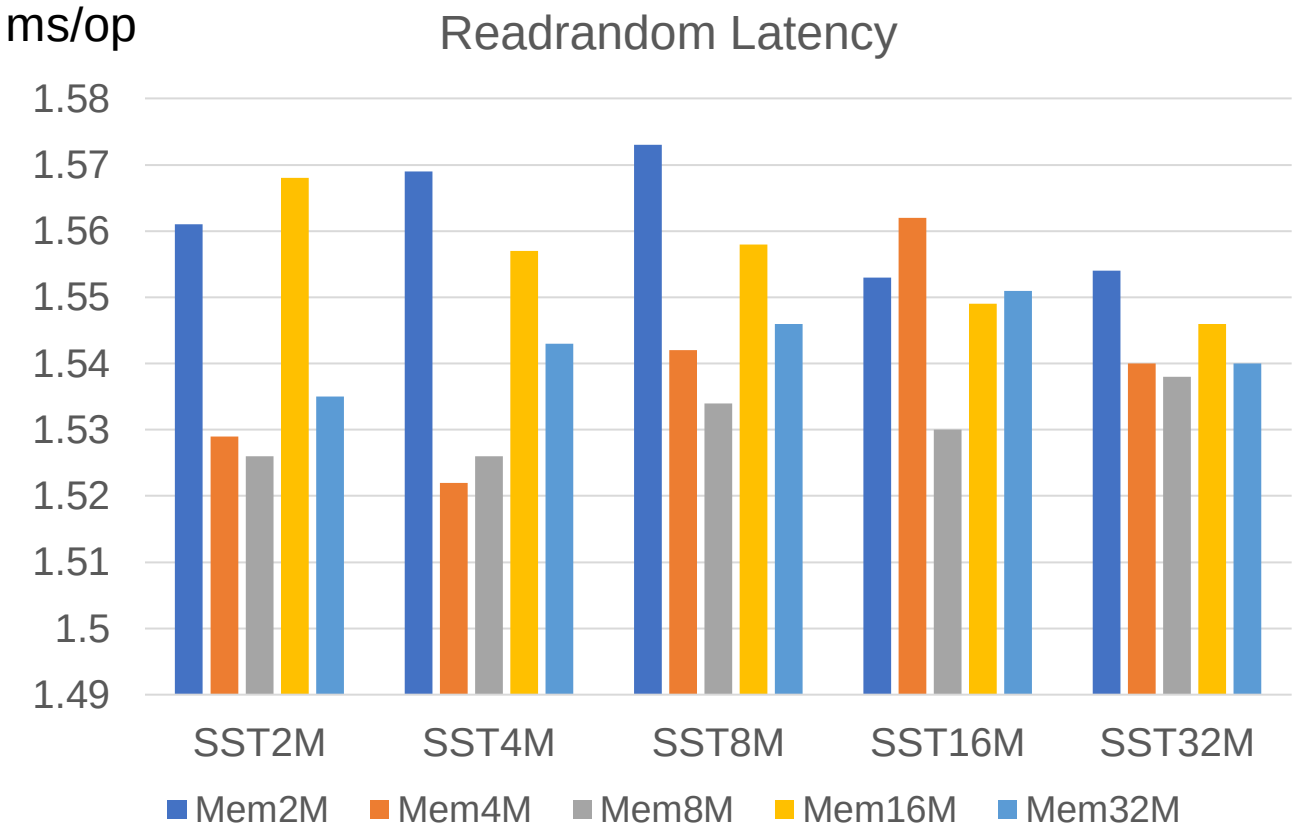
The larger the values of SST and mem, the smaller the decrease of latency

3. 실험결과


ms/op
Fillrandom Latency
12
10
8
6
4
2
0
SST2M
SST4M
SST8M
SST16M
SST32M
Mem2M
Mem4M
Mem8M
Mem16M
Mem32M

