Updatable Learned Index with Precise Positions
Experiment

Wu J, Zhang Y, Chen S, et al., **2021 VLDB**

2024. 02. 28

Presentation by Nakyeong Kim, Suhwan Shin
nkkim@dankook.ac.kr, shshin@dankook.ac.kr

Contents

1. Introduction

1) Motivation

Problems with LIPP

- Not tolerate errors
- Create child nodes when conflict occurs (conflict-based structural modification)
- The more conflicts → the higher height of tree → **space amplification**
- Violates the space efficient principle of learned inde

Goal

- Analyze the impact of space amplification due to conflicts
- <u>Try to solve it</u>

\+ Also, analyze the performance of range query

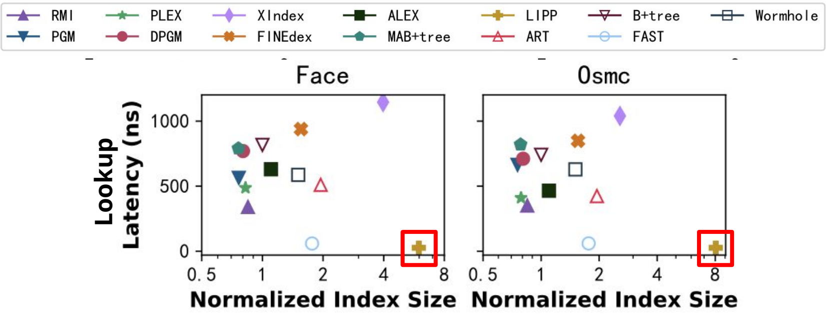


Figure 1: Trade-off of performance and normalized index size.

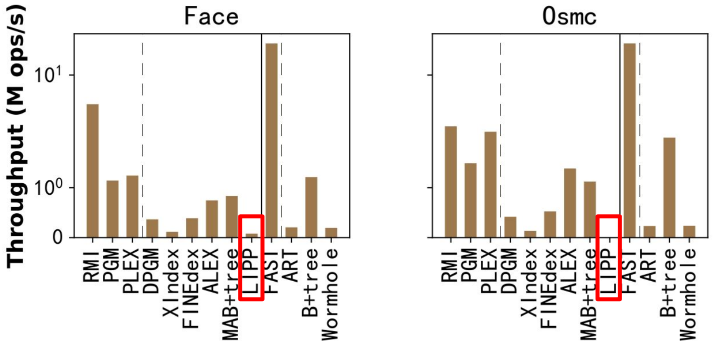


Figure 2: Throughput of range queries.

1. Introduction

Node Utilization: Proportion of non-null entries.

2) Observe with a focus on Utilization

Most nodes have less than 30% of all entries

→ There exists upper bound

$$T_{\mathcal{M}} = \max_{l \in [0, L-1]} |\{k \in \mathcal{K} | \mathcal{M}(k) == l\}|$$

We observe that there exists an upper bound for the minimum $T_{\mathcal{M}}$, i.e. $\exists \mathcal{M}, T_{\mathcal{M}} \leq \lceil \frac{N}{3} \rceil$ where N is the number of keys in $\mathcal{K}$, i.e. $N = |\mathcal{K}|$. However, the $\lceil \frac{N}{3} \rceil$ may not be the tightest upper bound in many cases. Thus, our goal is to find a best model $\mathcal{M} = A\mathcal{G}(k) + b$ with the minimum conflict degree $T_{\mathcal{M}}$.

