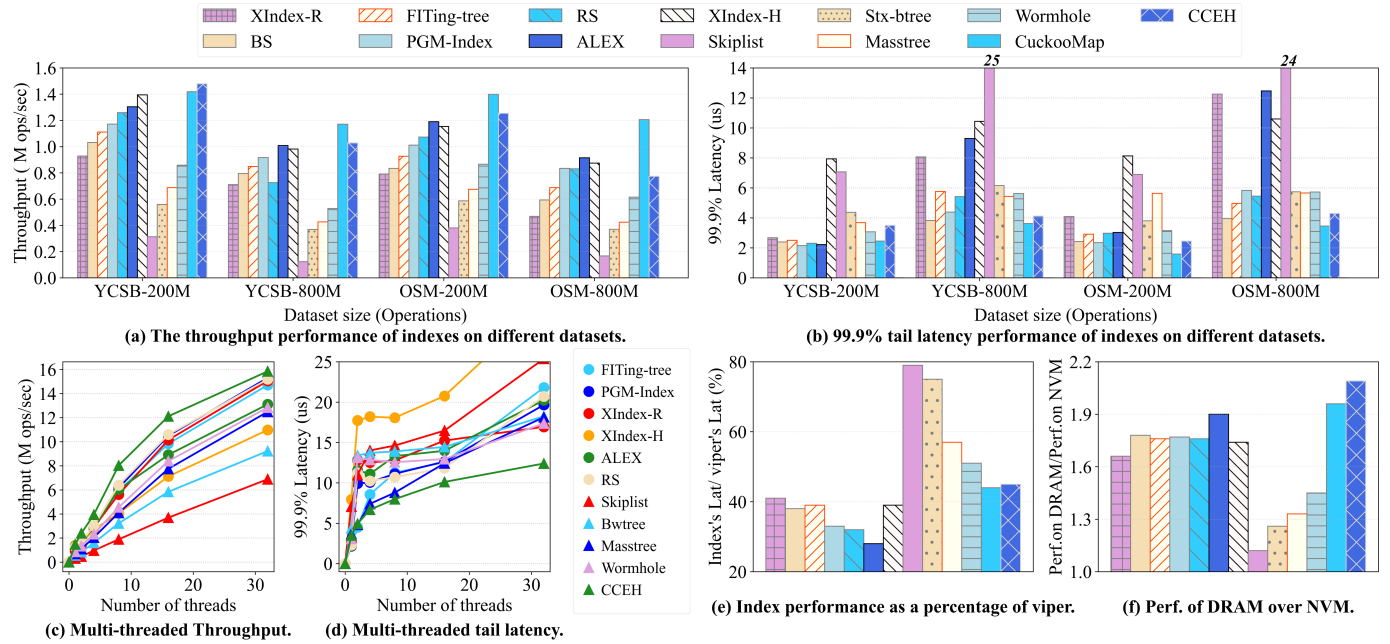
Cutting Learned Index into Pieces: An In-depth Inquiry into Updatable Learned Indexes

Jiake Ge[1,2], Boyu Shi[1,2], Yanfeng Chai[1,2], Yuanhui Luo[1,2], Yunda Guo[1,2], Yinxuan He[1,2] and Yunpeng Chai*[1,2]

[1]Key Laboratory of Data Engineering and Knowledge Engineering, MOE, China
[2]School of Information, Renmin University of China, China
*Corresponding Author: ypchai@ruc.edu.cn

APPENDIX

1. Read-only



(a) The throughput performance of indexes on different datasets.

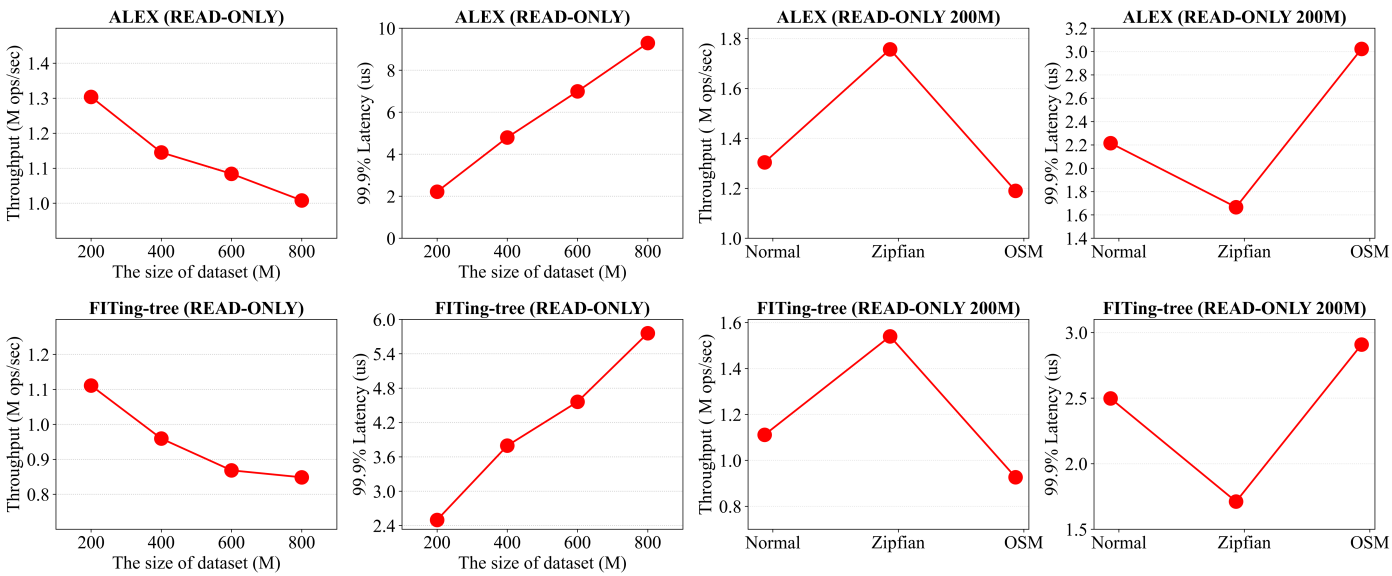
(b) 99.9% tail latency performance of indexes on different datasets.

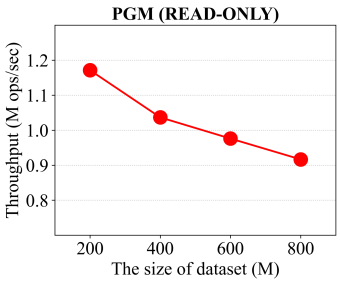
(c) Multi-threaded Throughput.