3. 실험결과

- Readrandom

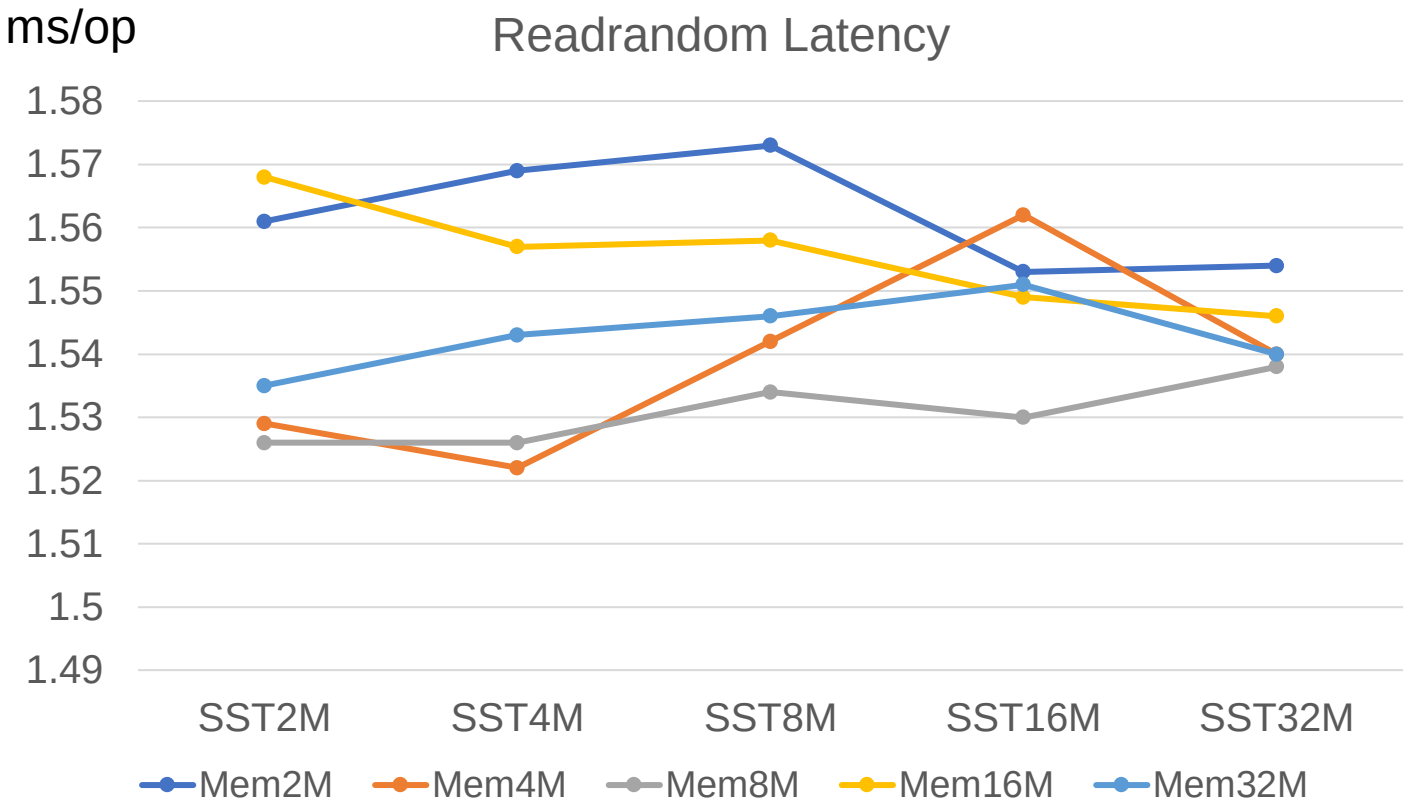


- fillseq

	SST2M	SST4M	SST8M	SST16M	SST32M
Mem2M	1.561	1.569	1.573	1.553	1.554
Mem4M	1.529	1.522	1.542	1.562	1.54
Mem8M	1.526	1.526	1.534	1.53	1.538
Mem16M	1.568	1.557	1.558	1.549	1.546
Mem32M	1.535	1.543	1.546	1.551	1.54