We think that **TM** will be an important factor of space amplification

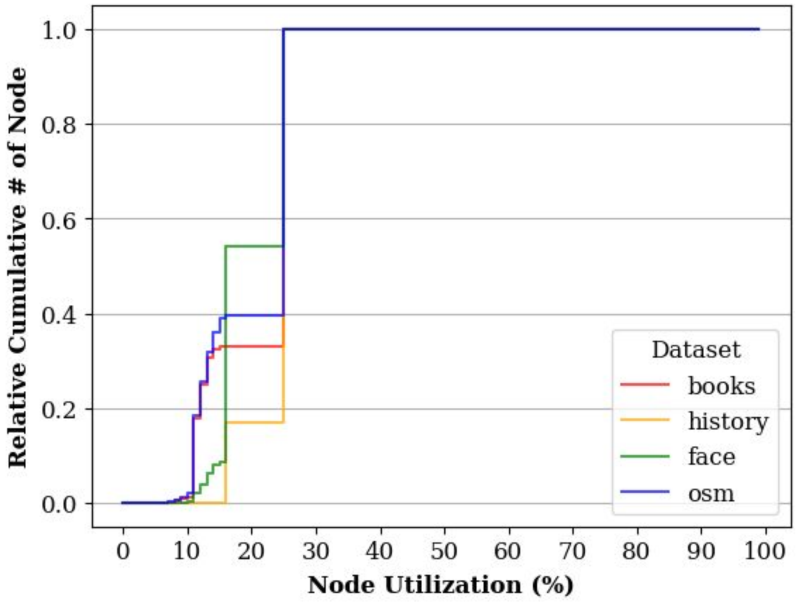


Figure 3: Node utilization CDF.

1. Introduction

Fill factor: How many fill space when given new keys, inverse of utilization
e.g., if fill factor is 2, node utilization is 1/2

3) Parameters of LIPP

- We said that our observation, node utilization upper-bound, is caused by TM, but it is not true

- We found that the factor of affecting to utilization is fill factor(**initial node size**, **gap count**)

- We assume that controlling this factor will change performance and index size

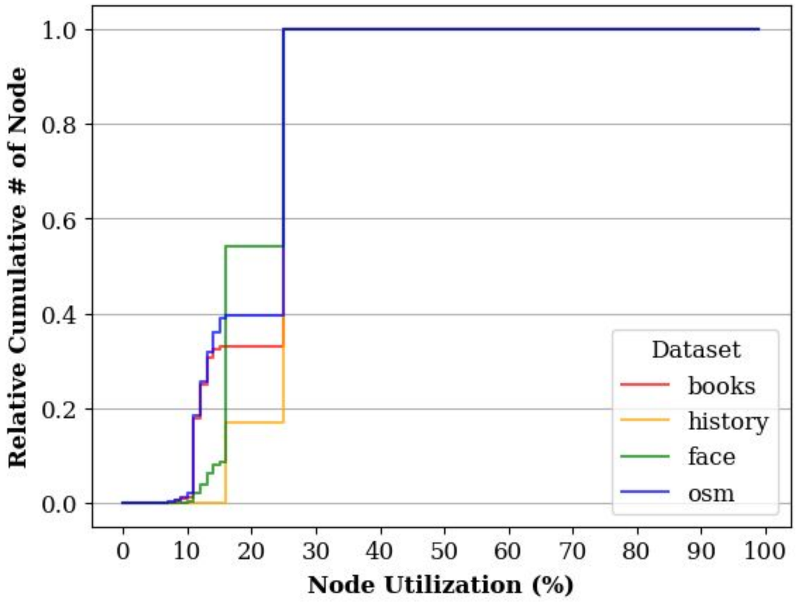


Figure 3: Node utilization CDF.

1. Introduction

4) To do Works

1. A sensitivity analysis into updatable learned index structure
   - Node utilization management policy: **fill factor**
   - Model: simple linear regression vs kernelized linear regression
   - Conflict resolving: shifting vs chaining
   - SMO(Structural modification operation): cost-benefit(fanout tree) vs conflict-proportion

1. Introduction

4) To do Works

2. Performance comparison between ALEX and LIPP through size
   - Need understanding of fill factor(parameter) each indexes

3. Which techniques are appropriate when considering performance versus space?
   - Create a new index based on that analysis
     + Conflict resolving : error-controlled approach (shift-chain hybrid)
     + Concurrency-friendly : semi-ordered

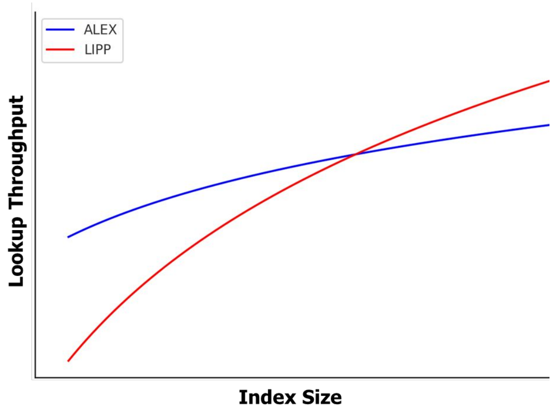


Figure 4: Expected lookup throughput according to size.

