Technical Report

In this report, we show the additional results of the experiment for concurrent query as Reviewer #5 suggested. We also slightly change the number of strips in Section 4.2 from $\lceil N^{1/3} \rceil$ to $\lceil (N/B)^{1/3} \rceil$ as Reviewer #4 asked. Reviewer #6 suggested that we can add more detailed analysis, so we also show the proofs of complexities.

Additional Experiment

The concurrent read-write operation is also possible in our method. For concurrency in insertion, we can do the merge operation in a background process. For the query, we can create several threads access to the data structure concurrently to improve the query efficiency. We conducted an experiment to get some preliminary results of concurrent query.

We picked the largest dataset in our experiment. BerlinMOD-large has 561 million insertions of 5-dimensional records. We built a WORQ-tree with $M = 100000$ for this dataset. Since our CPU is 6-core, we used our query algorithm to answer 100 queries of selectivity 0.5% with 1 to 6 threads. Table 5 shows how running time changes with the number of threads increasing. We can see that with a single thread we need 204.3 seconds to answer 100 queries, but we only need 42.9 seconds with 6 threads, which means that our method is fully compatible with concurrency query and concurrency can improve the query efficiency a lot.

Number of Threads	Running Time
1	204.3s
2	107.3s
3	72.2s
4	56.4s
5	49.8s
6	42.9s

Table 5: The result of additional experiment

Additional Proof

We present Lemma 1 and Lemma 2 to discuss the write efficiency and query efficiency of the LSM-segment tree.

LEMMA 1. *The amortized insertion I/O cost of LSM-Segment tree is* $O(\log_2 \frac{N}{M}/B)$.

PROOF. We first calculate the I/O cost of merging two segment trees. To build a segment tree of $M \cdot 2^i$ leaf nodes, we need a merge sort of all the leaf nodes, which costs $O(M \cdot 2^i/B)$ I/Os. Starting from the leaf level, we perform a linear scan of all the nodes in a level to build the upper level, which means we need $O(M \cdot 2^i/B)$ I/Os to scan a level with $M \cdot 2^i$ nodes and create a new level with $M \cdot 2^i/B$ nodes. So, the total cost of the level construction is $O(M \cdot 2^i/B) + O(M \cdot 2^i/B^2) + O(M \cdot 2^i/B^3) \cdots = O(M \cdot 2^i/B)$. Therefore, to build a segment tree of $M \cdot 2^i$ leaf nodes costs $O(M \cdot 2^i/B)$ I/Os.

Next,the i-th segment tree with $M \cdot 2^{i-1}$ leaf nodes will need at most $N/(M \cdot 2^{i-1})$ times of merging, so it totally needs $O(M \cdot 2^{i-1}/B) \cdot N/(M \cdot 2^{i-1}) = O(N/B)$ IOs. There are $\log_2 \frac{N}{M}$ segment trees, so the total I/O cost is $\log_2 \frac{N}{M} \cdot O(N/B) = O(\log_2 \frac{N}{M} N/B)$. Hence, for each insertion, the amortized I/O cost is $O(\log_2 \frac{N}{M} N/B)/N = O(\log_2 \frac{N}{M}/B)$. □

LEMMA 2. *The worst-case query I/O cost of LSM-Segment tree is* $O(\log_2 \frac{N}{M} \log_B \frac{N}{B})$.

PROOF. We first calculate the I/O cost of query on one segment tree. We access at most two blocks in each level in the query. The reason is that the query range will partially overlap with at most two blocks as the two ends on a segment. Since a segment tree can only have $\log_B(N/B)$ levels, to answer the query in one segment tree costs $O(\log_B(N/B))$ I/Os.

Since there are $\log_2 \frac{N}{M}$ segment trees, the total cost to obtain the whole aggregate result is $O(\log_2 \frac{N}{M} \log_B \frac{N}{B})$. □

By Lemma 1 and Lemma 2, we know that the write efficiency of the LSM-Segment tree is $O(\log_2 \frac{N}{M}/B)$ and the query efficiency is $O(\log_2 \frac{N}{M} \log_B \frac{N}{B})$.

Next, we present Lemma 3 and Lemma 4 to discuss the write efficiency and query efficiency of the LSM-Partition-segment tree.

LEMMA 3. *The amortized insertion I/O cost of LSM-Partition-Segment tree is* $O(\log_M \frac{N}{B} \log_2 \frac{N}{M}/B)$.

PROOF. We first calculate the I/O cost of one Partition-Segment tree. To build a partition-segment tree, we need an external sorting by $1-st$ dimension. The external sorting costs $O(\frac{N}{B} \log_M \frac{N}{B})$ I/Os. Then we need to sort each strip and build a segment tree for each strip, which totally costs $O(\frac{N}{B} \log_M \frac{N}{B})$ I/Os. Finally, we need to sort each grid and build a segment tree for each grid, which totally costs $O(\frac{N}{B} \log_M \frac{N}{B})$ I/Os. Therefore, it costs $O(\frac{N}{B} \log_M \frac{N}{B})$ I/Os to build a partition-segment tree.