3. 실험결과

- Readrandom

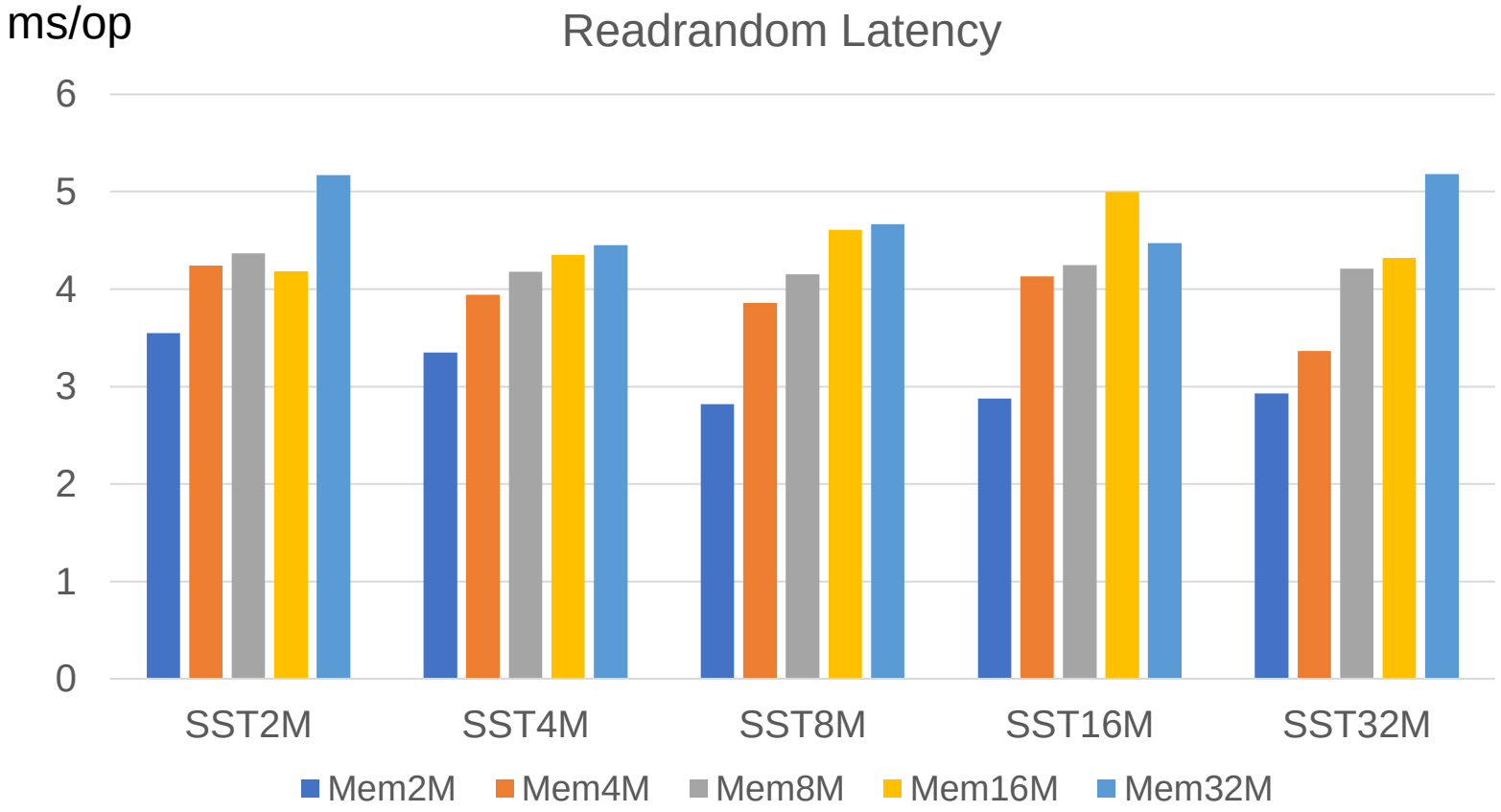


- fillseq

	SST2M	SST4M	SST8M	SST16M	SST32M
Mem2M	1.561	1.569	1.573	1.553	1.554
Mem4M	1.529	1.522	1.542	1.562	1.54
Mem8M	1.526	1.526	1.534	1.53	1.538
Mem16M	1.568	1.557	1.558	1.549	1.546
Mem32M	1.535	1.543	1.546	1.551	1.54

