The **scale** role :

1. General Purpose of scale:

- The scale parameter in the **LatencyParams** structure determines the size of the workload generated for the testing of different database backends.

- Specifically, scale influences the number of transactions (or rows) that are generated and executed in the test scenario.

2. Influence on Transaction Count:

- In the code within the **test_backend_latency** function, scale is directly used to control how many transactions are created and executed during the test:

- let n = params.scale;

Here, n represents the total number of transactions to be executed, which is derived from the scale parameter.

- The scale parameter is divided by 1000 to compute the effective scale (in terms of thousands of transactions) when building the base database or when performing operations like batch executions:

```
let base = format!("{}-{}-{}k-fan{}-ratio{}-mem{}", result_path, params.index_name,
params.scale/1000, params.mht_fanout, params.size_ratio, params.mem_size);
```

3. Relation to Block Height:

- Although not directly controlling the block height, the scale parameter indirectly influences it by determining how many transactions need to be packed into each block.

- The code packs a block after a certain number of transactions (defined by params.tx_in_block):

```
if requests.len() == tx_in_block {
    // Pack up a block and execute it
    println!("block id: {}", block_id);
    batch_exec_tx(requests.clone(), caller_address, block_id, &mut backend);
    block_id += 1;
    requests.clear();
}
```

- The block height is then incremented with each batch of transactions.

The relationship between the scale parameter and the block height

Long story short, it is indirect and depends on the number of transactions processed per block and the total number of transactions generated by the scale parameter.

1. Scale Parameter:

- The scale parameter represents the number of transactions (n) that are used in the test. This is the main factor determining how many transactions will be executed in the latency test.

2. Number of Transactions in a Block (tx_in_block):

- The **tx_in_block** parameter defines how many transactions are packed into a single block.

3. Total Number of Blocks (Block Height):

- The block height in this context refers to the total number of blocks created during the test. It is determined by dividing the total number of transactions (n) by the number of transactions per block (tx_in_block).

Relation Between Scale and Block Height

The relation can be summarized with the following formula:

Block Height = [scale / tx_in_block]

- The code packs **tx_in_block** transactions into a block. After all transactions are processed, the block height equals the total number of blocks needed to process all transactions.

For example, if scale is 1000 and tx_in_block is 100, then 10 blocks will be created.

Example From Code:

- In the Rust code inside the **test_backend_latency** function, transactions are generated and grouped into blocks.

- Transactions are processed in blocks with the following line:

```
if requests.len() == tx_in_block {...
```

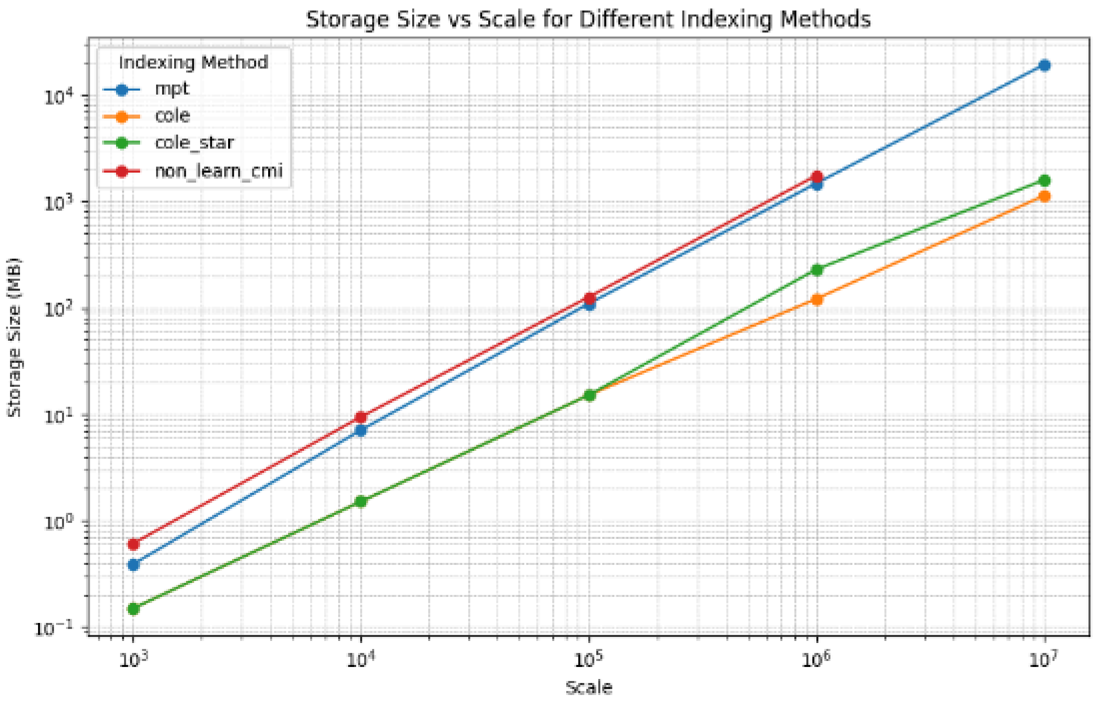
which triggers a new block when the number of transactions (requests.len()) reaches **tx_in_block**.