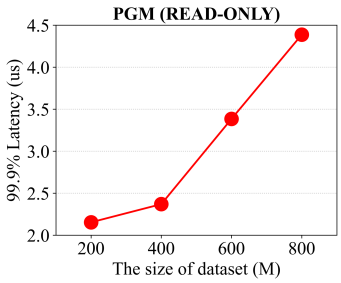
(d) Multi-threaded tail latency.

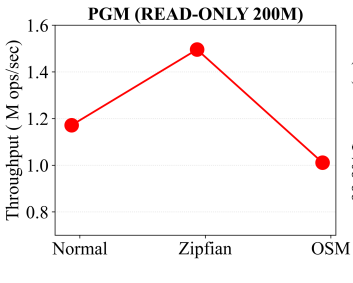
(e) Index performance as a percentage of viper.

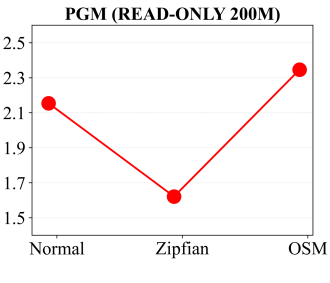
(f) Perf. of DRAM over NVM.

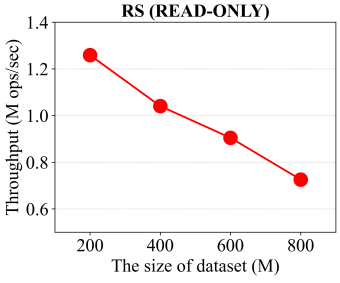


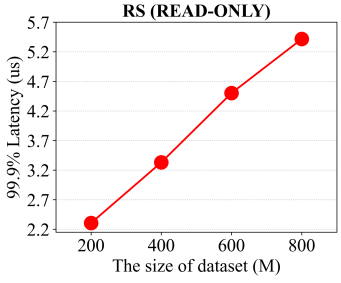


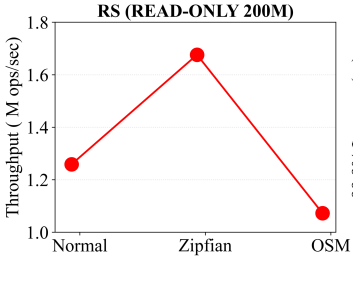


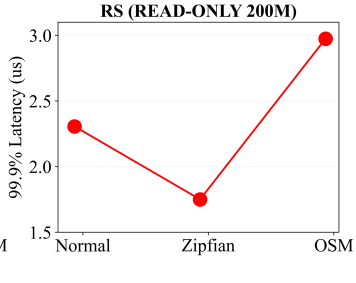


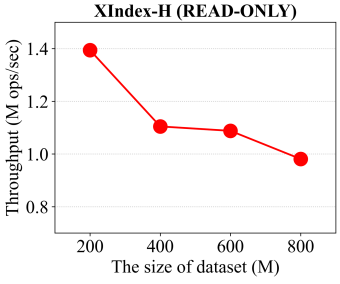


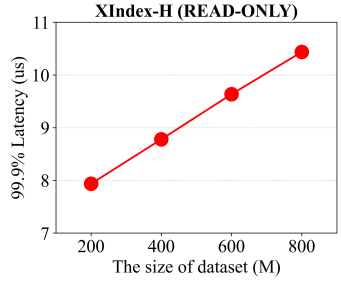


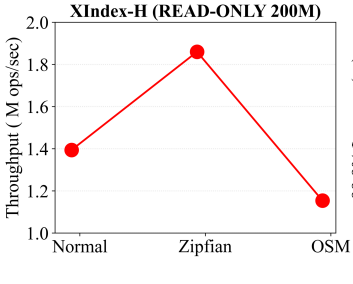


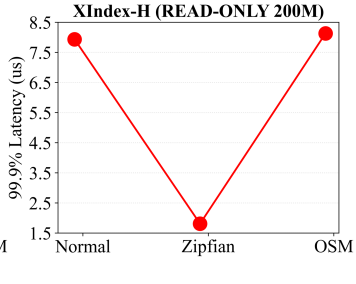


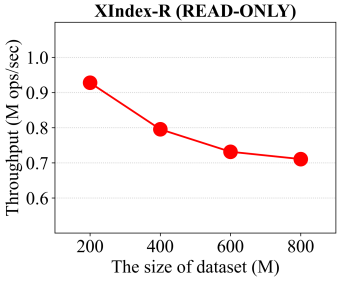


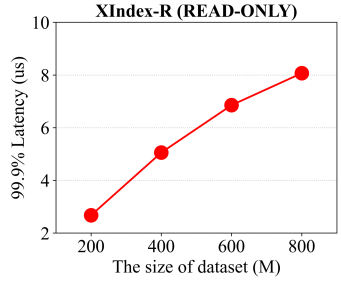


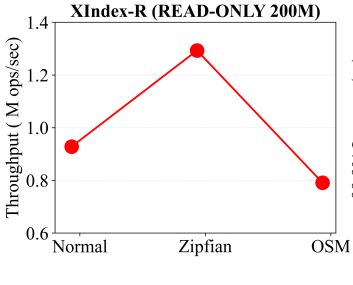


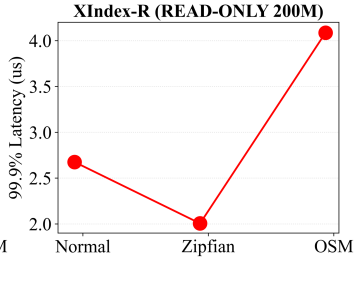


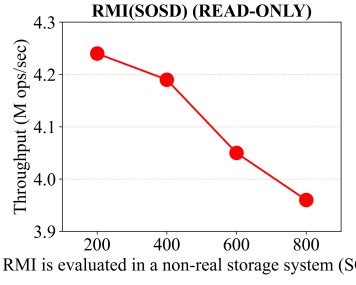


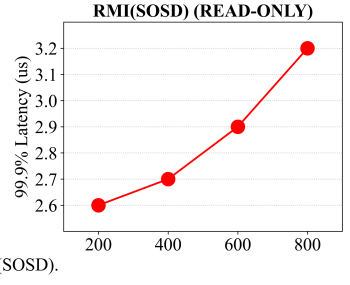


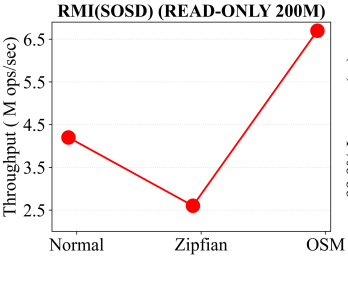


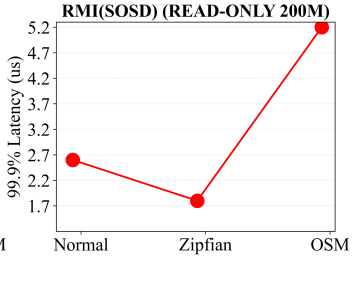




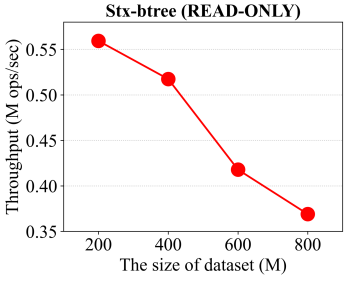


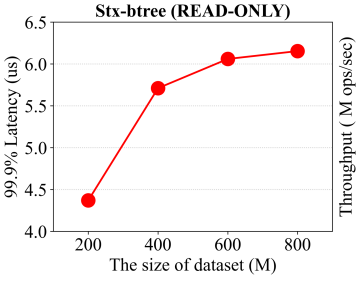


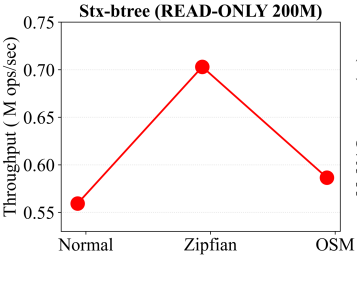


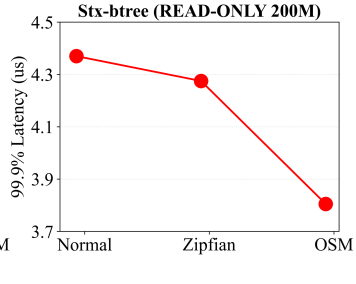


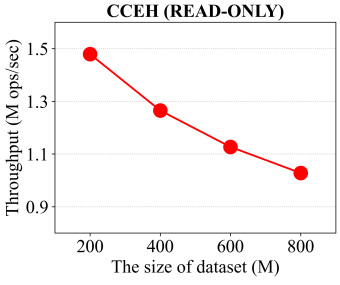
RMI is evaluated in a non-real storage system (SOSD).