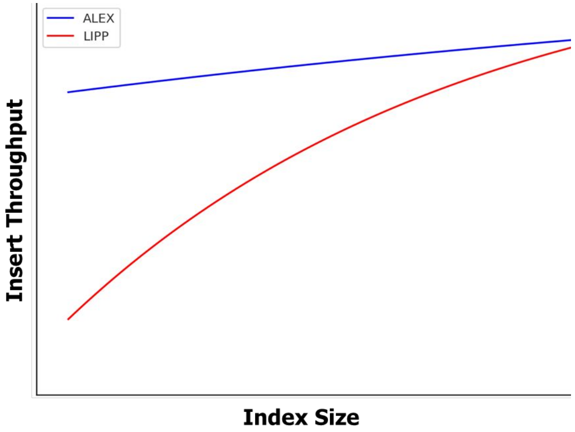


Figure 5: Expected insert throughput according to size.

2. Observations

1-1) Node Utilization

Node size set when

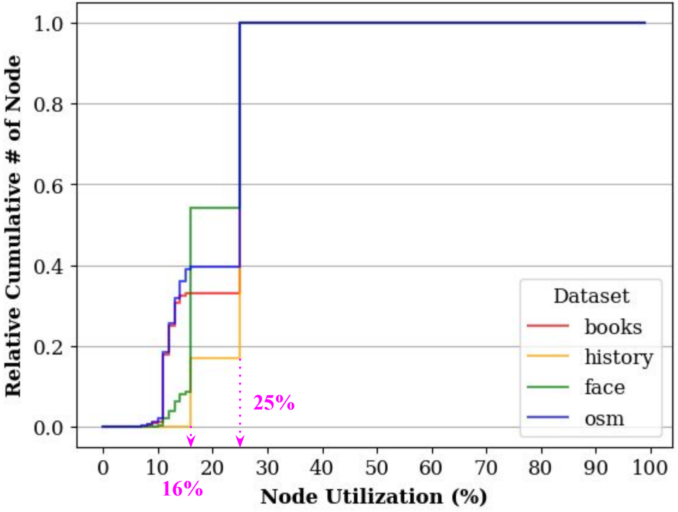
(1) Build tree at first (bulk load)
initial node size: 8

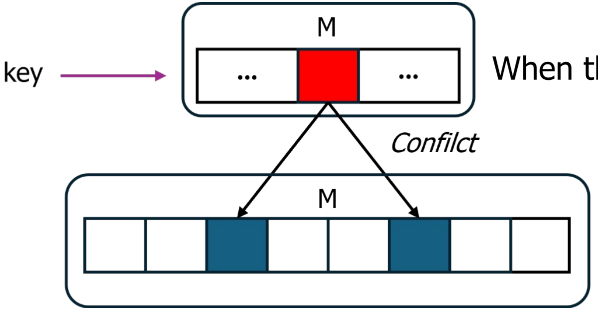
(2) Rebuild (adjust)
gap count: 1,2,5

```
const int BUILD_GAP_CNT = compute_gap_count(size);

node->is_two = 0;
node->build_size = size;
node->size = size;
node->fixed = 0;
node->num_inserts = node->num_insert_to_data = 0;

{
    const int L = size * static_cast<int>(BUILD_GAP_CNT + 1);
node->num_items = L;
```





When the entry is already ***Data*** type

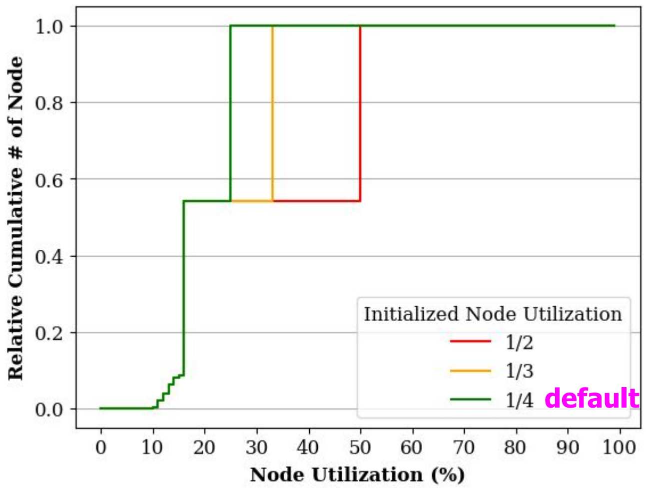
→ Utilization: 2/8 (initial node size) = 25%