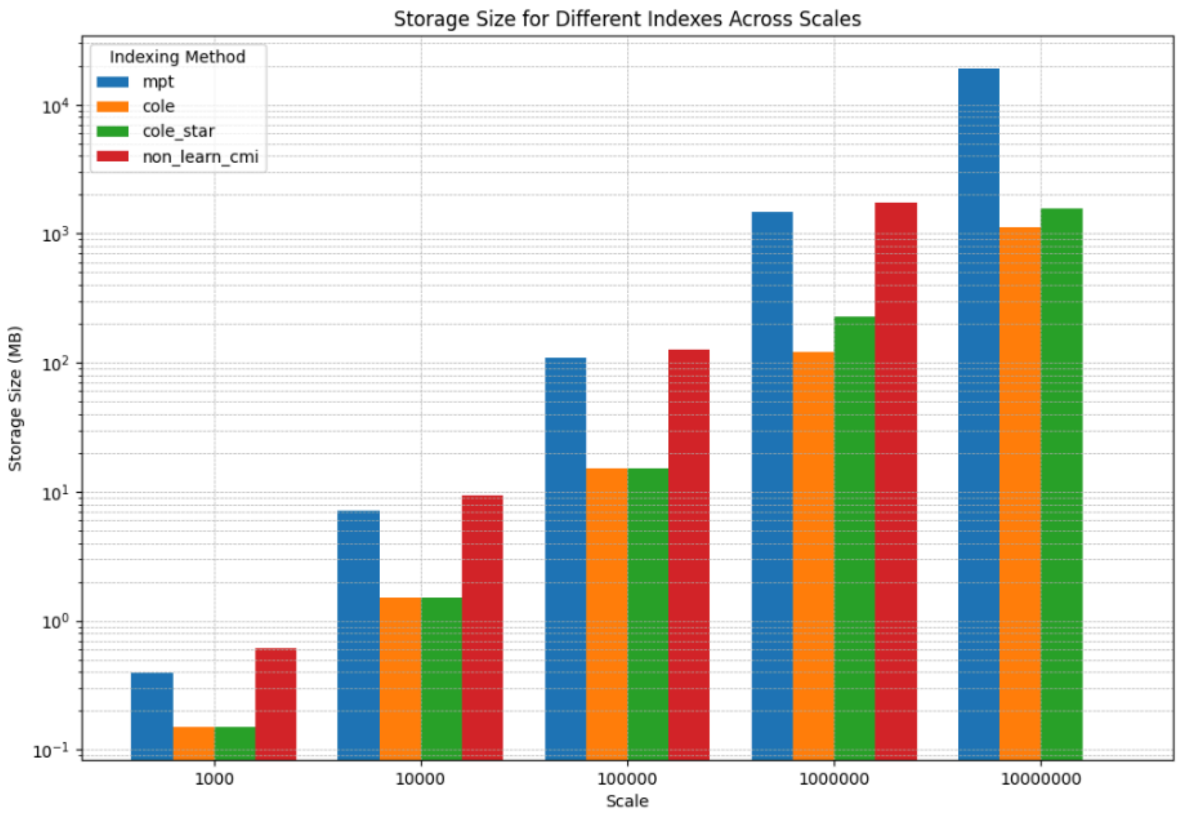
- This loop continues until all transactions (scale number) are processed, determining the final block height.