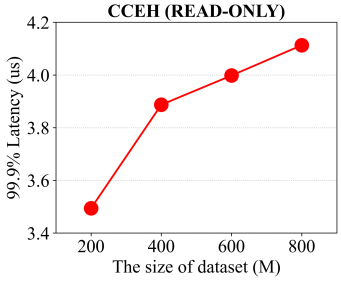


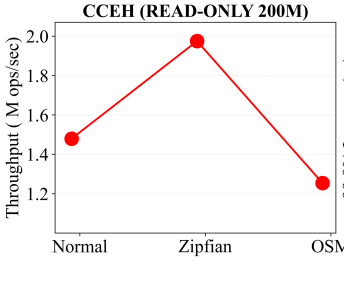


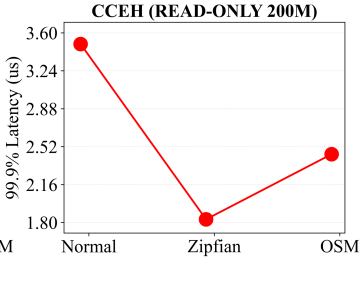


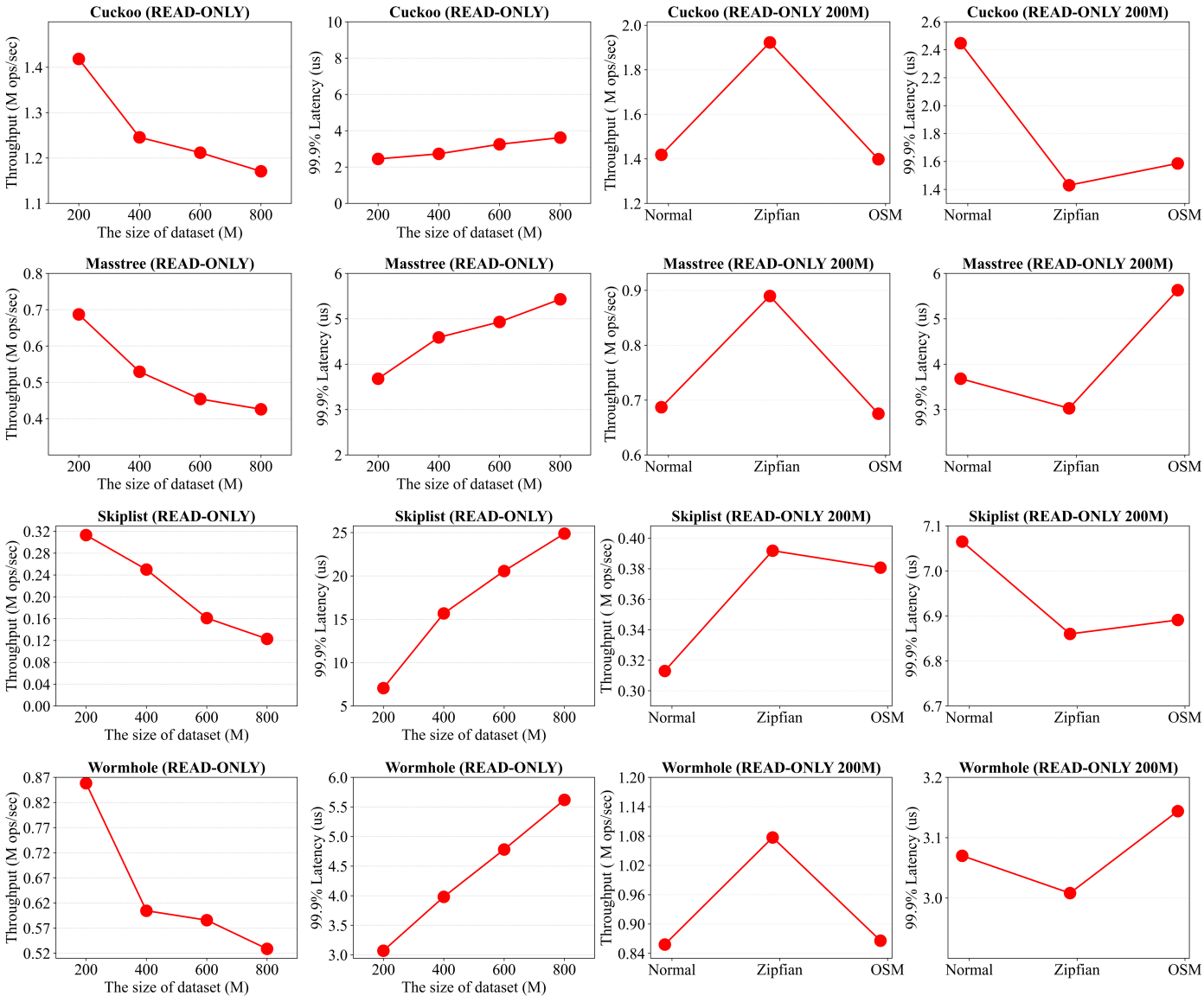




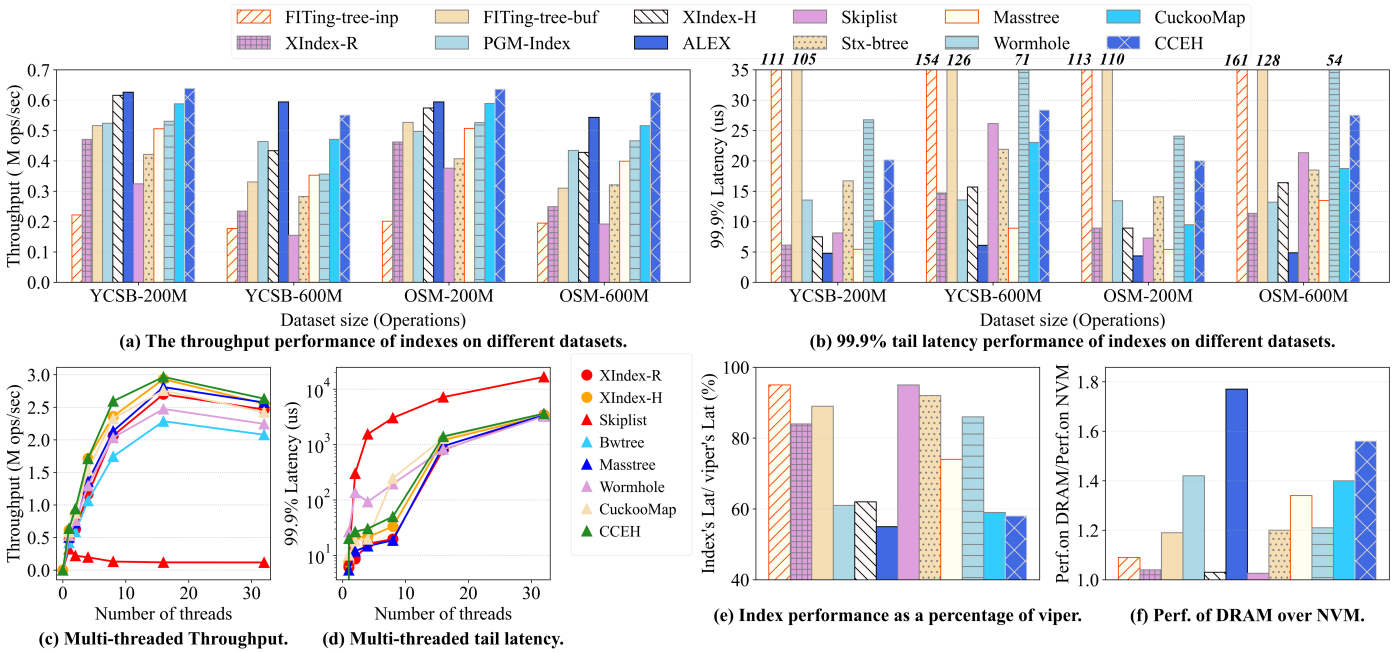