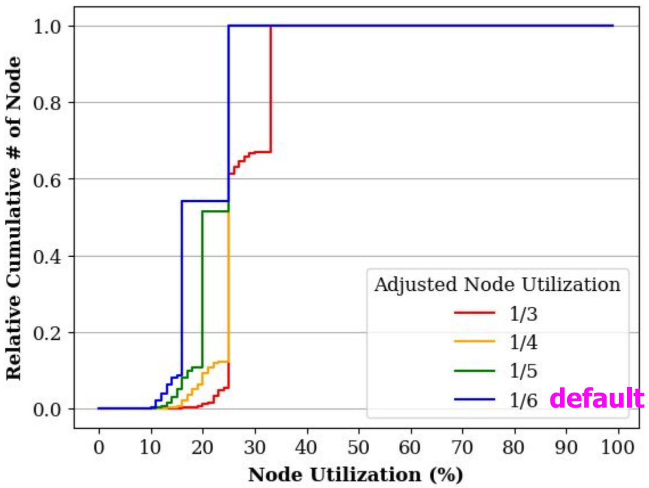
Figure 6: Making "Node" type entry when conflict.

2. Observations

Observed after bulk load **100M** keys
Dataset: **Face (1.6GB)**

1-2) Node Utilization CDF





- The initial node size greater, the utilization lower
- The gap size greater, the utilization lower

2. Observations

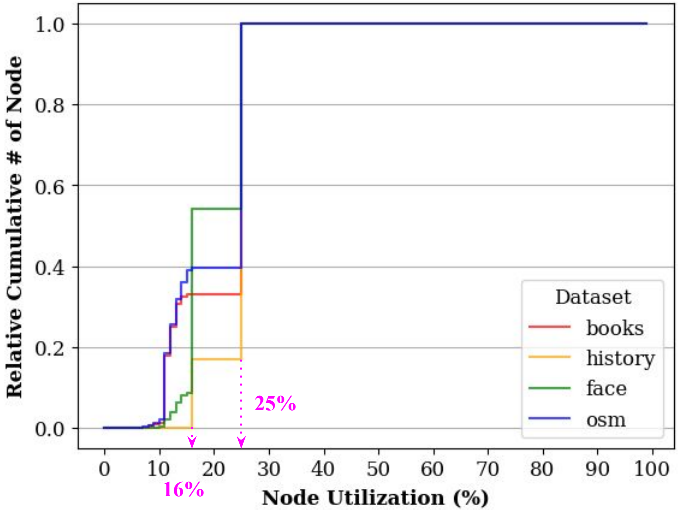
2) Static Gap Count

- It is important to determine **the array length of new node**, because of trade-off between performance and space consumption

- When new node is first created, gaps are hard-coded with **1,2,5** depending on the keys size
  - 1, size ≥ 10^6
  - 2, size ≥ 10^5
  - 5, default

- Almost all nodes have a count lower than 100_000, which means that on average, <u>⅙ utilized</u>