Similar to Lemma 1, for each insertion, the amortized I/O cost is $O(\log_M \frac{N}{B} \log_2 \frac{N}{M}/B)$. □

LEMMA 4. *The worst-case query I/O cost of LSM-Partition-Segment tree is* $O((N/B)^{\frac{1}{3}} \log_B N \log_2 \frac{N}{M})$.

PROOF. We first calculate the I/O cost of query on one partition-segment tree. For the at most $\lceil (N/B)^{1/3} \rceil$ fully overlapped strips, search the second-dimension segment trees, which cost at most $O((N/B)^{\frac{1}{3}} \log_B \frac{N}{B})$ I/Os. For the at most 2 partially overlapped ones, there are two kinds of grids. For the at most $\lceil (N/B)^{1/3} \rceil$ fully overlapped ones, search the first-dimension segment trees, which cost at most $O((N/B)^{\frac{1}{3}} \log_B \frac{N}{B})$ I/Os. For at most 2 partially overlapped ones, scan the whole grid the obtain the answer, which cost $O(N^{\frac{1}{3}}/B)$ I/Os. Hence, the total query I/O cost is $O((N/B)^{\frac{1}{3}} \log_B \frac{N}{B}) + 2 \cdot O((N/B)^{\frac{1}{3}} \log_B \frac{N}{B}) + 4 \cdot O(N^{\frac{1}{3}}/B) = O((N/B)^{\frac{1}{3}} \log_B \frac{N}{B})$.

Since there are $\log_2 \frac{N}{M}$ partition-segment trees, the total cost to obtain the whole aggregate result is $O((N/B)^{\frac{1}{3}} \log_B N \log_2 \frac{N}{M})$. □

By Lemma 3 and Lemma 4, we know that the write efficiency of the LSM-Partition-Segment tree is $O(\log_M \frac{N}{B} \log_2 \frac{N}{M}/B)$ and the query efficiency is $O((N/B)^{\frac{1}{3}} \log_B N \log_2 \frac{N}{M})$.

Next, we present Lemma 5 and Lemma 6 to discuss the write efficiency and query efficiency of the WORQ-tree.

LEMMA 5. *The amortized insertion I/O cost of WORQ-tree is* $O(d \log_M \frac{N}{B} \log_2 \frac{N}{M}/B)$.

PROOF. Given that there are d levels of construction and in each level the construction costs $O(\frac{N}{B} \log_M \frac{N}{B})$ I/Os, it costs $O(d\frac{N}{B} \log_M \frac{N}{B})$ I/Os to build one tree in a WORQ-tree.

Hence, similar to Lemma 1, considering there are $\lceil \log_2 \frac{N}{M} \rceil$ trees, the amortized cost of insertion is $O(d \log_M \frac{N}{B} \log_2 \frac{N}{M}/B)$. $\square$

LEMMA 6. *The worst-case query I/O cost of WORQ-tree is* $O((N/B)^{\frac{2d-3}{2d-1}} \log_B \frac{N}{B} \log_2 \frac{N}{M})$.

PROOF. We prove it by induction. Assume that the time complexity for d-dimension queries in one tree is $O((N/B)^{\frac{2d-3}{2d-1}} \log_B \frac{N}{B})$. When $d = 2$, it obviously holds. When $d > 2$, there are two kinds of strips: at most two strips partially overlapped with the query range in the d-th dimansion and at most $\lceil (N/B)^{\frac{2}{2d-1}} \rceil$ strips fully overlapped. For the at most two partially overlapped strips, it costs $O(N^{\frac{2d-3}{2d-1}}/B)$ I/Os to scan the whole strip to obtain the result. For the at most $\lceil (N/B)^{\frac{2}{2d-1}} \rceil$ fully overlapped strips, it costs $O(\left((N/B)^{\frac{2d-3}{2d-1}}\right)^{\frac{2d-5}{2d-3}} \log_B N) = O((N/B)^{\frac{2d-5}{2d-1}} \log_B \frac{N}{B})$ I/Os in each strip, so it totally cost $O((N/B)^{\frac{2d-3}{2d-1}} \log_B \frac{N}{B})$ I/Os.

Hence, since there are $\lceil \log_2 \frac{N}{M} \rceil$ trees, the I/O cost of query is $O((N/B)^{\frac{2d-3}{2d-1}} \log_B \frac{N}{B} \log_2 \frac{N}{M})$. $\square$

By Lemma 5 and Lemma 6, we know that the write efficiency of the WORQ-tree is $O(d \log_M \frac{N}{B} \log_2 \frac{N}{M}/B)$ and the query efficiency is $O((N/B)^{\frac{2d-3}{2d-1}} \log_B \frac{N}{B} \log_2 \frac{N}{M})$.