2. Write-only



(a) The throughput performance of indexes on different datasets.

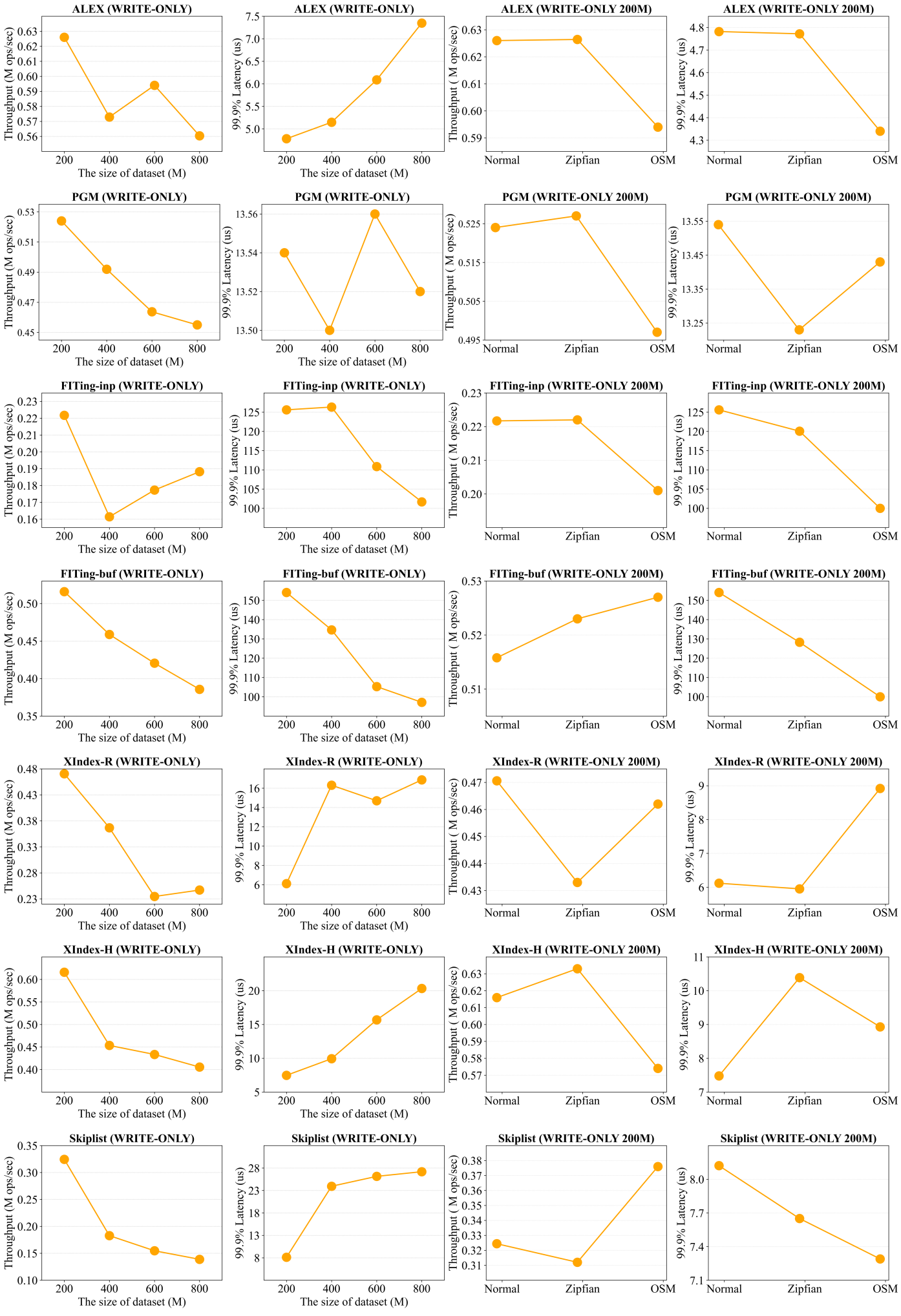
(b) 99.9% tail latency performance of indexes on different datasets.