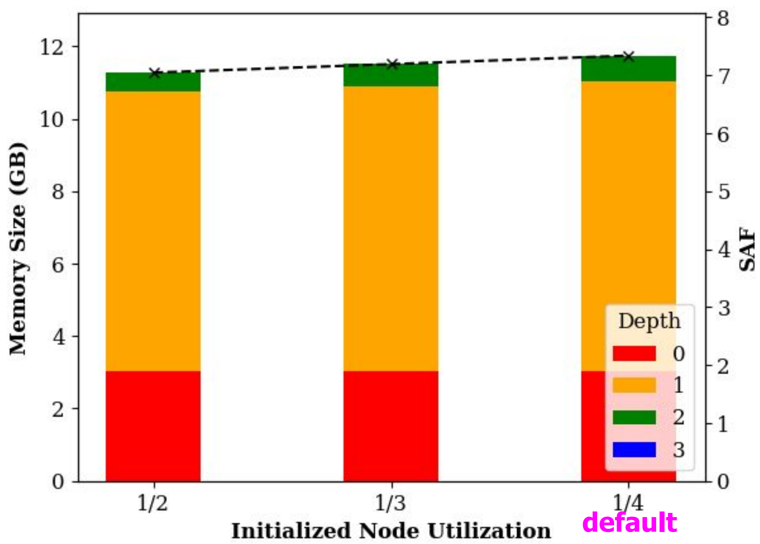
- It has low hotness

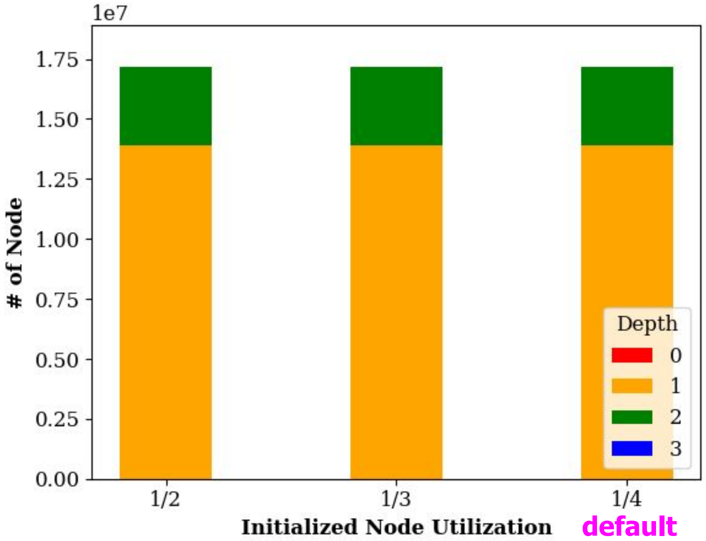


2. Observations

Observed after bulk load **100M** keys
Dataset: **Face (1.6GB)**

3-1) Initialized Node Utilization





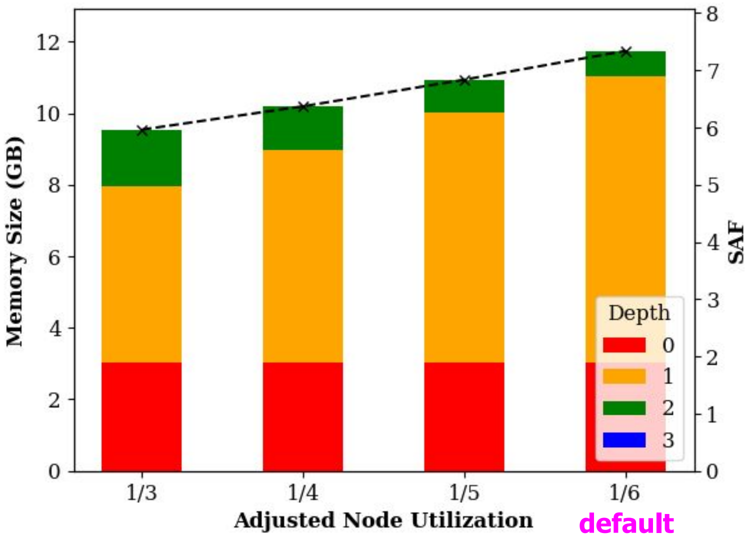
- **Root node** takes up a significant percentage of the total index size
- **Initial node** size doesn't have much impact on overall size
- Large number of nodes, but small percentage of size (MBs)

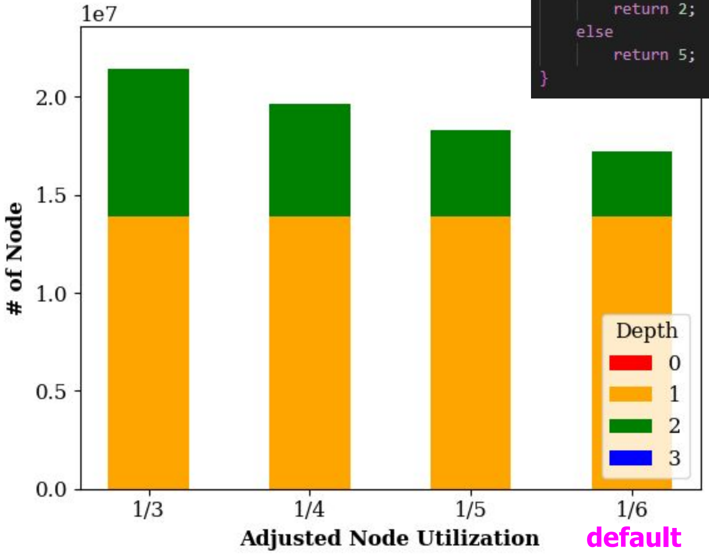
2. Observations

Observed after bulk load **100M** keys
Dataset: **Face**

```
inline int compute_gap_count(int size)
{
    if (size >= 1000000)
        return 1;
    else if (size >= 100000)
        return 2;
    else
        return 5;
}
```