3. 실험결과

- Readrandom

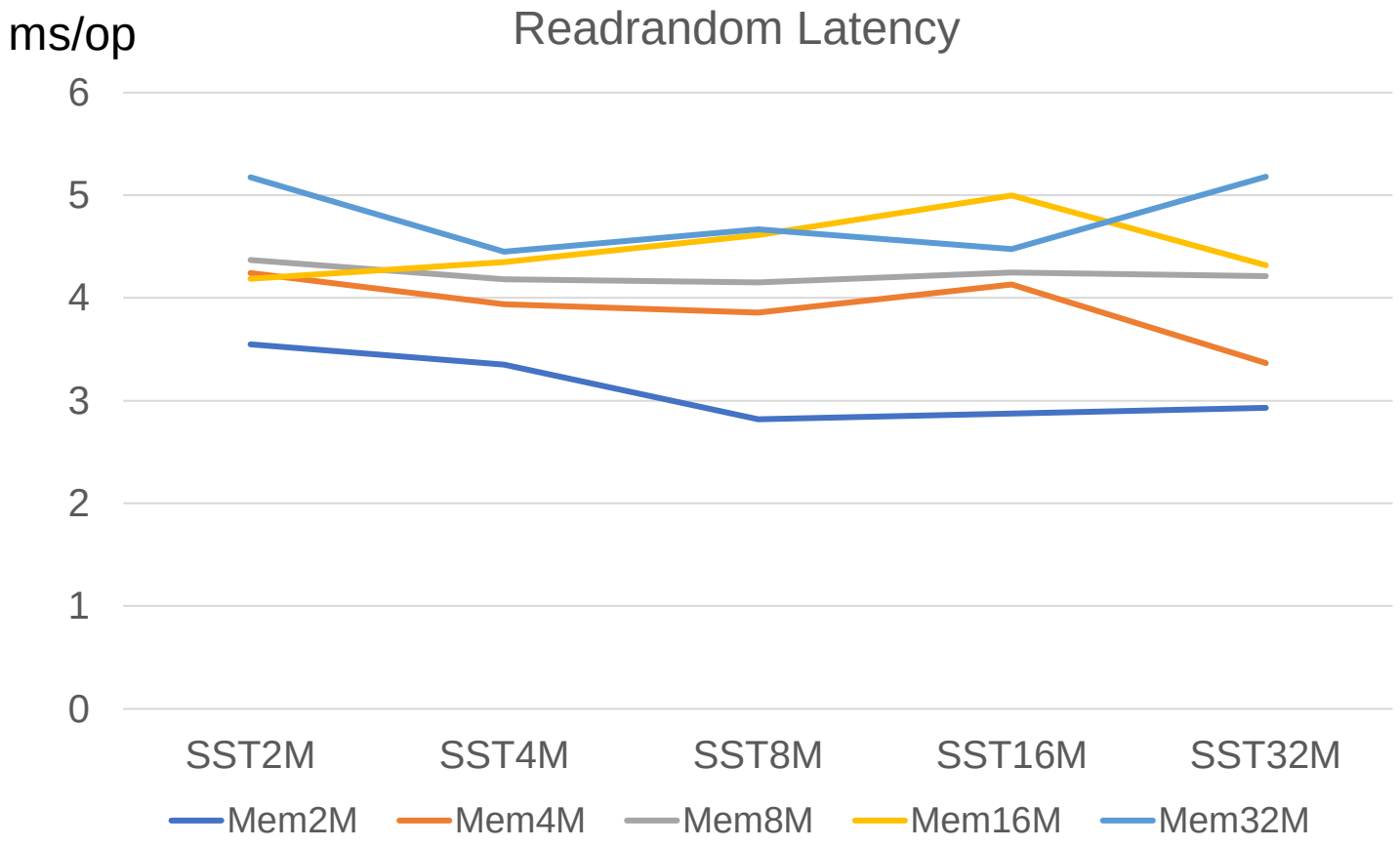


- fillrandom

	SST2M	SST4M	SST8M	SST16M	SST32M
Mem2M	3.548	3.35	2.82	2.875	2.932
Mem4M	4.24	3.94	3.856	4.131	3.365
Mem8M	4.369	4.181	4.152	4.247	4.211
Mem16M	4.186	4.35	4.611	4.999	4.319
Mem32M	5.172	4.452	4.667	4.473	5.18