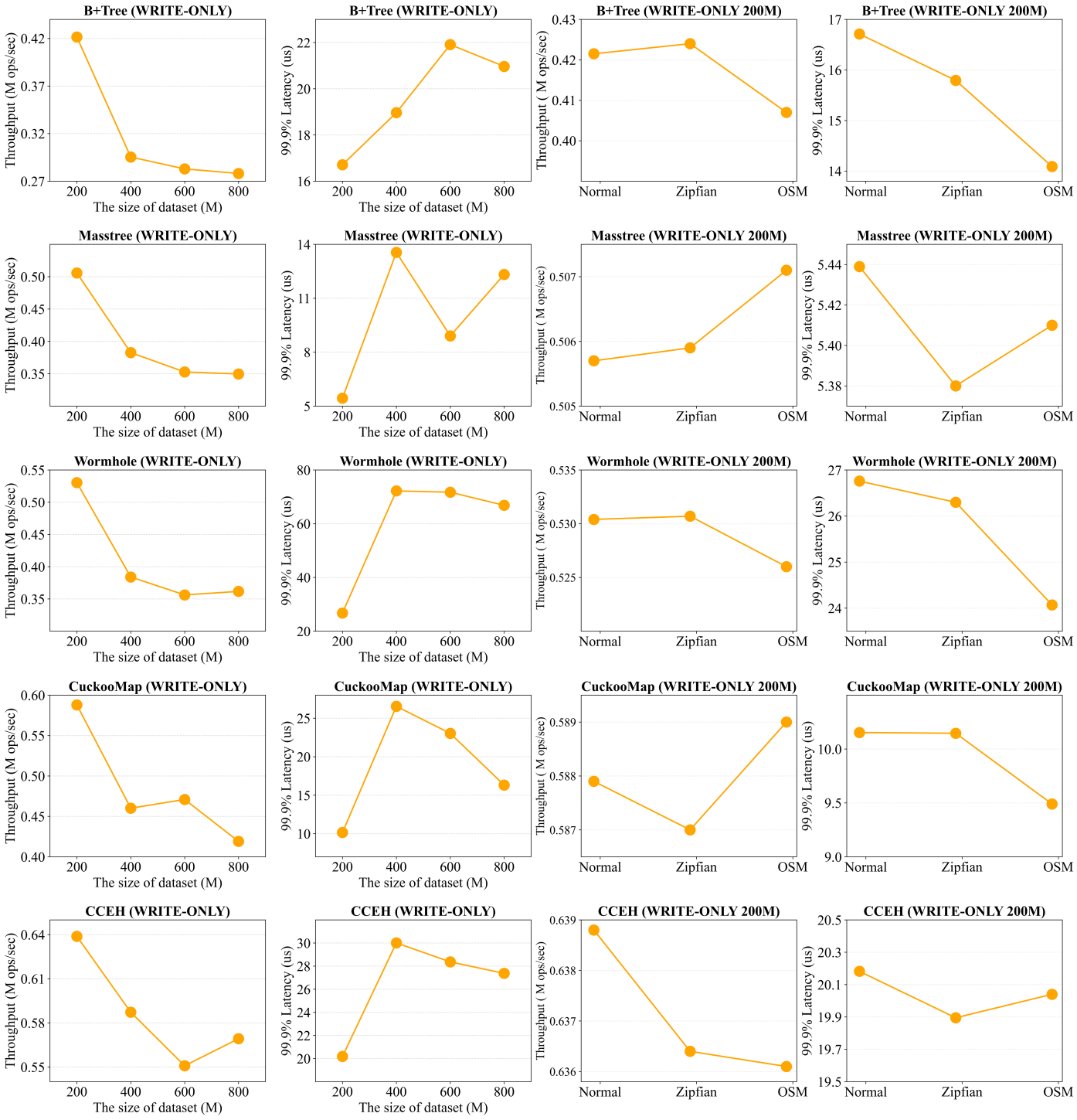
(c) Multi-threaded Throughput.

(d) Multi-threaded tail latency.

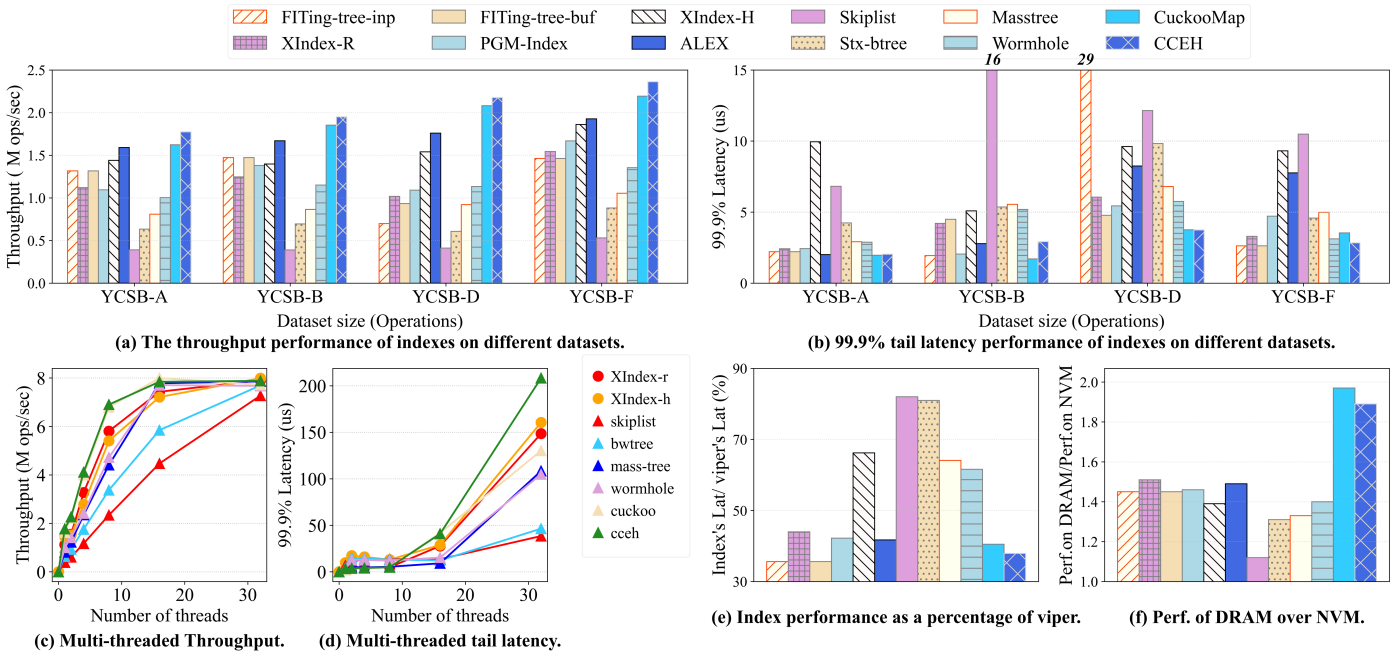
(e) Index performance as a percentage of viper.