3-2) Adjusted Node Utilization (< 10^5)

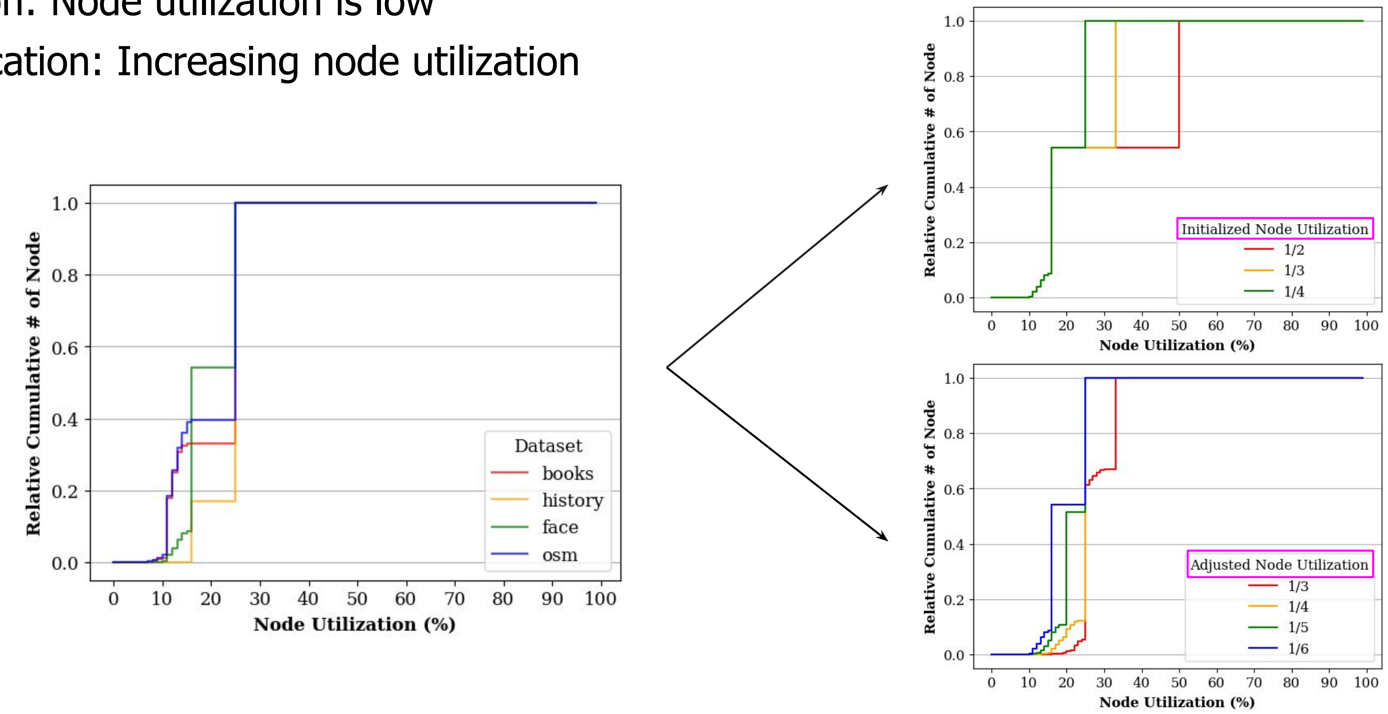




- Node size < 10^5 utilization does not matter
- Node size > 10^5, 10^6 (Root+Depth 1) is important → Need to change this gap count
- There may be some other factors (e.g., α, β, FMCD …)

3. Hypothesis

- Hypothesis: The cause of LIPP's SAF may be <u>array management policy</u> (node utilization)
- Reason: Node utilization is low
- Verification: Increasing node utilization