3. 실험결과

- Readrandom



- fillrandom

	SST2M	SST4M	SST8M	SST16M	SST32M
Mem2M	3.548	3.35	2.82	2.875	2.932
Mem4M	4.24	3.94	3.856	4.131	3.365
Mem8M	4.369	4.181	4.152	4.247	4.211
Mem16M	4.186	4.35	4.611	4.999	4.319
Mem32M	5.172	4.452	4.667	4.473	5.18

3. 실험결과

Readrandom Latency
(fillseq db load)

	SST2M	SST4M	SST8M	SST16M	SST32M
Mem2M	1.561	1.569	1.573	1.553	1.554
Mem4M	1.529	1.522	1.542	1.562	1.54
Mem8M	1.526	1.526	1.534	1.53	1.538
Mem16M	1.568	1.557	1.558	1.549	1.546
Mem32M	1.535	1.543	1.546	1.551	1.54

Readrandom Latency
(fillrandom db load)

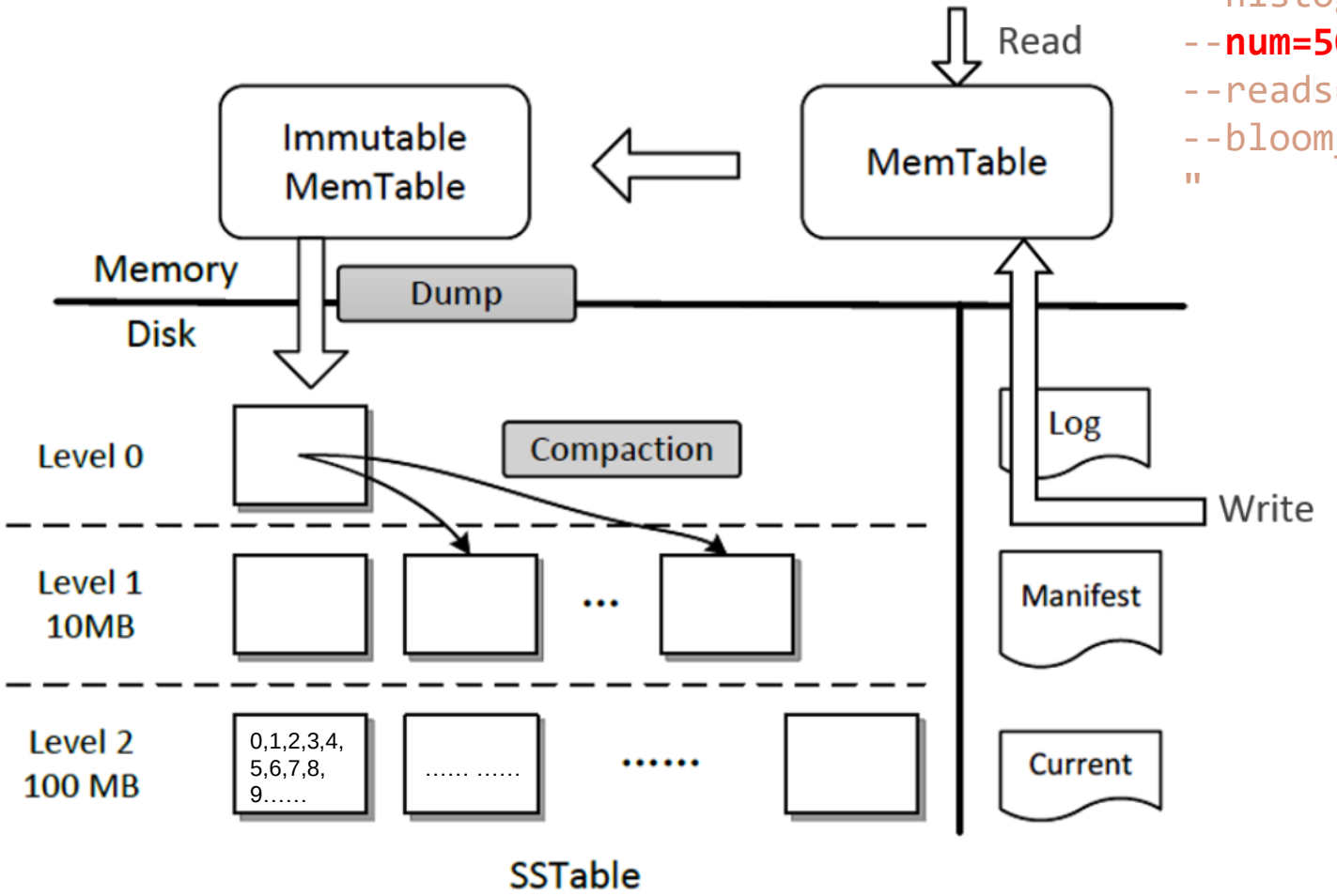
	SST2M	SST4M	SST8M	SST16M	SST32M
Mem2M	3.548	3.35	2.82	2.875	2.932
Mem4M	4.24	3.94	3.856	4.131	3.365
Mem8M	4.369	4.181	4.152	4.247	4.211
Mem16M	4.186	4.35	4.611	4.999	4.319
Mem32M	5.172	4.452	4.667	4.473	5.18