(f) Perf. of DRAM over NVM.


ALEX (WRITE-ONLY)
Throughput (M ops/sec)
The size of dataset (M)
ALEX (WRITE-ONLY)
99.9% Latency (us)
The size of dataset (M)
ALEX (WRITE-ONLY 200M)
Throughput (M ops/sec)
Normal
Zipfian
OSM
ALEX (WRITE-ONLY 200M)
99.9% Latency (us)
Normal
Zipfian
OSM
PGM (WRITE-ONLY)
Throughput (M ops/sec)
The size of dataset (M)
PGM (WRITE-ONLY)
99.9% Latency (us)
The size of dataset (M)
PGM (WRITE-ONLY 200M)
Throughput (M ops/sec)
Normal
Zipfian
OSM
PGM (WRITE-ONLY 200M)
99.9% Latency (us)
Normal
Zipfian
OSM
FITing-inp (WRITE-ONLY)
Throughput (M ops/sec)
The size of dataset (M)
FITing-inp (WRITE-ONLY)
99.9% Latency (us)
The size of dataset (M)
FITing-inp (WRITE-ONLY 200M)
Throughput (M ops/sec)
Normal
Zipfian
OSM
FITing-inp (WRITE-ONLY 200M)
99.9% Latency (us)
Normal
Zipfian
OSM
FITing-buf (WRITE-ONLY)
Throughput (M ops/sec)
The size of dataset (M)
FITing-buf (WRITE-ONLY)
99.9% Latency (us)
The size of dataset (M)
FITing-buf (WRITE-ONLY 200M)
Throughput (M ops/sec)
Normal
Zipfian
OSM
FITing-buf (WRITE-ONLY 200M)
99.9% Latency (us)
Normal
Zipfian
OSM
XIndex-R (WRITE-ONLY)
Throughput (M ops/sec)
The size of dataset (M)
XIndex-R (WRITE-ONLY)
99.9% Latency (us)
The size of dataset (M)
XIndex-R (WRITE-ONLY 200M)
Throughput (M ops/sec)
Normal
Zipfian
OSM
XIndex-R (WRITE-ONLY 200M)
99.9% Latency (us)
Normal
Zipfian
OSM
XIndex-H (WRITE-ONLY)
Throughput (M ops/sec)
The size of dataset (M)
XIndex-H (WRITE-ONLY)
99.9% Latency (us)
The size of dataset (M)
XIndex-H (WRITE-ONLY 200M)
Throughput (M ops/sec)
Normal
Zipfian
OSM
XIndex-H (WRITE-ONLY 200M)
99.9% Latency (us)
Normal
Zipfian
OSM
Skiplist (WRITE-ONLY)
Throughput (M ops/sec)
The size of dataset (M)
Skiplist (WRITE-ONLY)
99.9% Latency (us)
The size of dataset (M)
Skiplist (WRITE-ONLY 200M)
Throughput (M ops/sec)
Normal
Zipfian
OSM
Skiplist (WRITE-ONLY 200M)
99.9% Latency (us)
Normal
Zipfian
OSM


B+Tree (WRITE-ONLY)
Throughput (M ops/sec)
0.42
0.37
0.32
0.27
200
400
600
800
The size of dataset (M)
B+Tree (WRITE-ONLY)
99.9% Latency (us)
22
20
18
16
200
400
600
800
The size of dataset (M)
B+Tree (WRITE-ONLY 200M)
Throughput (M ops/sec)
0.43
0.42
0.41
0.40
Normal
Zipfian
OSM
B+Tree (WRITE-ONLY 200M)
99.9% Latency (us)
17
16
15
14
Normal
Zipfian
OSM
Masstree (WRITE-ONLY)
Throughput (M ops/sec)
0.50
0.45
0.40
0.35
200
400
600
800
The size of dataset (M)
Masstree (WRITE-ONLY)
99.9% Latency (us)
14
11
8
5
200
400
600
800
The size of dataset (M)
Masstree (WRITE-ONLY 200M)
Throughput (M ops/sec)
0.507
0.506
0.505
Normal
Zipfian
OSM
Masstree (WRITE-ONLY 200M)
99.9% Latency (us)
5.44
5.42
5.40
5.38
Normal
Zipfian
OSM
Wormhole (WRITE-ONLY)
Throughput (M ops/sec)
0.55
0.50
0.45
0.40
0.35
200
400
600
800
The size of dataset (M)
Wormhole (WRITE-ONLY)
99.9% Latency (us)
80
60
40
20
200
400
600
800
The size of dataset (M)
Wormhole (WRITE-ONLY 200M)
Throughput (M ops/sec)
0.535
0.530
0.525
Normal
Zipfian
OSM
Wormhole (WRITE-ONLY 200M)
99.9% Latency (us)
27
26
25
24
Normal
Zipfian
OSM
CuckooMap (WRITE-ONLY)
Throughput (M ops/sec)
0.60
0.55
0.50
0.45
0.40
200
400
600
800
The size of dataset (M)
CuckooMap (WRITE-ONLY)
99.9% Latency (us)
25
20
15
10
200
400
600
800
The size of dataset (M)
CuckooMap (WRITE-ONLY 200M)
Throughput (M ops/sec)
0.589
0.588
0.587
Normal
Zipfian
OSM
CuckooMap (WRITE-ONLY 200M)
99.9% Latency (us)
10.0
9.5
9.0
Normal
Zipfian
OSM
CCEH (WRITE-ONLY)
Throughput (M ops/sec)
0.64
0.61
0.58
0.55
200
400
600
800
The size of dataset (M)
CCEH (WRITE-ONLY)
99.9% Latency (us)
30
28
26
24
22
20
200
400
600
800
The size of dataset (M)
CCEH (WRITE-ONLY 200M)
Throughput (M ops/sec)
0.639
0.638
0.637
0.636
Normal
Zipfian
OSM
CCEH (WRITE-ONLY 200M)
99.9% Latency (us)
20.5
20.3
20.1
19.9
19.7
19.5
Normal
Zipfian
OSM

3. Read-Write



(a) The throughput performance of indexes on different datasets.