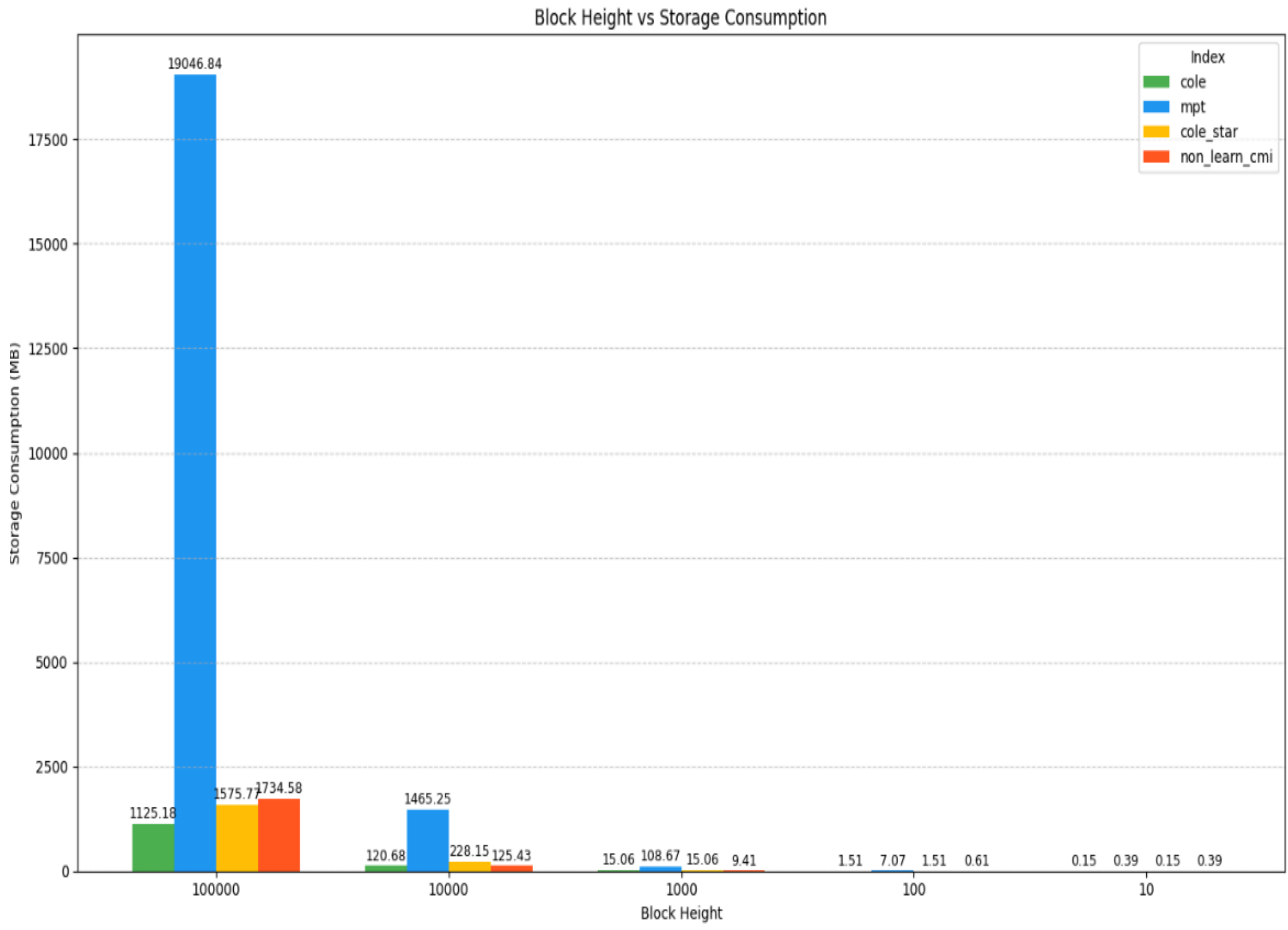
- increasing the Scale: If the scale parameter is increased, more transactions are created, and consequently, more blocks will be required if the tx_in_block parameter remains constant.

- Impact on Block Height: The block height is directly proportional to the scale when divided by the number of transactions per block (tx_in_block). The larger the scale, the more blocks will be generated.
This means that the scale parameter controls how many transactions are executed, which in turn determines how many blocks will be created during the latency test.

Plotting the indexes vs. different scales:


Storage Size vs Scale for Different Indexing Methods
Indexing Method
mpt
cole
cole_star
non_learn_cmi
Storage Size (MB)
Scale
10^{-1}
10^0
10^1
10^2
10^3
10^4
10^3
10^4
10^5
10^6
10^7


Storage Size for Different Indexes Across Scales
Indexing Method
mpt
cole
cole_star
non_learn_cmi
Storage Size (MB)
Scale
10^{-1}
10^0
10^1
10^2
10^3
10^4
1000
10000
100000
1000000
10000000


Block Height vs Storage Consumption
Index
cole
mpt
cole_star
non_learn_cmi
Storage Consumption (MB)
Block Height
0
2500
5000
7500
10000
12500
15000
17500
100000
10000
1000
100
10
1125.18
19046.84
1575.77
1734.58
120.68
1465.25
228.15
125.43
15.06
108.67
15.06
9.41
1.51
7.07
1.51
0.61
0.15
0.39
0.15
0.39