Both are readrandom. Why is fillrandom slower than fillseq?

3. 실험결과

fillseq

```
"./db_bench \
--use_existing_db=0 \
--compression_ratio=1 \
--comparisons=1 \
--benchmarks="fillseq" \
--histogram=1 \
--num=500000 \
--reads=100000 \
--bloom_bits=0 \
"
```

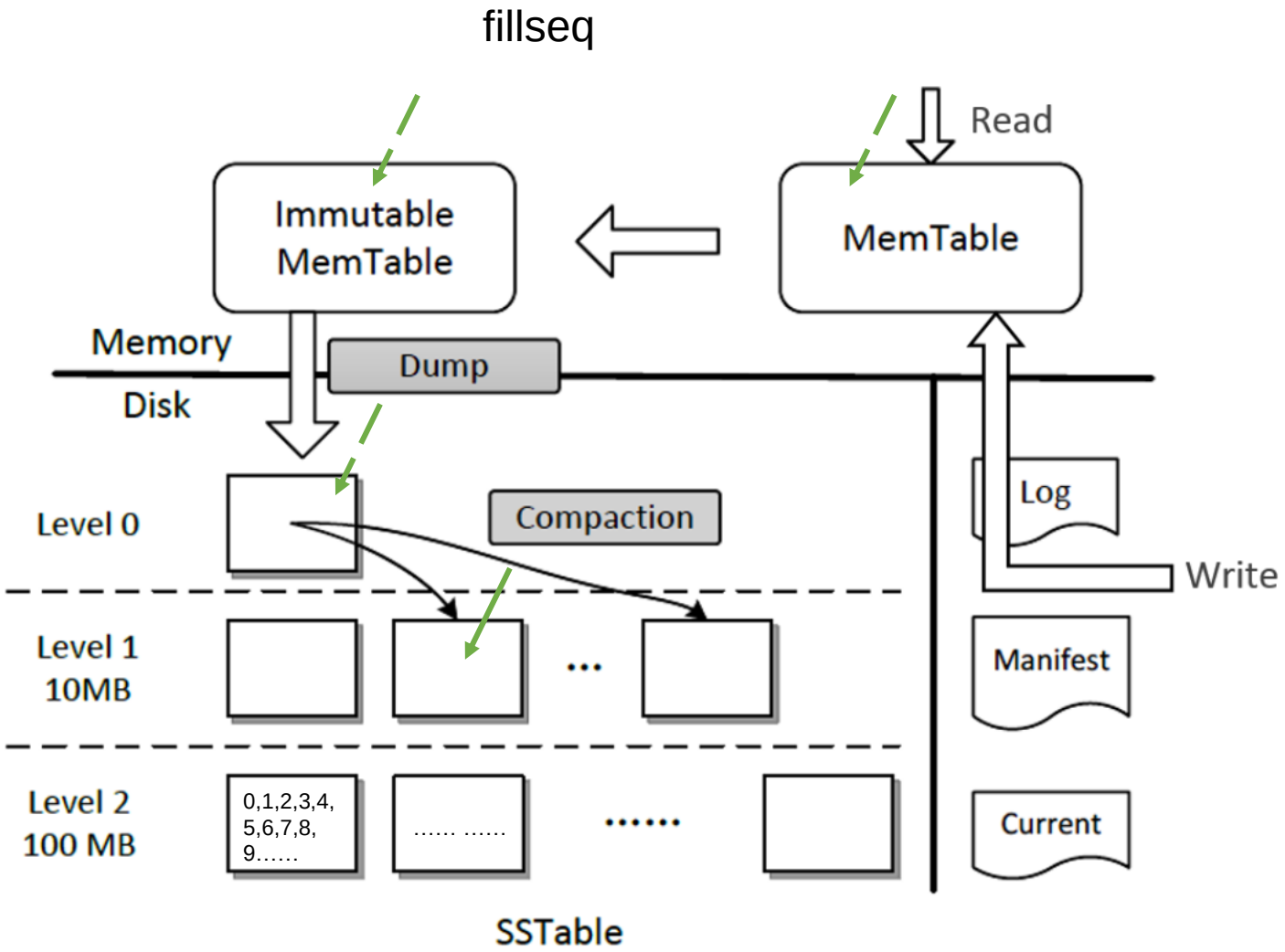


k:10001 v:……
k:10002 v:……
k:10003 v:……
k:10004 v:……
k:…

3. 실험결과

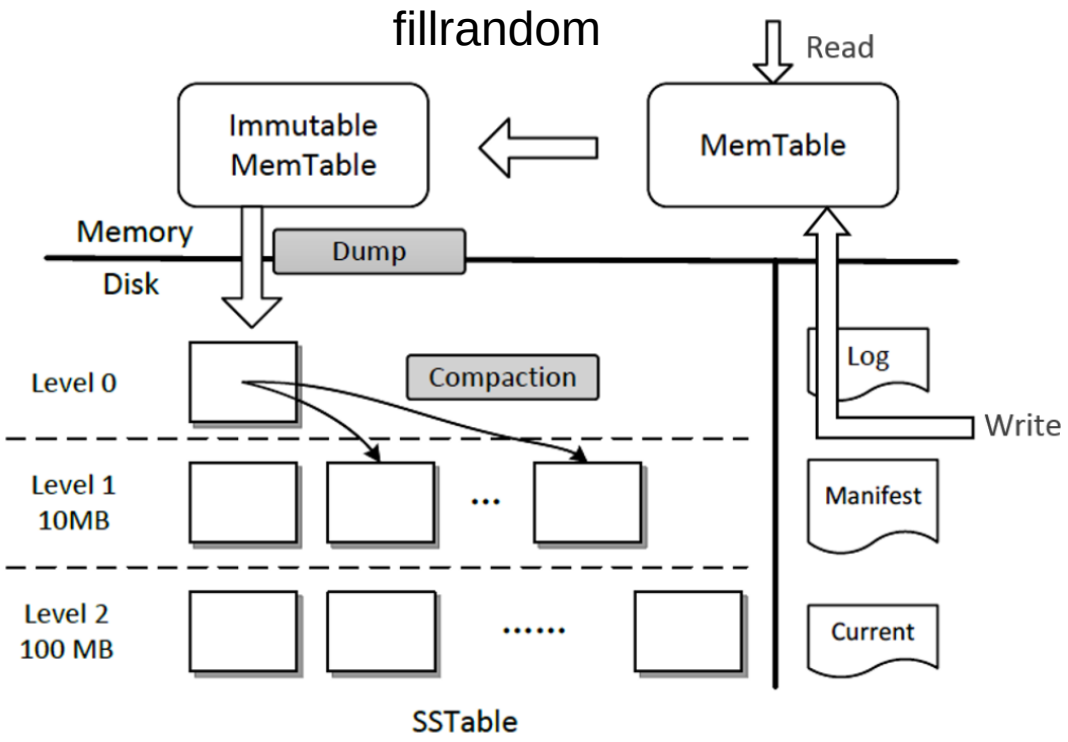
readrandom=
read random key * N

rand()
⬇
key
⬇
get(key)



k:10001 v:……
k:10002 v:……
k:10003 v:……
k:10004 v:……
k:…

3. 실험결과



rand()