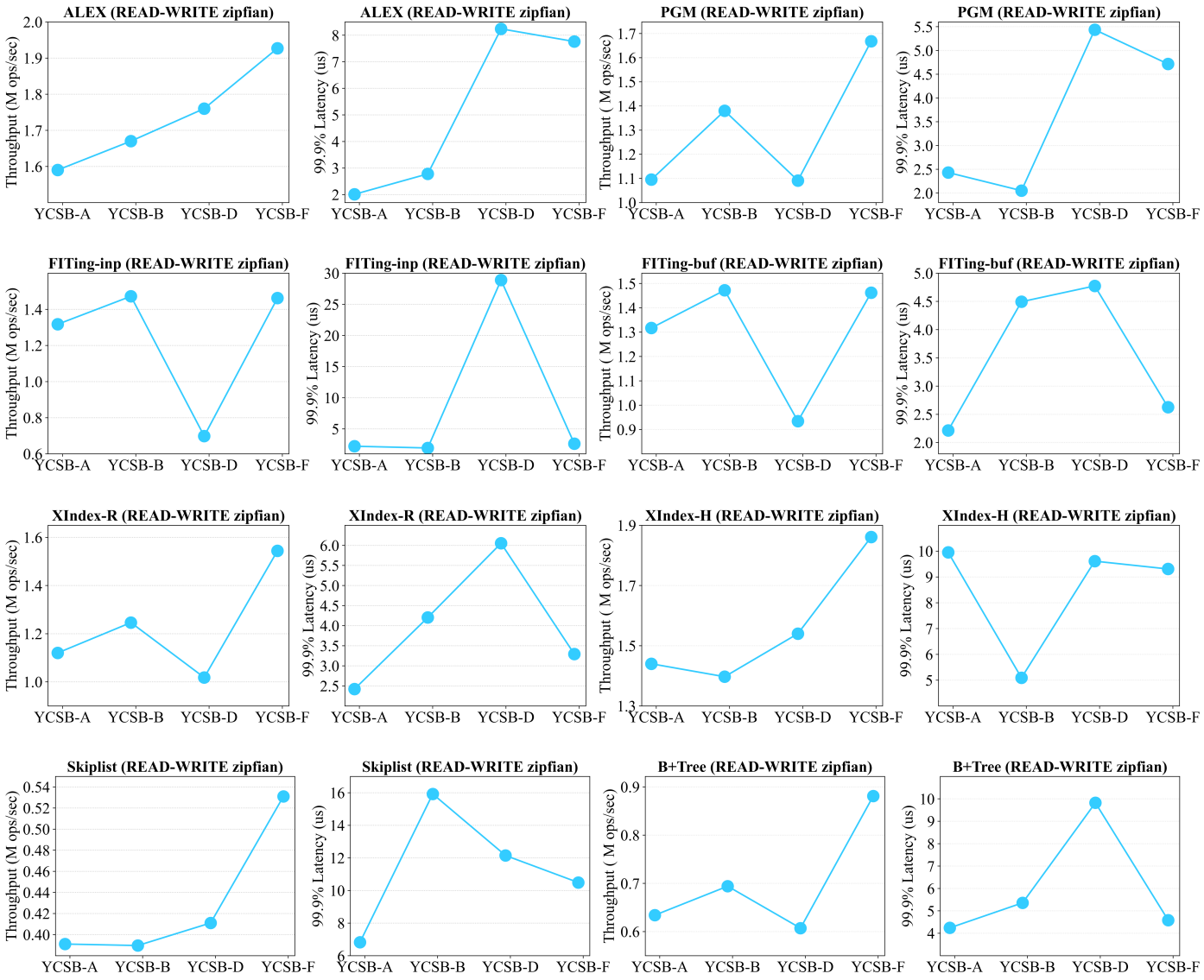
(b) 99.9% tail latency performance of indexes on different datasets.

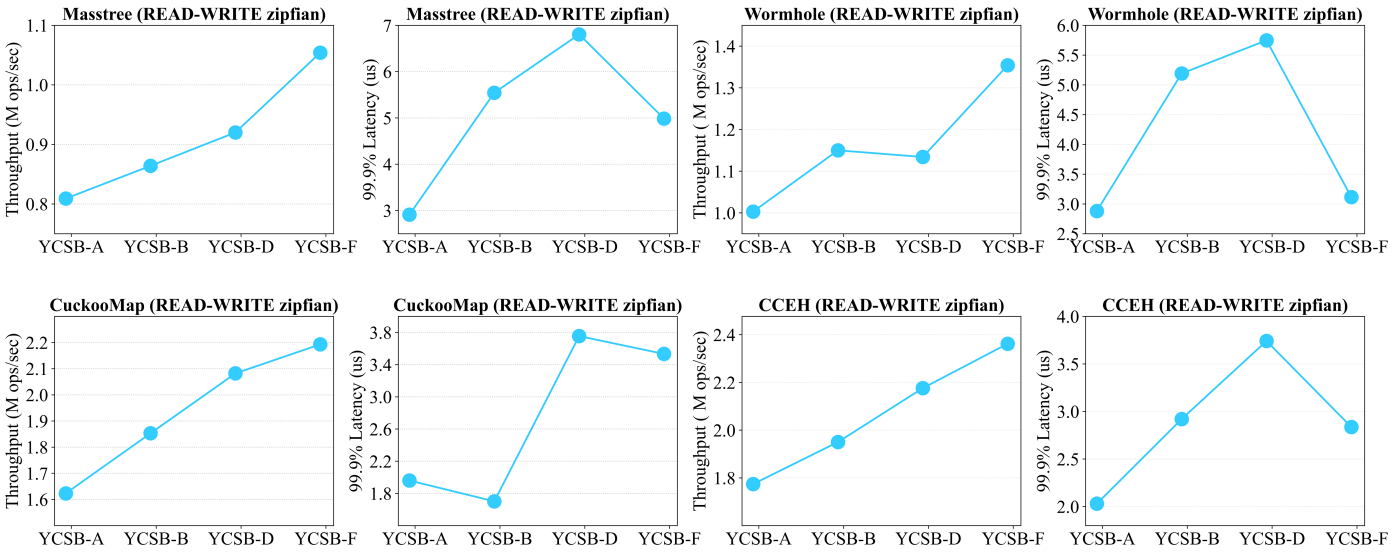
(c) Multi-threaded Throughput.