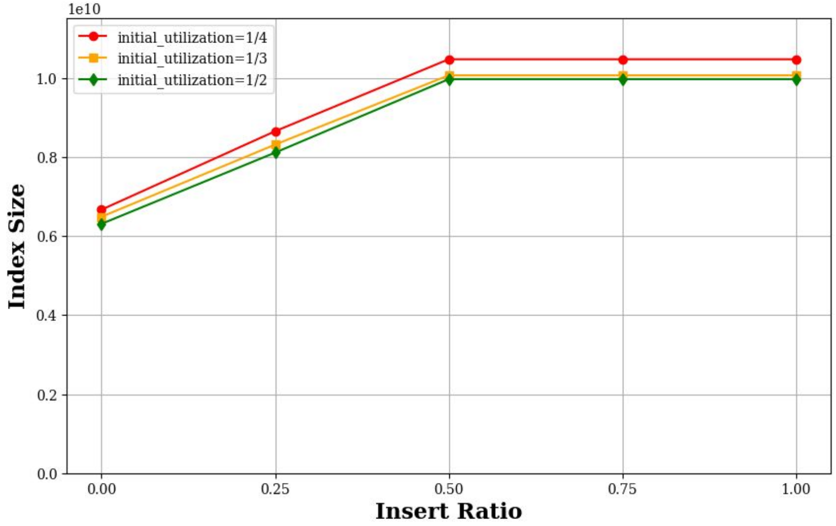


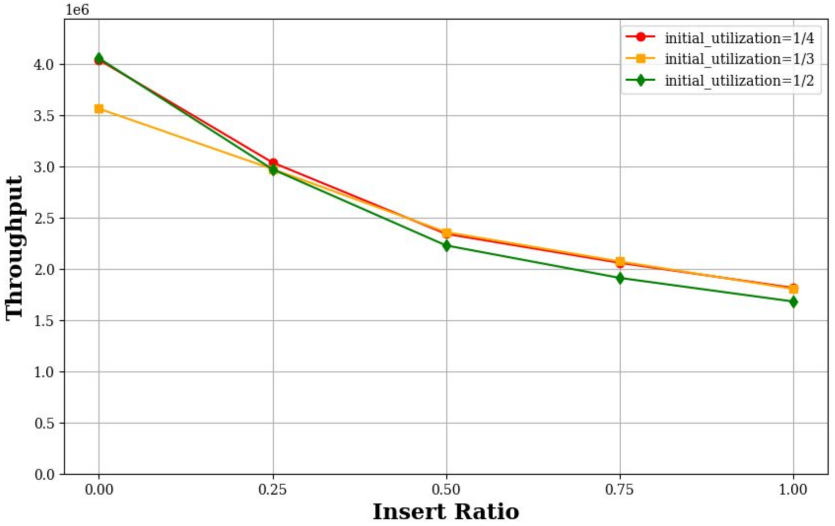
4. Experiments

Observed after bulk load **100M** keys
Dataset: **Face**

Goal : Impact of LIPP's array utilization policy (initial node size)

Observation. The **initial node size** has little effect on index size and read-write performance