Range(0~499999)
Repeat 500000 times

fill a key

memtable

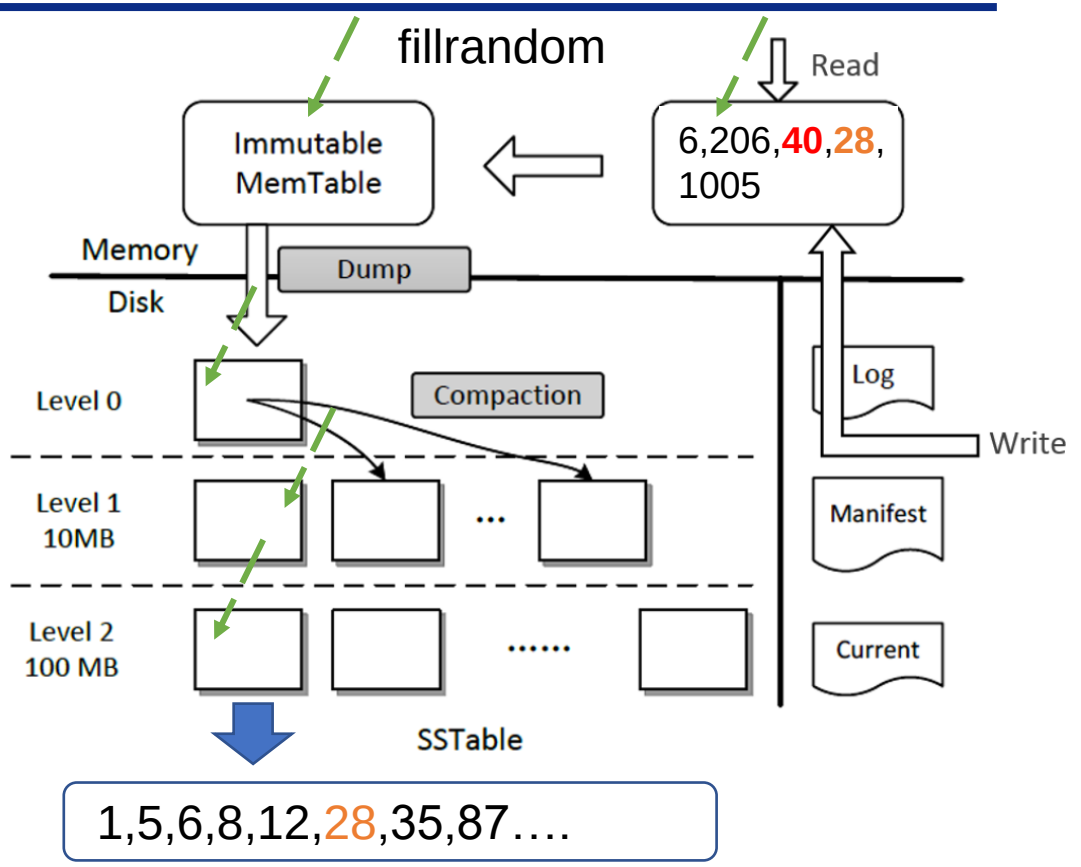
immutable

Level 0

⋮

```
--benchmarks="fillseq" \
--num=500000 \
```

3. 실험결과



40
658
60003
84121
1

readrandom=
read random key * N

rand()

Key = 4

get(key = 4)

Key not found

```
--benchmarks="fillseq" \
--num=500000 \
```

```
fillrandom   :       9.435 micros/op;   11.7 MB/s
Comparisons: 257066338

./db_bench --benchmarks=readrandom --use_existing_db=1 --reads=1

Keys:       16 bytes each
Values:     100 bytes each (50 bytes after compression)
Entries:    1000000
RawSize:    110.6 MB (estimated)
FileSize:   62.9 MB (estimated)
WARNING: Snappy compression is not enabled
------------------------------------------------
readrandom   :       2.932 micros/op; (706756 of 1000000 found)
```

Q&A

감사합니다

Thank you!