(d) Multi-threaded tail latency.

(e) Index performance as a percentage of viper.

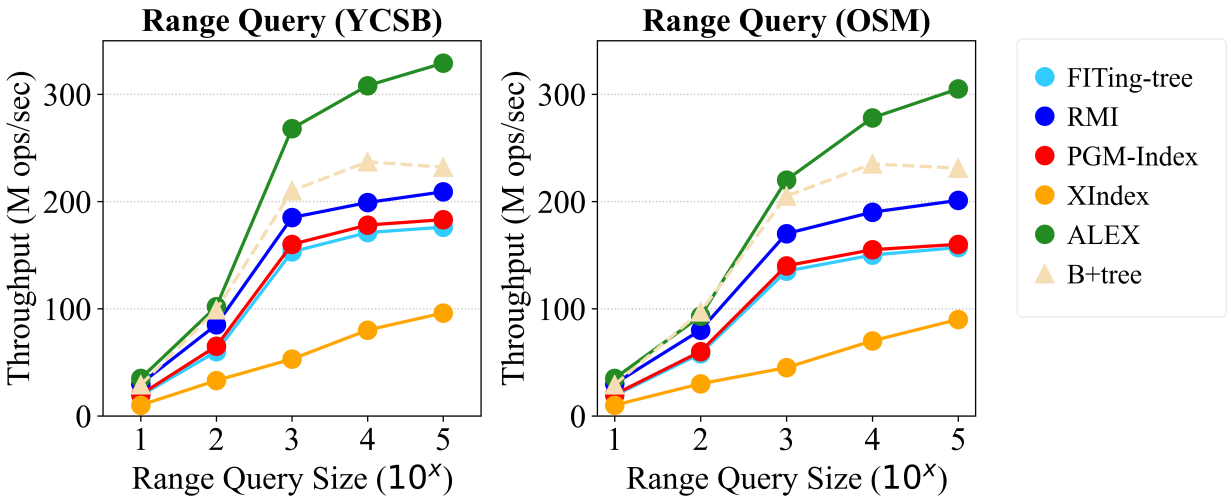
(f) Perf. of DRAM over NVM.



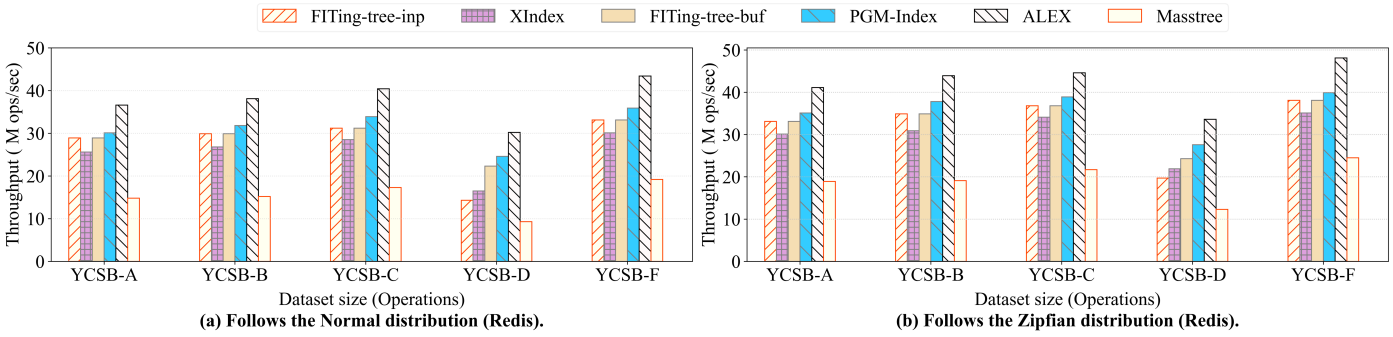


4. The Performance of The Range Query.

This experiment evaluates range query performance. We bulk load each index using the whole dataset of 200M keys and start a read-only scan workload. Each query picks a random start key *K* and fetches a fixed number of keys starting from *K*. Each workload issues 10 million range queries in total and we measure the through-put in number of keys accessed per second.



5. Employ the Redis as the evaluation environment.



(a) Follows the Normal distribution (Redis). **(b) Follows the Zipfian distribution (Redis).**

6. Recursive model indexes (RMI)

The Structure of RMI: When all keys can fit in memory, RMIs with more than two stages are almost never required. Thus, here we explain only two-stage RMIs for simplicity. A two-stage RMI is a CDF approximator A trained on $|D|$ data points (key / index pairs). The RMI approximator A is composed of a single first stage model f_1, and B second-stage models $f_2{}^i$. The value B is referred to as the "branching factor" of the RMI. Formally, the RMI is defined as:

$$A(x) = f_2^{\lfloor B \times f_1(x)/|D| \rfloor}(x)$$

Intuitively, the RMI first uses the stage-one model $f_1(x)$ to com-pute a rough approximation of the CDF of the input key x. This coarse-grained approximation is then scaled between 0 and the branching factor B, and this scaled value is used to select a model from the second stage, $f_2{}^i(x)$. The selected second-stage model is used to produce the final approximation. The stage-one model $f_1(x)$ can be thought of as partitioning the data into B buckets, and each second-stage model $f_2{}^i(x)$ is responsible for approximating the CDF of only the keys that fall into the ith bucket.

Training: Let $(x, y) \in D$ be the set of key / index pairs in the underlying data. Then, an RMI is trained by adjusting the parameters contained in $f_1(x)$ and $f_2{}^i(x)$ to minimize:

$$\sum_{(x,y)\in D} (F(x) - y)^2$$

Intuitively, minimizing Equation 2 is done by training "top down": first, the stage one model is trained, and then each stage 2 model is trained to fine-tune the prediction. Details can be found in paper "T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis. The Case for Learned Index Structures. In Proceedings of the 2018 International Conference on Management of Data, SIGMOD 18, New York, NY, USA, 2018. ACM."