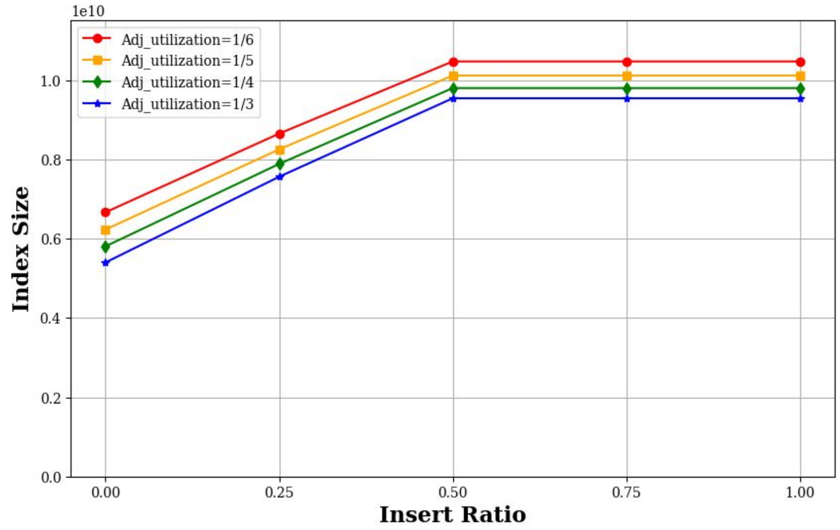


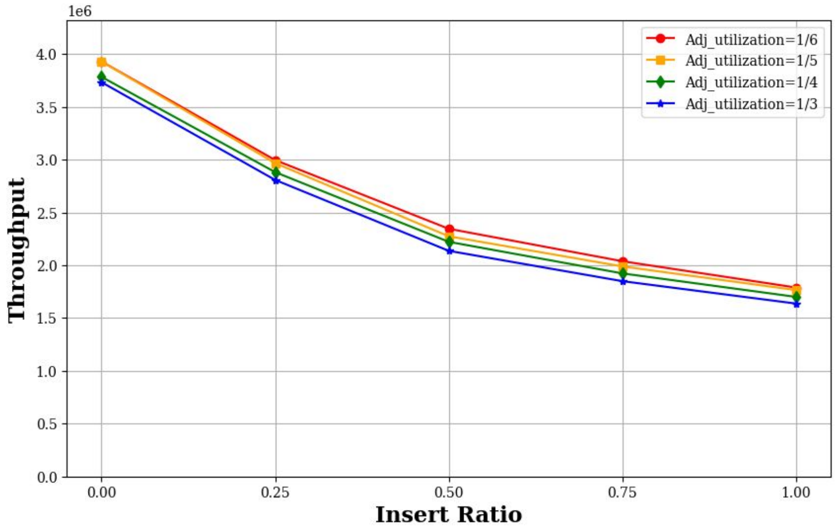
4. Experiments

Observed after bulk load **100M** keys
Dataset: **Face**

Goal : Impact of LIPP's array utilization policy (gap count)

Observation. The **gap count** has a greater impact than the initial node size, but it is not the main cause





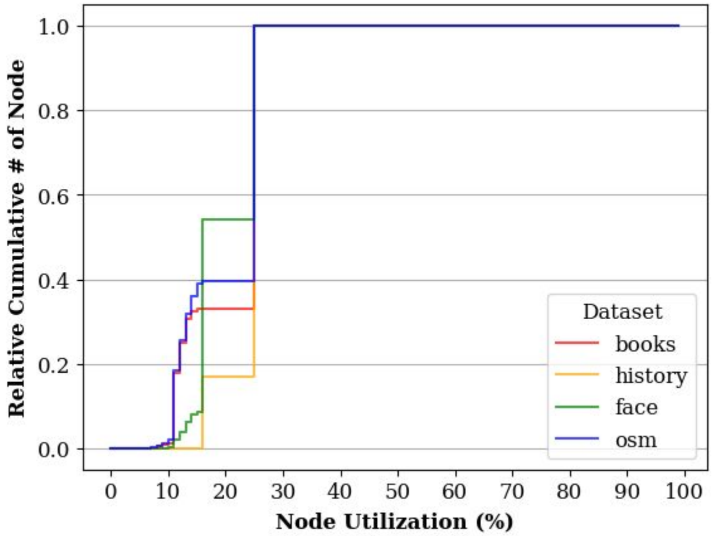
5. Conclusion

Experiment conclusion:

- The issue was not with the initial node size and the utilization of nodes smaller than 10^5.
- The parameters of nodes larger than 10^5 need to be adjusted.
- There may be some other factors (e.g., α, β, FMCD …)

Question: No nodes exceed initial node utilization (25%)

- If fill factor is the problem, 3/8 should also exist!
- But, node with a utilization rate of 3/8 is not observed
  → Probably FMCD, conflict issue or something…



6. Future work

1. A sensitivity analysis into updatable learned index structure
   - **Current Hypothesis: Array management policy**
     - **fill factor: initial node size, gap_count**
   - **New Hypothesis (Expectation)**
     - **rebuilding condition: α, β**
     - **training model: FMCD**
   - **Model : simple linear regression vs kernelized linear regression**
   - Conflict resolving : shifting vs chaining
   - SMO (Structural modification operation) : cost-benefit (fanout tree) vs rebuilding

$$\frac{n.element\_num}{n.build\_num} \geq \beta \quad \beta \text{ is set to 2 by default}$$

$$\frac{n.conflict\_num}{n.element\_num - n.build\_num} \geq \alpha \quad \text{we set the threshold } \alpha = 0.1$$

2. Performance comparison between ALEX and LIPP through size
   - Need understanding of fill factor parameter each indexes

3. Which techniques are appropriate when considering performance versus space?
   - Create a new index based on that analysis
   + Conflict resolving : error-controlled approach (shift-chain hybrid)
   + Concurrency-friendly : semi-ordered

Q&A

Thank you!